Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2022 SCPE. Volume 23, Issues 4, pp. 389–402, DOI 10.12694/scpe.v23i4.2049

A MULTI OBJECTIVE HYBRID COLLISION-FREE OPTIMAL PATH FINDER FOR AUTONOMOUS ROBOTS IN KNOWN STATIC ENVIRONMENTS

KADARI NEERAJA* AND GUGULOTHU NARSIMHA†

Abstract. The most important field of robotics study is path planning. Path planning problem in general is an NP-complete problem. Though several attempts have been made using A*, PRM, RRT, and RRT* these algorithms explore too many nodes in the state space, not completely captured kinematic constraints, and are not optimal in real-time. In this paper, a Multi-Objective Hybrid Collision- free Optimal Path Finder (MOHC-OPF) is proposed which is an attempt to obtain a near-optimal solution by exploring fewer nodes compare to the above existing methods while considering kinematic constraints aiming to generate optimal drivable paths. The empirical study revealed that the proposed algorithm is capable of detecting static obstacles and finding a collision-free nearest-optimal, smooth and safe path to the destination in a static known environment. Multiple criteria, including path length, collision-free, execution time, and smooth path, are used to determine an optimal path.. The proposed algorithm shows efficiency in finding the shortest path length and execution time decreased in 90% of the experiments.

Key words: Path Planning, A^* algorithm, Optimal Path, Collision-Free, Kinematic Constraints.

AMS subject classifications. 68U35, 65Y04, 05C38

1. Introduction. Path planning is the ascertainment of a collision-free path in any given environment, which may often be bestrewed in the real world [1]. Sometimes it is termed as motion planning because it helps to decide on the motion of any object within the working world. An object could be a robot which is autonomous because it utilizes the path-finding algorithm to determine its traversing states in space. A robot is referred to as a mobile robot [2]. The planned path determines if a mobile robot is capable of achieving reliable as well as efficient autonomous navigation. On that account, path planning plays a very crucial role in mobile robot navigation. With the extensive application of mobile robots is leading to research on path planning more popular. Path planning could also be explained as the procedure of dividing the desired path movement into numerous iterative steps to make discrete motions to optimize some metrics.

Environment plays an important role in path planning problems. On the basis of the nature of the environment, path planning could be classified into offline and online path planning. In the former, the data concerning the environment accommodating stationary obstacles is known in advance, termed as static known environment, which is given as input, and thereafter the path is found utilizing the algorithm [3]. While in later, an environment that has static as well as dynamic obstacles is called a dynamic environment, and the robot makes use of sensors or real-time data acquisition equipment to figure out the location of the unknown obstacle that continually moves throughout the environment [4].

In a static and known environment, the robot knows the entire information of the environment before starting its traveling. Hence, the optimal path could be computed offline preceding the movement of the robot.

A path planning algorithm is necessary to be capable of finding a path that is collision-free, smooth, and drivable in real-time. The broadly used path planning algorithms are Sampling-based algorithms and Graph-based algorithms. The Probabilistic Road Map (PRM) [2] and the Rapidly Exploring Random Trees (RRTs) [2, 5] and its variants RRT^* [6] are the archetypal sample-based algorithms. Sampling-based algorithms often provide jerky and dynamically–impossible paths. The typical Graph-based path planning algorithms are A^* [7] and its variants [3, 8, 9, 10, 11, 12, 13, 14, 15], however they are likely to yield pathways that are not smooth and do not adhere to the non-holonomic constraints of the robot or the vehicle. In the case of right-angle

*Research Scholar and Associate professor, JNTUH College of Engineering, Kukatpally, Hyderabad-500090, India (kadari.neeraja @gmail.com).

†Professor, Department of CSE, JNTUH College of Engineering, Kukatpally, Hyderabad-500090, India (narsimha06@gmail.com)

turning angle, the turning motion of the mobile robot is broken down into three steps: slowdown, turning on the spot, and movement. All of this significantly slows down the mobile robot's pace.

In this paper a novel path planning algorithm called A Multi-Objective Hybrid Collision-free Optimal Path Finder (MOHC-OPF) is proposed based on A* to determine an optimal path in a hybrid environment . It is faster, and attains efficiency near to A^*. When the turning angle is right, the mobile robot's turning motion is split into three parts: slowing down, turning on the spot, and moving over fewer nodes. Compared to existing methods, this method takes kinematic constraints into account while making optimal driveable paths.

Our contributions in this paper are as follows:

1. An algorithm named Multi-Objective Hybrid Collision-free Optimal Path Finder (MOHC-OPF) is proposed to find collision-free near-optimal drivable paths in a hybrid i.e. continuous and discrete environment, and obtain efficiency near to A* algorithm.
2. It incorporates kinematic constraints such as orientation and steering angle of the robot to generate shortest, less execution time, and smooth paths apt for real-time environments.
3. It is applied to Simple and Complex static environments. The success rate is 100%.
4. The proposed technique shows efficient results in known static environments.

The rest of the paper is organised as follows. Section 2 examines exisiting path-planning methods for autonomous robots. Section 3 puts forward the proposed methodology and underlying algorithm for path planning in the static known environment. Section 4 includes experimental findings and evaluates the performance of the proposed method. The paper is concluded in Section 5, which also offers guidelines for future work.

2. Related Works. A^* [7] and its variants [3, 8, 9, 10, 11, 12, 13, 14, 15] are a large part of earlier path planning work which produces quick solutions for discrete-state spaces. The non-holonomic limitations of autonomous robots or vehicles make it impossible to use these strategies, which lead to paths that are not smooth. An alternative approach, that can guarantee kinematic feasibility is Sampling based approach in continuous coordinates, like Rapidly Exploring Random Trees (RRTs) [2, 5] and its variants such as RRT* [1, 16].

The configuration space is divided into cells by grid-based planners to map it into the grid form. For path planning, these planners use discrete approaches. As a collection of contiguous cells, they produce a path. However, due to their ineffective search efficiency, Dijkstra [17, 18, 19] and Extended Dijkstra [20] were not suitable for real-time path-planning applications. A* is a well-known Grid-based planner that was inspired by Dijkstra variations [7]. However, A* generates unrealistic paths. These restrictions on the development of grid-based planners have been addressed by showcasing the various A* variants [3, 8, 9, 10, 11, 12, 13, 14, 15]. Some of them are Time efficient A* [12] has shortened the time by reducing the exploration of state space. Iterative-deepening A* (IDA*) [3, 15] addressed large memory requirements issue. Nevertheless, it commonly leads to revisiting the same nodes, lengthening the exploration period and ensuring the optimality of grid-based algorithms like A* up to the grid resolution. However, they generate sharp turn paths which are not suitable for non-holonomic robots or vehicles.

Sampling-based algorithms like the Rapidly-exploring Random Tree (RRT)[2] has been proposed as an effective solution to computationally challenge motion planning problems. Although the RRT method is proven to be able to locate a workable solution rapidly, there is no assurance that it will be the optimal one. The authors developed a novel algorithm called RRT* [1] to get around this restriction. It maintains the same properties of the traditional RRT algorithm in terms of computation of feasible solutions and computational complexity while guaranteeing asymptotic optimality, i.e., virtually certain convergence of the solution returned by the algorithm to an optimal solution.

Larsen et al. describe a system that can plan routes for industrial robots. We compare sampling-based methods like PRM or RRT with methods like genetic algorithms that are based on Computational Intelligence (CI). Both sampling-based planners and genetic algorithms could be shown to be useful for planning the paths of 6-DOF industrial robots. On the one hand, RRT and TRRT are much faster than the genetic planner, but on the other hand, the paths found by the genetic planner are shorter, even though RRT* took 1200s to calculate. In real production situations, it seems that probabilistic planners are better to use because the paths are much smoother [31].

In many research problems, there are often too many things to take into account when making the final choice. These things are basically features, which are variables. The more features there are, the harder it is to see what the training set looks like and then work on it. Sometimes, most of these things are related to each other, which makes them redundant. This is where algorithms that reduce the number of dimensions come in. Dimensionality reduction is the process of getting a set of principal variables from a group of random variables to cut down on the number of variables that need to be considered. It can be split into two parts: selecting features and getting information about them. Gunupudi et.al [32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54] discusses various dimensionality reductions and data transformation techniques used for various learning methods in their proposed research work.

Rybus et al. in thier proposed method creates a path for the arm that avoids collisions and leads to the gripper being in the right place and facing the right way and the spacecraft being in the right attitude. The main benefit of the proposed plan is that it might be possible to reach all of these goals at the same time. Experiments done on the planar air-bearing microgravity simulator (for a flat case) and numerical simulations done with the Monte Carlo method show that the method presented can be used in the real world (for a spatial case). In the planar case, the best path is compared to the path made by the Rapidly-exploring Random Trees (RRT) algorithm, which works in both directions. The proposed method makes a path that is shorter and easier to follow than the RRT method [55].

Santos et al. suggests a way to do multiobjective optimization that takes into account more than one part of motion. The effect of the mass of the manipulator is looked at. Then, the effect of multiple goals, such as the satellite's position, the arm's ability to be moved, and the maximum torque, on the best path is figured out. Also, the position of the end-effector, the avoidance of a collision between the arm and the spacecraft, and the reduction of torque requirements are thought to be goals to be minimised, depending on how uncertain the berthing box is. The numerical procedure uses a machine learning strategy that can learn from both training data and mission tasks. During an inverse kinematics analysis, when the Cartesian position is the input parameter and the estimated joint angle is the output, it is used. This information makes it more likely that the optimization process will end up with the right value for the joint angle. When there are five or more samples, the learning strategy works well to estimate the answer, and the result gets better as more data is added to the analysis. The fact that the numerical experiments looked at many different scenarios, metrics, and parameters proves that the proposed method works and is strong [56].

Tan et al., in their contribution, show a strong model using predictive control-based strategy for following the path of 4WS4WD vehicles when there are disturbances from the outside. Model predictive control (MPC) and control allocation are both parts of the strategy, which has an upper-lower structure. The main goal of this project is to make path tracking more stable and reliable by putting an MPC algorithm in the upper layer. In the offset model, the design of the controller takes into account both general disturbances caused by allocation errors and sudden disturbances caused by an outside force. Using the offset model, you can turn the robustness constraints into a linear matrix inequality to get a robust MPC control law. The Lyapunov stability theorem shows that the control law is stable in situations with more than one disturbance. By comparing the proposed robust algorithm to a similar path-tracking control algorithm and testing it on different types of uneven ground, it was found that the proposed algorithm was able to handle disturbances in the system well.[57]

Tzafestas et al. show a System on Chip (SoC) for autonomous, non-holonomic mobile robots to follow a path. The SoC is made up of a Xilinx Microblaze soft processor core, a parameterized Digital Fuzzy Logic Controller (DFLC) core, and a flow control algorithm. The authors made a fuzzy path tracking algorithm that works with the fuzzy controller. The FPGA board with the SoC was connected to a real differential-drive Pioneer 3-DX8 robot, which was used in field tests to see how well the tracking scheme worked overall. Quantization problems and the limits set by the way the system is set up are also talked about [58].

3. Proposed Path Planning System. The Path planning problem could be stated as "Given a map, find the cost-optimal path from start-state S to a goal-state G subject to the kinematic-constraints that are given by the non-holonomic robot.

The block diagram of path planing system is given in Fig. 3.1. The path finding criteria receives the data such as Path length, duration of Time taken to find a path and path is being smoothened from Optimization Criteria. The four main key elements of path planning system are:

1. *Environment*: A static environment is known in advance and represented as occupancy grid map.

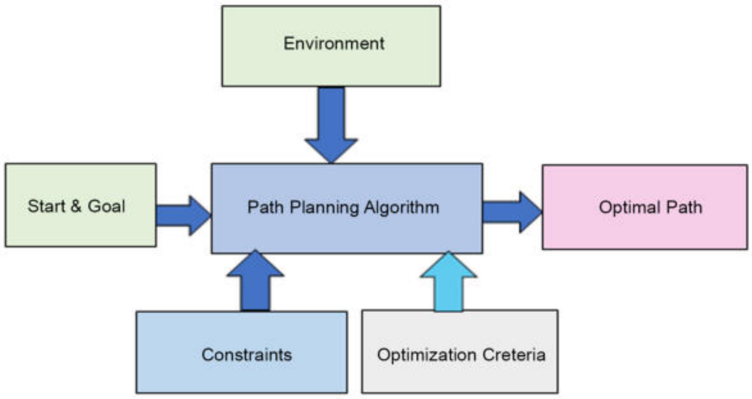

Fig. 3.1: Block Diagram for the proposed path planning system

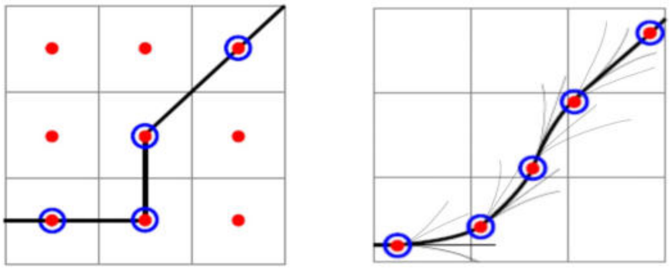

Fig. 3.2: a) Using traditional A* b) Using Proposed Method (MOHC-OPF)

2. *Constraints*: The constraints are kinematic constraints of non-holonomic car-like robot or vehicle.
3. *Optimization criterias*: Path length, Duration of time taken to find a path and Path smoothening.
4. *Path planning algorithm*: It is the main key element of the whole path planning system which finds a solution for a given path planning problem.

In this article an attempt is made to propose a new path planning algorithm named as "A Multi Objective Hybrid Collision-free Optimal Path Finder (MOHC-OPF)". It is a variation of traditional A*.

3.1. The working Principle of MOHC-OPF. The application of the traditional A^* algorithm [3] is restricted to discrete state spaces given in Fig.3.2(a). In case of a simple four or eight connected grid, it would need the robot or vehicle to turn on the spot, but in the presence of nonholonomic constraints it is not possible. The proposed algorithm, MOHC-OPF, is a variation of the traditional A* algorithm to overcome its limitations. It is done by incorporating the kinematics of the car-like robot or vehicle to predict the motion of the robot that is depending on the steering-angle in a continuous search-space. In the proposed algorithm, each continuous state is given by (x, y, θ), where (x, y) is the location and θ is the orientation of the robot or vehicle. It helps the path-finder to opt the right successor state which a nonholonomic robot or vehicle is able to follow. The states are expanded by one of the five steering-actions: maximum-left, left, maximum-right, right and no- steering actions, illustrated in Fig.3.2(b), which results in an arc of a circle with a minimum turning-radius on the basis of the simple car-like robot's or vehicle's constraints. Fig. 3.1.b. illustrates how the MOHC-OPF algorithm opts next states on the basis of these actions.

3.2. The proposed Algorithm MOHC-OPF. The algorithm, MOHC-OPF, uses 2 lists: Open-list and Closed-list. During the searching process they keep track of the states just like in the traditional A*. The open-list (OL) contains the neighbours of states that are already expanded during the search. The closed-list (CL) contains all states that are conclusively processed. The step-by-step description of the algorithm MOHC-OPF is given below:

Output: the near optimum path for an autonomous car-like mobile robot from the given starting position to the desired target position.

Step 1: Initialize Open list (OL) and Close list (CL)

Step 2: Set NODE to be the start state

Step 3: Find the 5 neighbors of NODE for all 5 steering angles. The kinematic model of the robot with a global location of (X, Y, θ) was calculated by using a simple car like model as follows (LaValle, S.M. Planning Algorithms, 2006 [6]).

$$\begin{aligned} x &= u \; cos \; \theta \\ y &= u \; sin \; \theta \\ \psi &= \frac{U_s}{L \; tan \; \theta} \\ \rho &= \frac{L}{tan \; \psi_\theta} \end{aligned} \tag{3.1}$$

where u - denotes action havnig {0,1}, ψ is the steering angle and ρ_{min} is the minimum turning radius and L is distance between the front and rear axles of a simple car.

Step-4: If any of the neighbors is a goal state, then quit.

Step 5: For each neighbor is not lead to collision then Estimate cost of each neighbor using

$$Cost\_function \; f \; = \; g \; + \; h \; + \; additional\_cost \tag{3.2}$$

where g is distance from start node to current node and h is the value the predicted distance to the goal additional cost required for switching heading.

Step 6: Set NODE to the neighbor with the minimum cost_function f value and take the corresponding action on the map. Store the old NODE in a open list along with the f values.

Step 7: Repeat from step 3 until Open list is empty.

The pseudo code of MOHC-OPF Algorithm overview is given in Algorithm 1.

3.3. The Function. The minimum cost estimated value from any-node to the goal-node in the map is the function. It helps reduce the no.of nodes navigated.

Hence, the efficiency of the algorithm is directly impacted by the choice of function. It uses the Euclidean distance.

$$h = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2} \tag{3.3}$$

3.4. Multi Objective functions for Optimization:. In this work multiple objectives are taken into consideration to optimize the found path.

3.4.1. Cost Function. It computes the expense of driving from the current-position to an adjacent-position. This cost f is the sum of the cost from start node to current node(g) , the estimated cost (h) to the goal from current node and the additional cost for changing the angle.

Cost function $f = g + h +$ additional cost for switching orientation.

At each state in the algorithm min-cost next-state is added to the path and it guarantees that the final path is minimum cost.

3.4.2. Path Length. The final path is a collection of path segments $P = P_1, P_2, ..., P_n$.

So, the final path length is the sum of all path segments which connect Start and Goal states through intermediatory states.

$$Path \; Length \; = \; \sum P_i \; where \; i \; = \; 1 \; to \; n \tag{3.4}$$

At each state minimum cost next-state is selected and minimum turning radius curves guarantee near shortest length path.

Algorithm 1 Pseudo code of Algorithm MOHC-OPF

```
Require: Start location x_start, //initial point for robot path
         Goal Location x_goal, //final point for robot path
         Binary Occupancy Grid Map //representation in the form of binary matrix
Ensure: OL = Θ //Initialization of Open List: OL
        CL = Θ //Initialization of Close List: OL
  Pred(x_strt) ⟵ null //initialization of prediction to x_start
  OL.push(x_strt) //Adding current prediction to Open List: OL
  while OL ≠ 0 do //Continue as long as Open List (OL) is not empty
     x ⟵ OL.popMin() //Calculating next feasiable new x
     CL.push(x) //Adding x to Close List
     if round(x) = round(x_goal) then
        continue; //checking if x is destination
     else
        for u ϵ U(x) do //if destination not reached, continue to predict new x to reach destination
           x_next ⟵ f(x, u) //calculating next x point as x_next
           if collision_check(x) = False) & x_next not in CL then //checking if new x do not have an hurdle
              g ⟵ g(x) + l(x, u) //Initialization of Open List: OL
              if (x_next notinOL)or(g < g(x_next)) then //calculating and validating next hop
                 Pred(x_next) ⟵ x //moving robot to next hop
                 g(x_next) ⟵ g //marking next hop
                 h(x_next) ⟵ (x_next, x_goal) //adding to current x to path travelled
                 cost(x_next) ⟵ g(x_next) + h(x_next)+ //calculating cost of travel
                       additional cost for switching orientaion
                 OL.push(x_next, cost(x_next)) //updating final cost
              else//in the case validation fails for new x
                 OL.decreaseKey(x_next) //rolling back to previous x point
              end if
           end if
        end for//predict new x till x_goal isreached
     end if
  end while//repeat this process till x_goal isreached
```

3.4.3. Execution Time. Time taken to execute the proposed method for finding a path. At each node continuous state is rounded off to discrete state to avoid increase in the size of the search graph. We have used tic and toc in MatLab, to measure the performance of our execution.

3.4.4. Smooth Path. In this algorithm at each step next-node is selected based On the kinematic constraints which results in minimum turning radius curves. So, the generated final path is smooth.

4. Experimental results. In MATLAB 2021a on Windows10 64-bit with Intel core i5 NIVIDIA G5 processor extensive experiments were carried out to examine the performance of the proposed method the new Multi Objective Hybrid Collision free Optimal Path Finder (MOHC-OPF) and it was observed that the performance was improved significantly. All simulations were done on various static known maps of sizes 25 X 25, 50 X 50 and 100 X 100 for various start and goal positions.

The performance measures that are considered for performance evaluation of the proposed method are Path-Length-Mean and Average-Execution-Time, Standard-Deviation and Direct-Euclidian Distance between given start and goal locations on the map. Every time it generates same path, a deterministic path planning approach will find the same path when the scenario and parameters are the same, like A*. So, Standard deviation is zero. Optimized path length and execution time are recorded, since the results will be compared with the performance of existing methods for discussion.

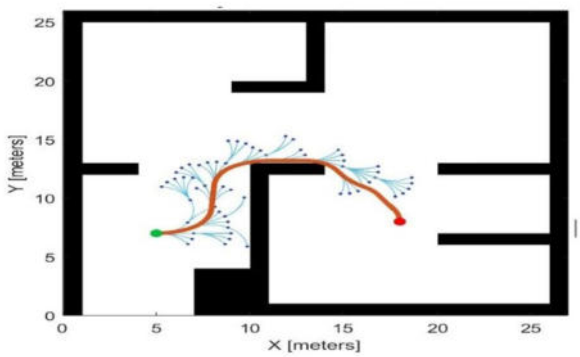

Fig. 4.1: Generated Near optimal path

Table 4.1: Results for Simple map

	Proposed method MOHC-OPF
Path Length Mean	18.7918
Path Length Std	0
Execution Time	0.29964
Direct Path Length	12.1440

Table 4.2: Results for Complex Map

	Proposed method
Path Length Mean	64.5589
Path Length Std	0
Avg Execution Time	0.457936
Direct Path Length	51.856

For the experimentation we have considered three sample environments one is simple and the other two are complex known environments. The criteria for distinguishing simple and complex is taken based on the matrix size and the number of decisions to make. In our case we have taken a matrix size of 25X25 and number of hurdles in the static known environment to 3 as simple map. In the other case, either higher matrix dimension or higher number of hurdles is categorized as complex known static environment map.

4.1. Case study one – Simple map. The proposed method MOHC-OPF was applied on simple static known environment of size 25 X 25 that contains static known obstacles. Fig 4.1. shows generated near optimal path for this environment.

The Table 4.1 shows the Path_Length_Mean and Average-Execution-Time results for Simple map.

4.2. Case Study Two: Complex Map. In case two, a more complex case is presented; the environment contains static known obstacles and it is a complex maze. Fig 4.2. shows a near optimal path generated by MOHC-OPF for this complex environment.

The Table 4.2. shows the Path_Length_Mean and Average-Execution-Time results for Complex map.

4.3. Case Study Three : Package pickup in Warehouse scenario. In case three, a Package pickup in warehouse scenario is presented and contains static known obstacles. Fig 4.3 shows MOHC-OPF generated near optimal path to go to package pickup location based on known static obstacles.

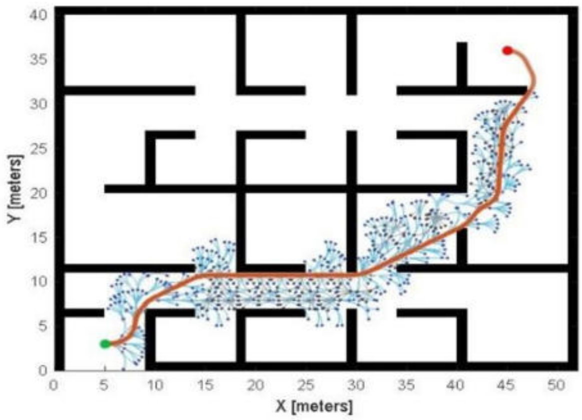

Fig. 4.2: Generated Near optimal path

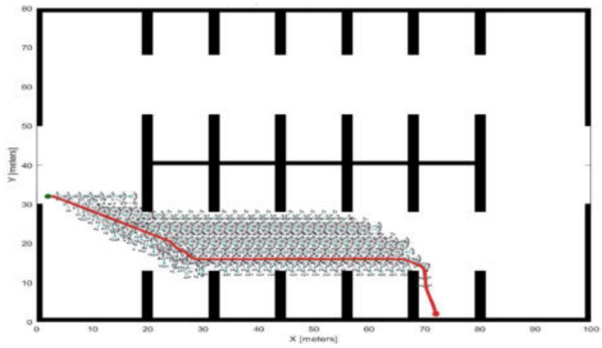

Fig. 4.3: MOHC-OPF- Near optimal path

Table 4.3: Results for Package pickup in warehouse scenario

	Proposed method MOHC-OPF
Path Length Mean	85.09725
Path Length Std	-
Avg Execution Time	0.927327
Direct Path Length	76.174

The Table 4.3. shows the Path_length mean and execution time results for this map.

4.4. Performance Evaluation. In this section, we present the performance efficiency of proposed algorithm MOHC-OPF and compare it with existing methods PRM, RRT and RRT*on the above 3 case studies.

Rigorous experimentation has been done on proposed method and existing methods PRM, RRT and RRT*. The performance comparison has shown in the following tables and graphs. The Proposed Method MOHC-OPF has given optimal path length nearer to A* when compared to PRM, RRT and RRT*. Execution time of MOHC_OPF is comparatively less.

4.4.1. Case Study 1: Simple Static Known Map. The table 4.4 shows the results of 500 iterations of proposed method as well as existing methods PRM, RRT and RRT*. These approaches proposed in the

Table 4.4: Performance efficiency Comparison in Simple Map

Planner/Performance Metric	Path length Mean	Path Length Standard Deviation	Average Execution Time
PRM	37.31334	8.25209	0.312554
RRT	25.77003	9.2276	1.312895
RRT Star	43.45421	3.315177	1.005047
Proposed Method for MOHC_SPF	18.79182	-	0.29964

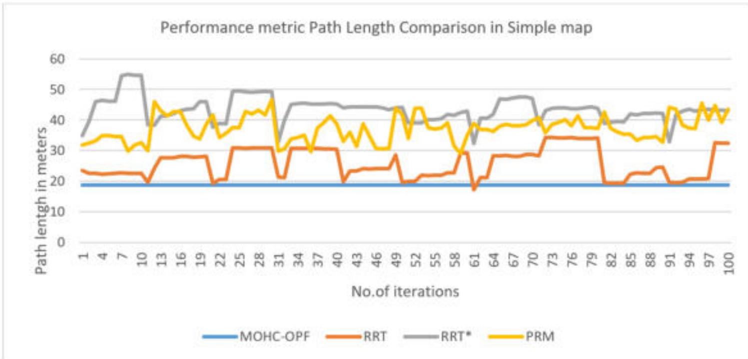

Fig. 4.4: Performance metric Path Length Comparison in Simple map

literature survey are tested with our sample space and the results obtained are compared. It shows the comparison of performance efficiency of the proposed method MOHC-OPF with the existing systems PRM, RRT and RRT* in terms of the performance metrics Path length and Execution Time. These methods are applied on Simple static known map. The values clearly shows that the proposed method has given shortest path length and less execution time taking paths.

Fig. 4.4 displays the performance graph of the proposed method and existing methods PRM, RRT and RRT* across number of iterations and path length. The blue line indicates path lengths generated by the proposed method MOHC-OPF. It generated shortest length paths in almost all iterations out of 100 iterations compared to other existing methods in Simple static known map.

Fig. 4.5 shows the performance graph of the proposed method and existing methods PRM, RRT and RRT* in terms of execution time for 100 iterations. The blue line indicates execution time taken by the proposed method MOHC-OPF. It took very less execution time in almost all iterations of 100 iterations compared to existing methods in Simple static known map.

Case Study 2: Complex Static Known Map Table 4.5 displays the comparison performance of proposed method and also existing methods PRM, RRT and RRT*. The Proposed method and these exiting methods applied 500 times on Complex Static Known Map. The comparison of performance efficiency is shown in the performance metrics Path length and Execution Time. The values clearly exhibits that the proposed method is performing efficiently when the complexity of map is increased.

Fig. 4.6 shows the efficient performance of the proposed method compared to existing methods PRM, RRT and RRT* in 100 iterations. The proposed method performance is indicated by blue line and it generated shortest length paths in all iterations compared to other existing methods in Complex static known map.

Fig. 4.7 displays the performance efficiency in terms of execution time. The proposed method MOHC-OPF is represented by blue line generated less time taking paths in comparison to existing methods PRM, RRT and RRT in almost all 100 iterations in Complex static known map.

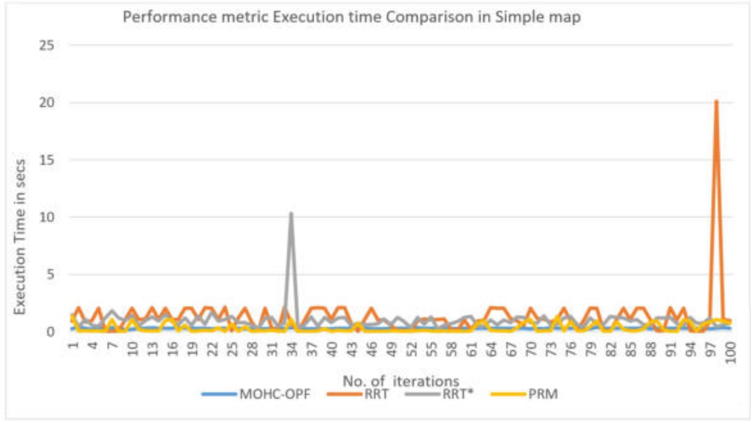

Fig. 4.5: Performance metric Execution time Comparison in Simple map

Table 4.5: Performance efficiency comparison in complex map

Planner/Performance Metric	Path length Mean	Path Length Standard Deviation	Average Execution Time
PRM	71.87536	7.380644	0.815483
RRT	107.5507	9.2276	2.435469
RRT Star	95.49642	24.78597	2.420371
Proposed Method for MOHC_SPF	64.5589	-	0.45973

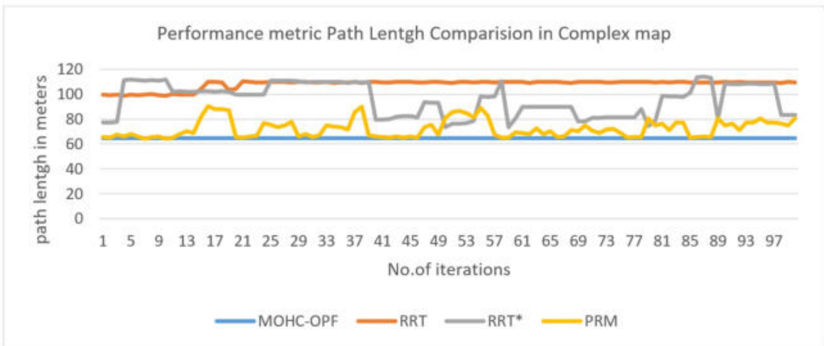

Fig. 4.6: Performance metric Path length Comparison in Complex map

Case Study Warehouse 3: Static Known Map for Package Pickup The Proposed method and the exiting methods PRM, RRT and RRT* applied 500 times on Warehouse scenario. Table 4.6 shows the comparison of performance metrics Path length and Execution Time of the proposed method as well as the existing methods. The proposed method generated shortest length and shortest time taking paths. The proposed method has exhibited efficient performance for this difficult warehouse map.

Fig 4.8 shows the efficient performance of the proposed method compared to existing methods PRM, RRT and RRT* in 100 iterations. The proposed method performance is indicated by blue line and it generated

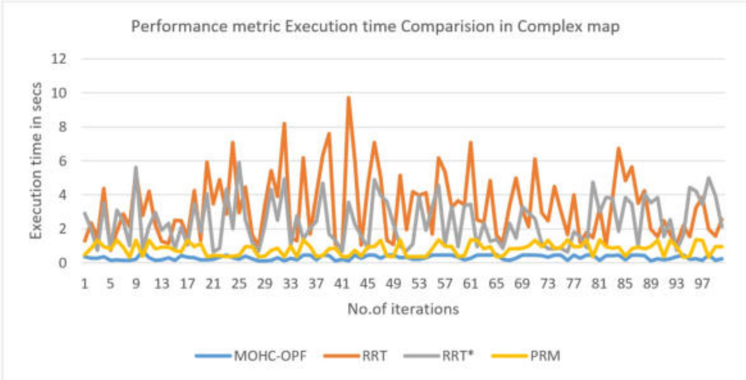

Fig. 4.7: Performance metric Execution time Comparison in Complex map

Table 4.6: Performance efficiency comparison in complex map

Planner/Performance Metric	**Path length Mean**	**Path Length Standard Deviation**	**Average Execution Time**
PRM	114.6131	16.5637	1.283141
RRT	186.1902	33.929	3.722676
RRT Star	95.49642	21.03764	3.489829
Proposed Method for MOHC_SPF	85.09725	-	0.927327

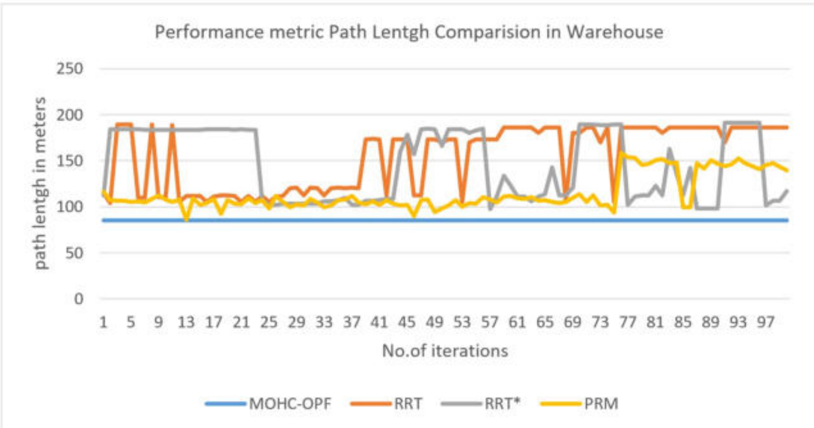

Fig. 4.8: Performance metric Path length Comparison in Warehouse map

shortest length paths in all iterations compared to other existing methods in Complex static known map.

The proposed method MOHC-OPF performance compared to existing methods PRM, RRT and RRT* in 100 iterations as shown in Fig 4.9. The proposed method has given the shortest time taking paths in almost all iterations compared to other existing methods in Warehouse static known map.

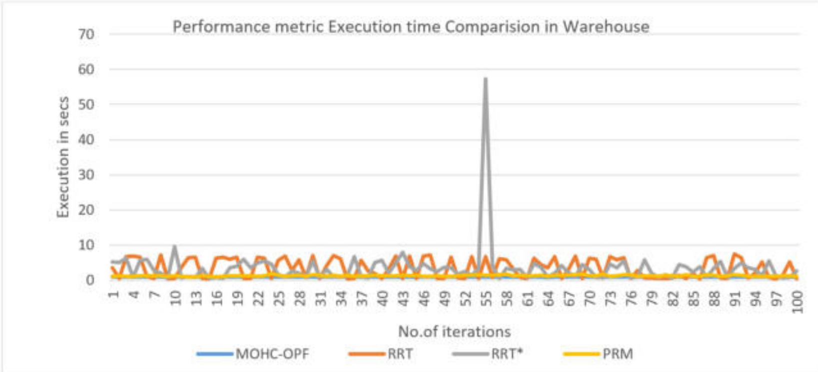

Fig. 4.9: Performance metric Execution time Comparison in Warehouse map

5. Conclusion and future work. In this paper, a new Multi Objective Hybrid Collision-free Optimal Path Finder (MOHC-OPF) is proposed to deal with the optimal path planning for autonomous mobile robots in static known environments. This is fundamentally works like A* but respects kinematic constraints of the robot. Hence, the drivable smooth paths are guaranteed to be generated which are needed for real time scenarios. The success rate for finding a smoother collision free optimized path is 100%. A Multiple Objective Optimization such as Path length, Execution Time, Cost function and Path smoothening are achieved by MOHC-OPF. In the experiments carried out during this work, it is clearly hlobserved that by using the proposed method recorded overall reduction of 15% in path length and 20% in execution-time in our experiments compared to existing methods PRM, RRT and RRT*. The proposed method has shown higher performance efficiency in complex environments compared to these existing methods

The proposed method MOHC-OPF can be adopted to environments which includes dynamic obstacles in future. It can also be applied to higher dimension environments.

REFERENCES

[1] S. Karaman, E. Frazzoli , *Incremental sampling-based algorithms for optimal motion planning*, in Proc. Robotics: Science and Systems (RSS), 2010

[2] 5. S. Karaman, E. Frazzoli, LU-*Optimal kinodynamic motion planning using incremental sampling-based methods*, in Proc. IEEE Conf. on Decision and Control (CDC), Dec. 2010.

[3] S. M. LaValle, *SRapidly-Exploring Random Trees: A New Tool for Path Planning*, 1998. Available online: http://lavalle.pl/papers/Lav98c.pdf.

[4] I. Noreen, A. Khan, Z. Habib, *A Optimal Path Planning using Memory Efficient A**, In Proceedings of the IEEE International Conference on Frontiers of Information Technology, Islamabad, Pakistan, 19–21 December 2016; pp. 142–146.

[5] P. E. Hart, N. J. Nilsson and B. Raphael, *A Formal Basis for the Determination of Minimum Cost Paths*, in IEEE Trans. Syst. Sci. Cybern. 1968, 4, 100–107.

[6] S.M. LaValle, *Planning Algorithms*, Cambridge University Press: Cambridge, UK, 2006. (1983), pp. 275–283.

[7] C.W. Warren , *Fast path planning using modified A* method*, In Proceedings of the IEEE International Conference on Robotics and Automation, Atlanta, GA, USA, 2–6 May 1993..

[8] L. Cheng, C. Liu, B. Yan, *Improved hierarchical A-star algorithm for optimal parking path planning of the large parking lot*, In Proceedings of the 2014 IEEE International Conference on Information and Automation (ICIA), Hailar, Hulun Buir, China, 28–30 July 2014; pp. 695–698.

[9] R. Korf, *Depth First Iterative Deepening: An Optimal Admissible Tree Search*, AI Journal 27 (1): 97-109, 1985.

[10] František. Ducho, Andrej. Babinec, Martin. Kajan, Peter. Beno, Martin. Florek, Tomáš. Fico, Ladislav. Jurišica, *Path planning with modified A star algorithm for a mobile robot* , In Proceedings of the Elseveir, International Conference on the Modelling of Mechanical and Mechatronic Systems MMaMS 2014.

[11] Y. Zhang, G. Tang, L. Chen, *Improved A* Algorithm For Time-dependent Vehicle Routing Problem*, In Proceedings of the 2012 International Conference on Computer Application and System Modeling, ICCASM 2012, Taiyuan, China, 27–29 July 2012.

[12] C. Liu, Q. Mao, X. Chu, *An improved A-Star algorithm considering water current, traffic separation and berthing for*

vessel path planning, Applied Sciences, vol. 9, no. 6, pp. 1–17, 2019.

[13] Guruji. Akshay Kumar, Agarwal. Himansh, D.K. Parsediya, *A Time-efficient A* Algorithm for Robot Path Planning*, Procedia Technology, 23, p144–149.

[14] Ismail, AL-Taharwa, Alaa Sheta, Mohammed Al-Weshah, *A mobile robot path planning using genetic algorithm in static environment*, Journal of Computer Science 4.4 (2008): 341-344.

[15] R. Korf, *Optimal Path Finding Algorithms*, Search in AI. Edited by Kanal, L., and Kumar, V., Springer Verlag, Symbolic Computation: 200-222, 1988.

[16] B. Hu, Z. Cao, M. Zhou , *An Efficient rrt-based framework for planning short and smooth wheeled robot motion under kinodynamic constraints*, IEEE Transactions on Industrial Electronics, vol. 68, no. 4, pp. 3292–3302, 2021.

[17] M. Barbehenn , *A Note on the Complexity of Dijkstra's Algorithm for Graphs with Weighted Vertices* , IEEE Transactions on Computers, Vol. 47, No.2, 1998.

[18] T. Dudi, R. Singhal,R. Kumar , *Shortest Path Evaluation with Enhanced Linear Graph and Dijkstra Algorithm*, In Proceedings of the 2020 59th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE); 2020; pp. 451–456.

[19] A. Ammar, H. Bennaceur,I. Chaari, A. Koubaa, M. Alajlan, *Relaxed Dijkstra and A* with linear complexity for robot path planning problems in large-scale grid environments*, Soft Comput. 2015, 20, 1–23.

[20] O.A. Gbadamosi, D.R. Aremu , *Design of a Modified Dijkstra's Algorithm for finding alternate routes for shortest-path problems with huge costs*, In Proceedings of the 2020 International Conference in Mathematics, Computer Engineering and Computer Science (ICMCECS), Lagos, Nigeria, 18–21 March 2020; pp. 1–6.

[21] G. Qing, Z. Zheng, X. Yue , *Path-planning of automated guided vehicle based on improved Dijkstra algorithm*, In Proceedings of the 2017 29th Chinese control and decision conference (CCDC), Chongqing, China, 28–30 May 2017; pp. 7138–7143.

[22] M. Noto, K. Univ, H. Sato , *A Method for the Shortest Path Search by Extended Dijkstra Algorithm*, In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Nashville, TN, USA, 8–11 October 2000; Volume 3, pp. 2316 – 2320.

[23] X. Cui, H. Shi , *A* - based Pathfinding in Modern Computer Games*, In Proceedings of the IEEE International Conference on Robotics and Automation, Atlanta, GA, USA, 2–6 May 1993.

[24] L. KavrakiP. M. Gibson, *Probabilistic roadmaps ˇ for path planning in high-dimensional configuration spaces*, IEEE Trans. on Robotics and Automation.

[25] O. Souissi, R. Benatitallah, D. Duvivier, A. Artiba, N. Belanger, P. Feyzeau, *Path planning: A 2013 survey*, In Proceedings of the 2013 International Conference on Industrial Engineering and Systems Management (IESM), Rabat, Morocco, 28–30 October 2013; pp. 1–8.

[26] T. Mac, C. Copot, D. Tran, R. Keyser, *approaches in robot path planning: A survey*, Robot. Auton. Syst. 2016, 86.

[27] Liu, Shuang, Sun. Dong, *Minimizing Energy Consumption of Wheeled Mobile Robots via Optimal Motion Planning.* IEEE/ASME Transactions on Mechatronics-2014, 19(2), p401–411.

[28] J. Latombe, *Motion planning: A journey of robots, molecules, digital actors, and other artifacts*, Int'l J. of Robotics Research, vol. 18, no. 11, pp. 1119–1128, 1999..

[29] H.M. Choset, *Principles of Robot Motion: Theory, Algorithms, and Implementation*, The MIT Press: Cambridge, MA, USA, 2005.

[30] B. Chen, G. Quan, *NP-Hard Problems of Learning from Examples*, In Proceedings of the 2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery, Jinan, China, 8–20 October 2008; Volume 2, pp. 182–186.

[31] L. Larsen, J. Kim, M. Kupke, A. Schuster, *Automatic Path Planning of Industrial Robots Comparing Sampling-based and Computational Intelligence Methods*, Procedia Manufacturing, Volume 11, 2017, Pages 241-248, ISSN 2351-9789.

[32] G. R. Kumar. M. Nimmala, G. Narsimha, *An Approach for Intrusion Detection Using Novel Gaussian Based Kernel Function*, Journal of Universal Computer Science, Volume 22, Issue 4, 2016, pp 589-604 ISSN: 0948-6968

[33] G. R. Kumar. M. Nimmala, G. Narsimha, *Intrusion Detection – A Text Mining Based Approach", International Journal of Computer Science and Information Security (IJCSIS)*, Special issue on "Computing Application", Volume 14, 2016, pp 76-88

[34] G. R. Kumar. M. Nimmala, G. Narsimha, *A Novel Similarity Measure for Intrusion Detection using Gaussian Function*, Technical Journal of the Faculty of Engineering, TJFE, Volume 39, Issue 2, 2016, pp 173-183

[35] G. R. Kumar. M. Nimmala, G. Narsimha, *CLAPP: A self constructing feature clustering approach for anomaly detection*, Future Generation Computer Systems, Volume 74, 2017, Pages 417-429

[36] G. R. Kumar. M. Nimmala, G. Narsimha, *A Feature Clustering Based Dimensionality Reduction For Intrusion Detection (FCBDR)*, IADIS International Journal on Computer Science & Information Systems . 2017, Vol. 12 Issue 1, p26-44.

[37] G. Narsimha, G. R. Kumar. M. Nimmala, *Optimising business intelligence results through strategic application of software process model*, Int. J. of Intelligent Enterprise, 2017 Vol.4, No.1/2, pp.128 – 142

[38] K. R. Gunupudi. M. Nimmala, G. Narsimha, *An Evolutionary Feature Clustering Approach for Anomaly Detection Using Improved Fuzzy Membership Function: Feature Clustering Approach for Anomaly Detection.* International Journal of Information Technology and Web Engineering (IJITWE), 14(4), 19-49. doi:10.4018/IJITWE.2019100102

[39] G. Narsimha, K. R. Gunupudi. M. Nimmala, *UTTAMA: An Intrusion Detection System Based on Feature Clustering and Feature Transformation*, Foundations of Science, 2019, https://doi.org/10.1007/s10699-019-09589-5

[40] K. R. Gunupudi. M. Nimmala, G. Narsimha, *An Evolutionary Feature Clustering Approach for Anomaly Detection Using Improved Fuzzy Membership Function: Feature Clustering Approach for Anomaly Detection.* International Journal of Information Technology and Web Engineering, vol. 14, no. 4, Oct. 2019, pp. 19–49. DOI.org (Crossref), https://doi.org/10.4018/IJITWE.2019100102.

[41] G. Narsimha, G. R. Kumar. M. Nimmala, *Optimising business intelligence results through strategic application of software process model*, International Journal of Intelligent Enterprise, Jan 2017, Vol. 4, Issue 1-2, pp. 128-142

[42] G. R. Kumar. M. Nimmala, G. Narsimha, *An approach for Intrusion Detection using Text Mining Techniques*. In Proceedings of the The International Conference on Engineering & MIS 2015 (ICEMIS '15). ACM, New York, NY, USA, , Article 63 , 6 pages. DOI: http://dx.doi.org/10.1145/2832987.2833076

[43] G. Narsimha, G. R. Kumar. M. Nimmala, *Strategic Application of Software Process Model to Optimize Business Intelligence Results*. In Proceedings of the The International Conference on Engineering & MIS 2015 (ICEMIS '15). ACM, New York, NY, USA, , Article 44 , 6 pages. DOI: http://dx.doi.org/10.1145/2832987.2833053

[44] G. Narsimha, G. R. Kumar. M. Nimmala, *Intrusion Detection Using Text Processing Techniques: A Recent Survey*. In Proceedings of the The International Conference on Engineering & MIS 2015 (ICEMIS '15). ACM, New York, NY, USA, , Article 55 , 6 pages. DOI: http://dx.doi.org/10.1145/2832987.2833067

[45] G. Narsimha, G. R. Kumar. M. Nimmala, *An improved k-Means Clustering algorithm for Intrusion Detection using Gaussian function*. In Proceedings of the The International Conference on Engineering & MIS 2015 (ICEMIS '15). ACM, New York, NY, USA, , Article 69 , 7 pages. DOI: http://dx.doi.org/10.1145/2832987.2833082

[46] G. Narsimha, G. R. Kumar. M. Nimmala, *An approach for intrusion detection using fuzzy feature clustering*, 2016 International Conference on Engineering & MIS (ICEMIS), Agadir, 2016, pp. 1-8. doi: 10.1109/ICEMIS.2016.7745345

[47] G. Narsimha, G. R. Kumar. M. Nimmala, *Text mining based approach for intrusion detection*, 2016 International Conference on Engineering & MIS (ICEMIS), Agadir, 2016, pp. 1-5. doi: 10.1109/ICEMIS.2016.7745351

[48] G. Narsimha, G. R. Kumar. M. Nimmala, *Design of novel fuzzy distribution function for dimensionality reduction and intrusion detection*, 2016 International Conference on Engineering & MIS (ICEMIS), Agadir, 2016, pp. 1-6.

[49] M. Nimmala, G. R. Kumar. G. Narsimha, *Optimization of Access Points in Wireless Sensor Network: An Approach towards Security*, Intelligent Systems in Cybernetics and Automation Theory, 2015, Volume 348, pp 299-306

[50] G. R. Kumar. M. Nimmala, G. Narsimha, *Evolutionary approach for intrusion detection*, 2017 International Conference on Engineering & MIS (ICEMIS), Monastir, 2017, pp. 1-6. doi: 10.1109/ICEMIS.2017.8273116

[51] M. Nimmala, G. R. Kumar. G. Narsimha, *A fuzzy measure for intrusion and anomaly detection*, 2017 International Conference on Engineering & MIS (ICEMIS), Monastir, 2017, pp. 1-6. doi: 10.1109/ICEMIS.2017.8273113

[52] A. Cheruvu, G. R. Kumar. M. Nimmala, G. Narsimha, *Feature Clustering for Anomaly Detection Using Improved Fuzzy Membership Function*. In Proceedings of the Fourth International Conference on Engineering & MIS 2018 (ICEMIS '18). ACM, New York, NY, USA, Article 35, 9 pages. DOI: https://doi.org/10.1145/3234698.3234733

[53] A. Shadi, M. Nimmala, G. R. Kumar. G. Narsimha, *A recent survey on challenges in security and privacy in internet of things*. In Proceedings of the 5th International Conference on Engineering and MIS (ICEMIS '19). ACM, New York, NY, USA, Article 25, 9 pages. DOI: https://doi.org/10.1145/3330431.3330457

[54] K. R. Gunupudi, M. Nimmala, G. Narsimha, *Similarity function for intrusion detection*. In Proceedings of the 5th International Conference on Engineering and MIS (ICEMIS '19). ACM, New York, NY, USA, Article 28, 4 pages. DOI: https://doi.org/10.1145/3330431.3330460

[55] T. Rybus, M. Wojtunik, F. L. Basmadji, *Optimal collision-free path planning of a free-floating space robot using spline-based trajectories*. Acta Astronautica, Volume 190, 2022, Pages 395-408, ISSN 0094-5765, https://doi.org/10.1016/j.actaastro.2021.10.012.

[56] R. R. Santos, D. A. Rade, I. M. da Fonseca, *A machine learning strategy for optimal path pl*. Acta Astronautica, Volume 191, 2022, Pages 41-54, ISSN 0094-5765, https://doi.org/10.1016/j.actaastro.2021.10.031.

[57] Q. Tan, C. Qiu, J. Huang, Y. Yin, X. Zhang, H. Liu, *Path tracking control strategy for off-road 4WS4WD vehicle based on robust model predictive control*. Robotics and Autonomous Systems, Volume 158, 2022, 104267, ISSN 0921-8890 https://doi.org/10.1016/j.robot.2022.104267.

[58] S.G. Tzafestas, K.M. Deliparaschos, G.P. Moustris, *Fuzzy logic path tracking control for autonomous non-holonomic mobile robots: Design of System on a Chip*. Robotics and Autonomous Systems, Volume 58, Issue 8, 2010, Pages 1017-1027, ISSN 0921-8890, https://doi.org/10.1016/j.robot.2010.03.014.

Edited by: Vinoth Kumar
Received: Aug 19, 2022
Accepted: Nov 8, 2022