Georgios Tsamis
Candidate of M.Sc. in Advanced Product Design Engineering & Manufacturing

«Multi-Objective Reward-Based Algorithms for the Complete Coverage Path Planning Problem on Arbitrary Grids»

ABSTRACT This paper addresses the complete coverage path planning (CCPP) problem in grid-like environments. The proposed work introduces a new multi-objective approach for the development of algorithms that guide the motion of an autonomous vehicle in real time, with the goal to cover every accessible cell in an arbitrary rectangular grid environment. The grid contains accessible cells and obstacles and its configuration is ex ante unknown to the algorithm. The method combines multiple objectives in a novel Reward function that assigns values to the neighbouring cells. By seeking the maximum positive Reward at each step, the algorithm navigates an autonomous vehicle to explore and completely cover every cell of a grid environment. The proposed method also includes a "Checkpoint" logic that guides the backtracking in order to ensure complete coverage in every case. The algorithm includes exploitation and exploration features to operate in a learning fashion, as the vehicle gradually discovers and accounts for the disposition of obstacles and accessible areas in its vicinity. In addition, the algorithm is optimized using different techniques, including an off-line learning-based methodology. Simulation tests demonstrate that the algorithm is computationally efficient and performs effectively against different unknown environments.

Keywords: Complete Coverage Path Planning, Multi-Objective Path Planning, On-line Path Planning

1. Introduction

Recent advances in technological and engineering fields, coupled with new business models in mobility-related sectors such as transport, distribution or surveillance, have driven the rapidly growing use of autonomous vehicles (AVs) over the last decade [1]. Autonomous vehicles of various types are developed for and deployed in an ever-expanding range of applications and environments, achieving increasing levels of performance, efficiency and safety, with no or minimal human guidance.
Automated navigation, and path planning in particular, is an important part of the autonomy features of any AV. A part of research efforts in AVs goes into the development and implementation of computationally efficient algorithms for different path planning problems.
The present work addresses a sub-category of navigation challenges for AVs, namely the complete coverage path planning (CCPP) problem, where the vehicle is tasked with surveying a known or unknown area by physically moving to every point in succession. Applications of CCPP include situations where the autonomous vehicle undertakes to explore and map an unknown area, patrol an area for disaster detection and assessment (for example in case of wildfire, flood etc.), scan the seabed to locate items of interest (wreckage debris, minerals, life forms etc.) or simply clean the floor of a building ([2],[3]).
In terms of computational complexity, solving the CCPP is an open problem and has been approached with various methods and techniques. Systematic reviews and surveys of path planning algorithms to automate the motion of autonomous vehicles can be found in [4] and [5]. Adopting the classification used in these surveys, the present work falls under in the category of Approximate Cellular Decomposition methods with rectangular grids [6]. Earlier research in the same thematic area includes the work of Yakoubi and Laskri [7] who use Genetic Algorithms to generate paths. Gajjar et al. [8] developed a cost-based CCPP algorithm, which navigates the AV by assigning cost val-

ues to the neighboring cells. Liu et al. [9], Zhu et al. [10], Xu et al. [11] and Muthugala et al. [12] implement and modify a bio-inspired neural network model that creates an artificial potential field that guides the AV to completely cover a grid environment. Shen et al. [13] worked on training a deep learning model to solve the CCPP problem.

The present work introduces a novel Reward-based methodology to solve the CCPP problem in real time. Section 2 provides a formal definition of the CCPP problem addressed, including a model of the idealized environment and of the motion of the AV. The main contribution of the work is the heuristic arguments and mathematical shaping of a multi-objective Reward function, on one hand, and of a "Checkpoint" logic that ensures no cell is missed, on the other. These are detailed in Section 3. Section 4 presents the results of simulation tests when applying the algorithm against different settings.

Section 5 features an off-line optimization procedure for the algorithm's parameters utilizing Genetic Algorithms. Section 6 introduces an adaptive approach to tuning the algorithm's parameter μ , utilizing machine learning. The concluding Section 7 summarizes the main findings and Section 8 discusses directions for further research.

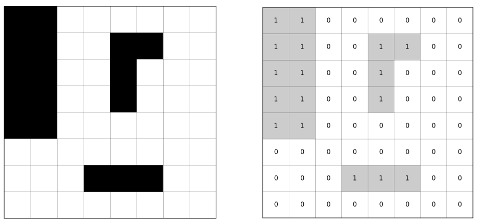
2. Statement of the Problem

The approach focuses on the two-dimensional version of the CCPP problem, as in most cases the AV has to cover a basically planar, two-dimensional environment (for example in the case of a cleaner rover robot). It is assumed that the 2D environment which the vehicle has to navigate is unknown beforehand; therefore the algorithm must run recursively in real time and include both exploration and exploitation features. At each step, a sensor system provides information about nearby objects (i.e. cells) within a limited local range; a computationally inexpensive path planning algorithm operates on this information in real time.

By using the approximate cell decomposition, any arbitrary 2D environment is partitioned as a grid of rectangular cells; the size of cells depends on the vehicle's area of effect (i.e. the field of view of the AV, the sweeping area of a cleaner rover robot etc.) The cells can be classified into "obstacles" (inaccessible) or "free" (accessible), hence the aim of a CCPP algorithm is to navigate the environment by moving into every "free" cell effectively and efficiently. In this sense, the CCPP problem is a sequential optimization process that builds the path (sequence of "free" cells) that achieves minimum cost (typically expressed as a combination of metrics such as path length, energy consumption, number of turns etc.) while respecting a set of constraints (i.e. the imperative to avoid obstacles). Ideally, the solution should use limited prior information (partial knowledge) about the environment). Here we assume that the layout of the environment is completely unknown at start.

As shown below, the algorithm manipulates and operates on matrices. Thus the extension to the 3D case is mostly straightforward. In the 2D case examined here, the environment and the motion of the vehicle are mathematically modelled as follows.

Environment The 2D environment is a two-dimensional grid with square-shaped cells, modelled by a two-dimensional matrix that is referred to as Q_{env}. In the matrix, a "Free" cell is represented with an entry of 0, while an "obstacle" cell with a 1. This matrix remains unknown to the algorithm.



1	1	0	0	0	0	0	0
1	1	0	0	1	1	0	0
1	1	0	0	1	0	0	0
1	1	0	0	1	0	0	0
1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0
0	0	0	0	0	0	0	0

Figure 1: An example environment and the corresponding Q_{env} matrix.

Sensor System's Area of Effect The sensor system is assumed to cover the 7×7 sub-grid of cells centered at the current position of the vehicle. This matrix is refreshed at every step. Notably, as a result of the algorithm operating using this limited local information, the computation effort at each step is independent of the overall size (dimension) of the grid. The total computing cost depends, of course, on the topology of the environment, which determines the number of steps required for complete coverage (including the need for backtracking motion).

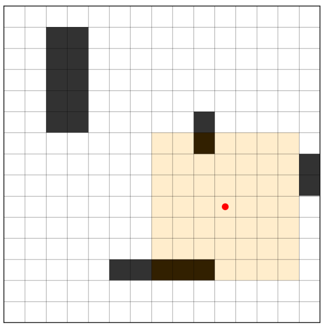


Figure 2: Visualization of the area of effect of the assumed sensor system.

(Current) Known Environment The subset of the 2D environment which the sensor system has already detected as the AV progresses. This is also modelled as a two-dimensional matrix containing 0s or 1s and is referred to as $\hat{Q_{env}}$. The $\hat{Q_{env}}$ matrix is initialized with 1 values. Each time the vehicle discovers a free cell, the value of the corresponding entry is flipped to zero - thus this matrix is updated at each step.

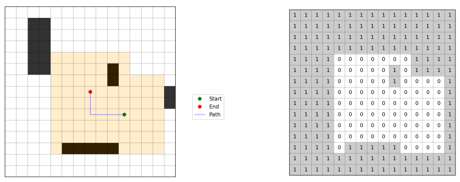


Figure 3: An example environment and the corresponding $\hat{Q_{env}}$ matrix after a few steps.

Vehicle's Motion The presented approach assumes (but is not restricted to) a taxi-cab movement for the AV, also known as 4-N movement. On the square grid, the vehicle is able to move into any of the four neighbouring (4-N) cells. The methodology can be readily extended to include diagonal (8-N) movement, although this is beyond the scope of the present work.

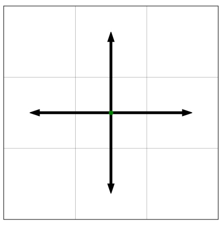


Figure 4: 4-N Movement.

Covered Areas Covered areas are modelled using a two-dimensional matrix, referred to as Q_{cov}. It initially contains only zeros, and is refreshed at every step the vehicle takes. Each element is a positive integer representing the number of times the corresponding cell has been visited.

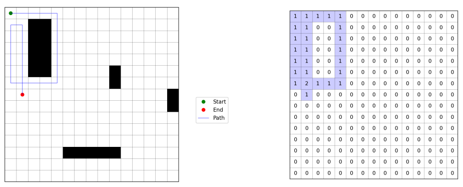


Figure 5: Example of environment and path, and the corresponding Q_{cov} matrix.

The featured methodology aims to solve the online CCPP problem. Thus, the problem can be stated as:

Goal 1. *Generate and follow a path such that each cell in Q_{env} with $Q_{env,cell} = 0$ will have $Q_{cov,cell} \geq 1$, given the information in $\hat{Q_{env}}$.*

3. Heuristics and the Reward Function

Reward Function The approach shapes the CCPP as a recursive optimization problem. Taking a multi-objective view, a Reward function brings together the environment's characteristics and a number of parameters. The algorithm assigns Reward values to the cells adjacent to the current AV position, chooses the cell with the maximum positive Reward to step into next, and repeats this procedure until every cell is covered. By manipulating the parameters of the Reward model, the algorithm can adopt different behaviours, such as avoiding excessive turns, moving into the most inaccessible cells in priority etc. The Reward function is presented in Equation 1 and its parameters are explained below; the subscript k is used to refer to an arbitrary cell. In Equation 2, the movement policy is presented; where p^i, p^{i+1} are the positions at steps i and $i+1$ respectively and N_k represents the 4-N neighbouring cells (hence $k = 1, \dots, 4$). The algorithm's stop criterion is presented in Equation 10.

$$R_k = m_k \Big(\left((1-g_k)\, I_k + g_k F_k \right) d_k + E\hat{Q_{env}}_k - b \Big) \tag{1}$$

$$p^{i+1} = argmax(N_k(p^i))[R_k | R_k > 0], k = 1, \dots, 4 \tag{2}$$

Inaccessibility factor *I* This factor represents the degree of inaccessibility of the cell and it depends on the number of neighbouring obstacles and covered cells. I is the product of the convolution presented in Equation 3.

$$I_k = kernel * \left(\hat{Q_{env,loc,k}} + s(Q_{cov,loc,k})\right) \quad (3)$$

- $\hat{Q_{env,loc,k}}$ and $Q_{cov,loc,k}$ are the 5×5 local sub-matrices of the sensed environment matrix $\hat{Q_{env}}$ and the covered areas matrix Q_{cov}, centered at the cell *k*. In Figure 6, the red area represents the 5×5 cell data $\hat{Q_{env,loc}}$ and $Q_{cov,loc}$ that is required for the Reward calculation of the cell marked with “x”.

- $s(x)$ is the exponential function (Equation 4) that maps the number of times a cell has been visited to a corresponding “obstacle” value; as $Q_{cov\ k}$ increases, $s(Q_{cov\ k})$ tends to 1.

$$s(Q_{cov\ k}) = 1 - e^{-a_c Q_{cov\ k}}, a_c > 0 \quad (4)$$

- *kernel* is a 5×5 distribution of weights (Equation 5) that affect the convolution product. *kernel* shifts the attention towards the 8-N cells, as the corresponding weights are higher than the 24-N ones. The matrix is divided by the sum of its elements in order to keep the convolution product bounded in $[0, 1]$.

$$kernel = \frac{1}{16}\begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & 1 & 1 & 1 & 0.5 \\ 0.5 & 1 & 0 & 1 & 0.5 \\ 0.5 & 1 & 1 & 1 & 0.5 \\ 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \end{bmatrix} \quad (5)$$

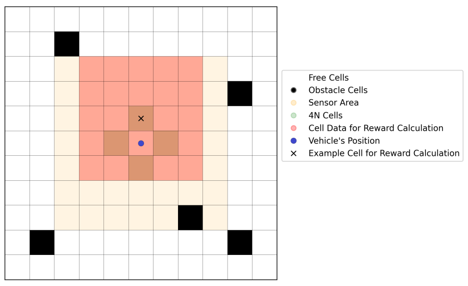

Free Cells
● Obstacle Cells
● Sensor Area
● 4N Cells
● Cell Data for Reward Calculation
● Vehicle's Position
× Example Cell for Reward Calculation

Figure 6: Reward calculation example at a random step of an arbitrary environment.

Freedom factor *F* This factor represents the degree of freedom around a cell and depends on the number of obstacles and covered cells in the cell’s neighbourhood. It is a complementary quantity to the inaccessibility factor *I* presented above. *F* is the a convolution product calculated as in Equation 6.

$$F_k = kernel * \left(1 - \left(\hat{Q_{env,loc,k}} + s(Q_{cov,loc,k})\right)\right) \quad (6)$$

Participation factor *g* The *g* factor shifts the balance from *I* to *F* depending on the number of times the cell, for which the Reward is calculated, is visited. It is calculated using an exponential function (Equation 7). The Reward value of a cell that has not been visited ($Q_{cov,k} = 0$) will entirely depend on the *I* factor ($g(Q_{cov,k}) = 0$), while the value of a cell that has been visited an arbitrary number of times will depend more on the *F* factor.

$$g(Q_{cov\ k}) = 1 - e^{-a_g Q_{cov\ k}}, a_g > 0 \quad (7)$$

Decay factor *d* The *d* factor decays the main part of the Reward value depending on the number of times the particular cell (for which the Reward is calculated) has been visited. It is calculated using the decaying exponential function presented in Equation 8. This factor motivates the AV to avoid stepping into already covered cells.

$$d(Q_{cov\ k}) = e^{-a_d Q_{cov\ k}}, a_d > 0 \quad (8)$$

Obstacle constant *E* This constant is a large negative number, which multiplies the $\hat{Q_{env\ k}}$ quantity of the cell. In the case where the cell is an obstacle ($\hat{Q_{env\ k}} = 1$), the Reward value is decreased by $|E|$, making the Reward very negative. This factor motivates the AV to avoid stepping into obstacle cells.

Bias factor *b* The bias factor *b* is a positive integer constant. It universally decreases the Reward values by $|b|$, pushing small Reward values below zero. This factor motivates the AV to ignore cells with small positive Rewards.

Momentum factor *m* The momentum factor multiplies the Reward values of the 4-N cells and motivates the vehicle to keep a straight path. Its value depends on the angle the AV makes when moving from the previous position to the next (Figure 7).

$$m_k = \begin{cases} 1, \theta = 0 \\ 0.7 < \mu < 1, \theta = \frac{\pi}{2} \\ 0.7, \theta = \pi \end{cases} \quad (9)$$

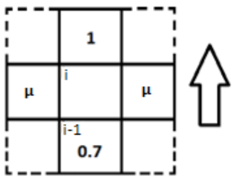


Figure 7: Weighted scheme for the momentum factor.

Checkpoint System The methodology also implements a "Checkpoint" logic: the list of covered cells with available neighbouring areas is saved and refreshed at every step. The checkpoint cells that do not have uncovered neighbouring cells are removed from the Checkpoint list. When the vehicle is trapped between covered or obstacle cells (in a 'dead end'), a shortest-path algorithm is used to return the AV to the most recent (closest) checkpoint. The A* algorithm [14] is implemented here. This technique solves the "Backtracking" issue and ensures complete coverage in every case. An example of a Checkpoint and backtracking scenario is presented in Figure 8. The Checkpoint list is also used to form the algorithm's stop criterion, which is presented in Equation 10.

$$stop\ when: \bigwedge_{k=1}^{4} (R_k \leq 0) \ \wedge \ Checkpoints = \{\} \quad (10)$$

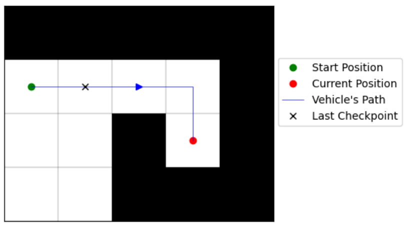


Figure 8: Example of a Checkpoint.

Parameters By tuning the model's parameters $\{\mu, a_c, a_g, a_d, E, b\}$, the algorithm can adopt different behaviours. As a default, the following set of values for the parameters is proposed:

$$\{\mu = 0.85, a_c = 2, a_g = 1.5, a_d = 4, E = -100, b = 0.02\}$$

Table 1: Heuristics Overview.

		Spatial Context	
		Local Scale	Global Scale
Behaviour Context	Exploration	*I, F, d*	Q_{env}
	Exploitation	*m, b*	Q_{cov}

Algorithm 1: Reward-Based CCPP Algorithm

```
Data: $Q_{env}$
Input : $p^0, Parameters$
Output: Path
Initialize $\hat{Q_{env}}, Q_{cov}, Checkpoints$;
while $\bigvee_{k=1}^{4}(R_k > 0) \vee Checkpoints \neq \{\}$ do
    Update $\hat{Q_{env}}$;
    Calculate $R_k, k = 1, \ldots, 4$;
    if $\bigwedge_{k=1}^{4}(R_k \leq 0)$ then
        Backtrack to last Checkpoint;
    end
    else
        $p^{i+1} = argmax(N_k(p^i))[R_k | R_k > 0], k = 1, \ldots, 4$;
        if $p^i$ has free 4N cells then
            Add $p^i$ to $Checkpoints$;
        end
    end
    Update $Checkpoints$;
end
```

4. Simulations & Results

Performance Metrics Several metrics are introduced in order to provide a comparison context for the performance of the algorithm. An idealized solution would aim to minimize all the performance metrics (and maximize the total coverage percentage). However, different metrics are often in competition, as they drive the solution in different directions - a common case for multi-objective optimization problems [15]. In general, a low repetition rate implies a higher turn rate and vice versa. It is important to note that all of the performance metrics depend on both the algorithm's characteristics and the environment.

- Total Coverage Percentage: It is defined as the total number of visited cells divided by the number of free cells in the environment. In this paper, this performance metric is not used, as the proposed methodology achieves complete coverage in any valid scenario.
- Path Length: In the case of the 4-N movement, the path length is the same as the total number of steps, as each step is of equal distance.
- Total Number of Turns: It is defined as the total number of direction changes.
- Repetition Rate: The repetition rate describes the algorithm's ability to completely cover the environment without

covering cells multiple times. It can be defined as the sum of repeated visits divided by the total number of steps.

- Turn Rate: The turn rate describes the algorithm's ability to avoid making excessive turns. It can be defined as the total number of turns divided by the total number of steps.
- Custom Cost - Energy Functions: Depending on the application and the type of the vehicle, a custom cost function can be introduced to be minimized.

Results In this section, the performance of the algorithm is showcased for a set of different environments. In Figure 9, the generated paths are presented. Table 2 summarizes the results. It is worth noting that the algorithm has no prior knowledge of the environments and the presented paths are generated in real time, as the algorithm explores the environment. Processing time indicate that the algorithm is computationally inexpensive. The simulations were coded in Python scripts and ran on Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz.

Table 2: Simulation Results.

Environment		Performance Metrics				
No.	Size	Path Length	Turns	Repetition Rate (%)	Turn Rate (%)	Runtime (ms)
1	20×20	329	126	7.3	38.3	188.51
2	20×20	328	113	1.2	34.5	188.23
3	40×40	891	341	10.1	38.3	1573.90
4	30×20	521	250	10.7	48.0	400.61

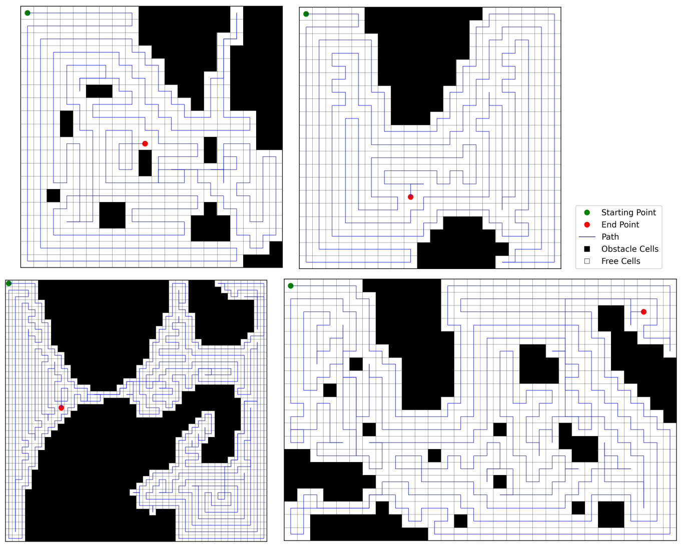


Figure 9: The algorithm's performance is showcased in a set of different environments.

5. Performance Optimization using Genetic Algorithms

The algorithm's parameters define the behaviour of the path planning algorithm. By tuning the values of the parameters, different paths may be generated. In this sense, the algorithm's performance for an arbitrary environment depends on the values of the parameters. In this section, a genetic algorithm is used to tune the algorithm's parameters in order to optimize its performance in a specific environment; the environment of Figure 10 is used as an example. However, the iterative process of the parameters' tuning require prior knowledge of the environment. Hence, the work presented in this section optimizes the algorithm's performance off-line.
Note: The «E» factor is not included as it is selected as a constant and very negative number and it does not affect the algorithm's overall performance.

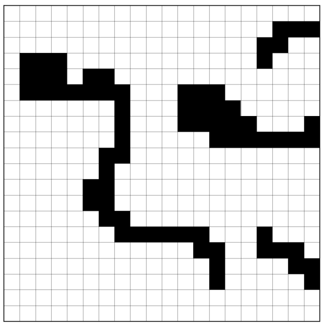


Figure 10: A 20×20 *environment is used as an example for the optimization procedure.*

Taking the performance metrics into consideration, the algorithm's optimization can be formulated as:

$$\begin{array}{ll} & \text{for the Environment Figure 10:} \\ \text{min} & \text{Path Length, Turns} \\ \text{s.t.} & \mu \in [0.7, 1], \quad a_c, a_g, a_d \in [1, 4], \quad b \in [0, 0.05] \end{array}$$

Pareto Front & Choosing a Final Solution As the optimization problem is a multi-objective one, the non-dominated solutions form the Pareto Front. Each solution in the Pareto Front represents a different trade-off between the conflicting objectives, and the choice of which solution to use depends on the decision maker's preferences. To solve this problem, a weighted cost, that is relevant to the Euclidean distance of each solution from the origin, is introduced. In order to select a final solution from the Pareto Front, a weight coefficient, which introduces a preference towards the one or the other objective, is selected, and the cost of each solution from the Front is calculated. The solution with the minimum cost is then selected. In this way, an objective-oriented preference is introduced, by shifting the balance towards the one or the other objective. For the scope of this optimization case, the weight coefficient is set equal to 0.8, thus shifting the balance towards the path length, but without introducing any loss of generality to the methodology.

$$Cost = \sqrt{w\tilde{L}^2 + (1-w)\tilde{T}^2}$$

Where: w is a weight that shifts the balance towards the one or the other objective (set equal to 0.8), $\tilde{L}$ is the path length normalized by the maximum path length in the Pareto front and $\tilde{T}$ is the total number of turns normalized by the maximum number of turns in the Pareto front:

$$\tilde{L} = \frac{L}{max(L)}, \tilde{T} = \frac{T}{max(T)}$$

Results The set-up of the Genetic algorithm used for the optimization process is presented in Table 3. To evaluate each individual from the population, a simulation is run for the set of variable values defined by the individual's genotype. The resulting path length and number of turns are used as the objective values. For the termination, the criterion of stopping after 10 generations is set. In Figure 12-Left, the optimization process through the generations can be seen, as the individuals move from top-right to bottom-left of the objective space through the generations. In Figure 12-Right, the non-dominated solutions are presented, along with the selected solution. The optimization process's cost was 150 total evaluations. The final solution's variable values are:

$$\{\mu = 0.795, b = 0.037, a_c = 3.50, a_g = 1.05, a_d = 3.25\}$$

With objective values:

$$\text{Path Length} = 384, \text{Number of Turns} = 121$$

Table 3: Genetic Algorithm Setup.

Parameter	Value
Population Size	15
Crossover	Two-Point Crossover
Mutation	Polynomial Mutation with $p = 0.1$
Elitism	2 Elites per Gen.
Total Generations	10

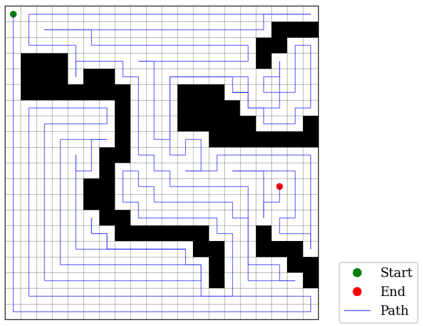
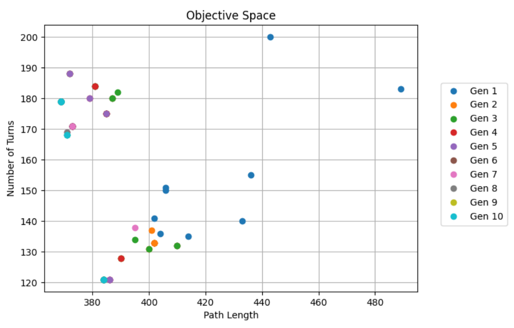


Figure 11: The generated path of the final solution.



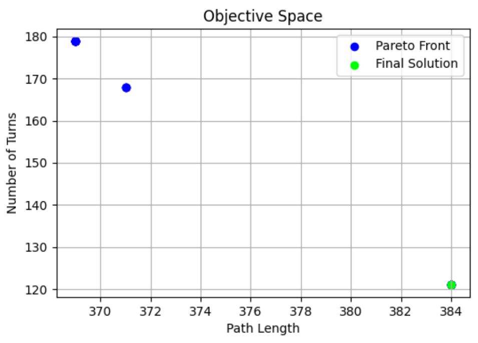


Figure 12: The Objective values of each generation's population (Left). The Pareto Front and the final solution of choice (Right).

6. Adaptive μ-Tuning CCPP Algorithm using Machine Learning

Adaptive Tuning Definition As the algorithm is initiated, it explores and covers the environment by generating a path that partially depends on the parameter μ. In this sense, an adaptive tuning approach would tune μ based on the known environment. Thus, the $\hat{Q_{env}}$ matrix can be used, as it contains information about the known environment up to this point. Using a trained convolutional neural network (CNN), the $\hat{Q_{env}}$ matrix is used as an input to get a suitable μ value as output.
The aforementioned technique imply creating a mapping mechanism between the environment and a μ value. Hence, a technique that assigns a μ value for each environment has to be utilized to gather data for the model's training. To solve this problem, the optimization procedure presented in Section 5 can be implemented. Using the presented optimization-oriented methodology, a suitable μ value can be assigned for each environment; the rest of the algorithm's parameters are considered to be fixed. As the optimization problem includes only one variable and thus the total evaluation number will be small, the Grid Search method is used instead of a Genetic Algorithm.

Data Collection The training data consists of environments and corresponding "optimal" μ values. To create a database, 292 different environments were generated. The generated environments were arbitrary selected to be 30 × 30 grids, with different obstacle configurations. The environment size is selected to be fixed in order to create a consistent optimization context and to define the input of the CNN model. For each environment, the Grid Search method is used to run 61 simulations per environment, in order to find the best μ value for each one. In this way, the "optimal" μ value is found for each environment, creating a dataset of environments and μ values. To solve the 292 optimization problems and build the dataset, approximately 2.5 hours were needed using an Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz

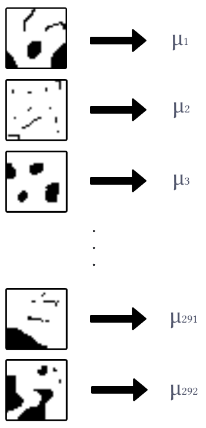


Figure 13: Each environment corresponds to an "optimal" μ value. The database includes 292 images of environments and 292 corresponding μ values.

Training The selected model is selected after a hyper-parameter optimization procedure. The selected model's characteristics and architecture is displayed below:

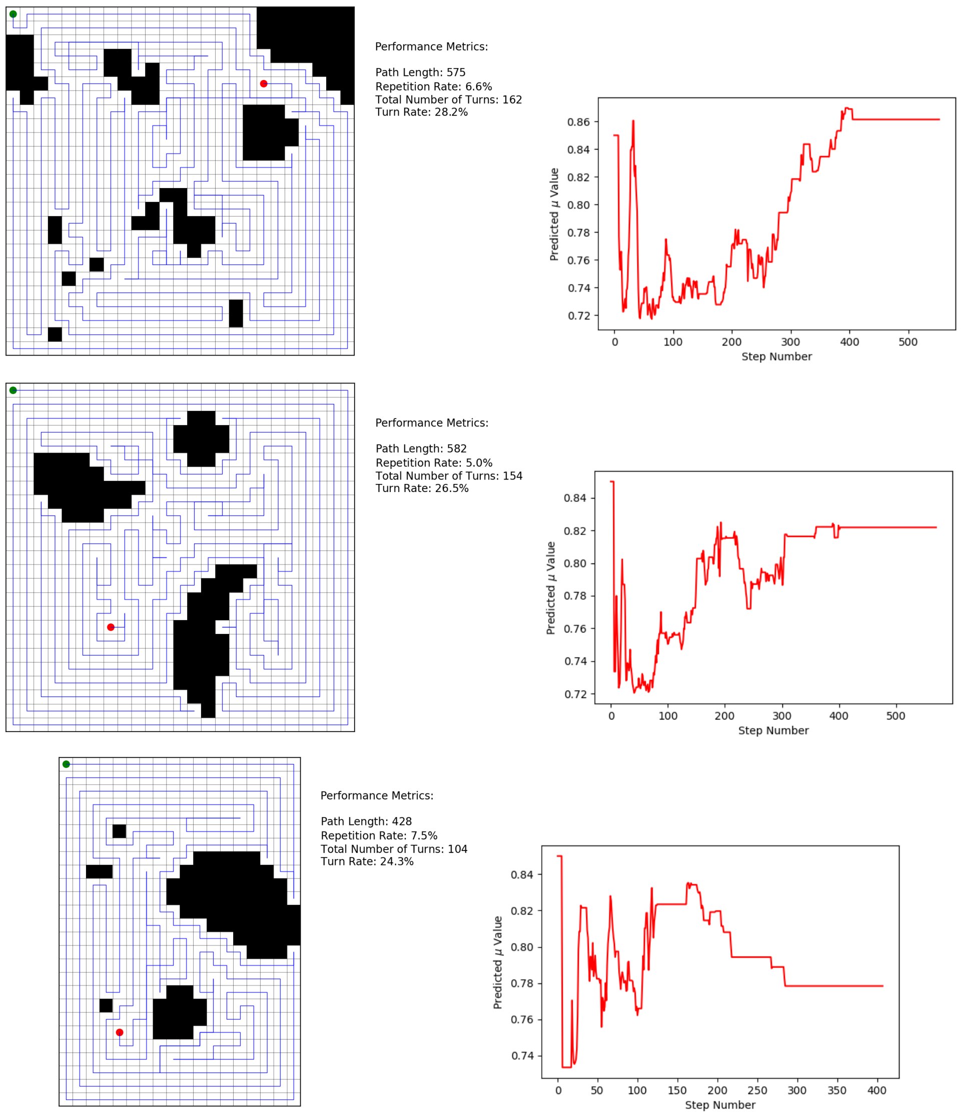
- Activation Functions: The ReLU function is used for the hidden layers and the Sigmoid function is used for the output layer.
- Convolution Kernel Size: [3,3]
- Number of Filters per Convolution Layer: $3 \cdot n$ (3 for the first Conv. layer, 6 for the second Conv. layer etc)
- Max Pooling Pool Size: [2,2]
- Training Epochs: 200
- Optimization-Training Algorithm: Adam [16]
- Loss Function: Mean Squared Error

Table 4: The model with the minimum validation error.

Validation Err.	0.07051
Convolution Layers	1
Dense Layers	3
Neurons per Layer	64
Dropout Rate	0.1
Batch Size	256

Model Implementation The selected model is implemented in the Reward-based CCPP algorithm to form the Adaptive Version. The trained model makes a prediction for the μ parameter before the Reward values' calculation, based on the known-so-far environment. The algorithm 1 is modified to use the trained model to predict suitable value for μ before calculating the Reward values R_k. To make sure that the model makes valid predictions, a fixed value for the μ is assigned ($\mu = 0.85$), until the explored area is larger or equal to a 10×10 grid area.

Performance Evaluation In Figure 14, the generated paths using the Adaptive version of the algorithm are presented for a number of different environments; the μ model's prediction at each step too.


Performance Metrics:

Path Length: 575
Repetition Rate: 6.6%
Total Number of Turns: 162
Turn Rate: 28.2%

Performance Metrics:

Path Length: 582
Repetition Rate: 5.0%
Total Number of Turns: 154
Turn Rate: 26.5%

Performance Metrics:

Path Length: 428
Repetition Rate: 7.5%
Total Number of Turns: 104
Turn Rate: 24.3%

Figure 14: Generated paths using the Adaptive version of the algorithm. The μ plots can be seen in the right.

In order to evaluate the Adaptive version's performance, the Grid Search method (for the μ parameter) is used to find the Pareto Front for each of the environments presented in Figure 14. The Grid Search interval for the μ parameter is $[0.7, 1]$ with 61 equally spaced values. The objective space is plotted and the Pareto Front is revealed. Then, the path generated by the Adaptive version is placed in the objective space and is compared to the Pareto Front. As it is presented in Figure 15, the paths generated by the Adaptive version demonstrate exceptionally favourable comparison to the non-dominated solutions obtained by the Grid Search optimization procedure.

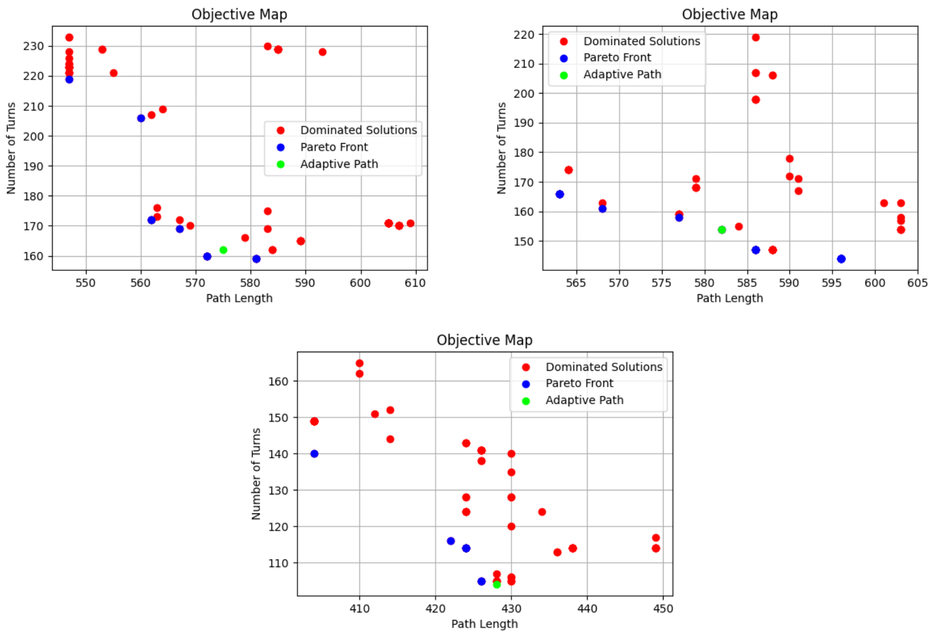


Figure 15: The paths generated by the Adaptive version shown in Figure 14 are compared to the non-dominated solutions obtained by the Grid Search algorithm.

7. Conclusions

The present paper proposed a novel solution to the problem of complete coverage path planning for an arbitrary and two-dimensional environment that is unknown beforehand. The proposed methodology contributes a new computationally inexpensive, multi-objective, Reward-based algorithm that runs on-line, and navigates an autonomous vehicle in any unknown grid-like environment to completely cover every cell. The approach builds on heuristic arguments that balance exploitation and exploration features, on one hand, and account for meaningful practical objectives and constraints. The algorithm performs satisfactorily in a range of situations (target environments), achieving complete coverage in every case, and resulting in low values of the total overall cost function. The paper also featured an off-line optimization methodology using Genetic Algorithms, and an on-line Adaptive tuning approach for the algorithm's parameters.

8. Future Work

The current paper not only presents a new solution to the complex problem of complete coverage path planning, but also establishes a foundation for further research on the featured topic. Path planning in general, and complete coverage specifically, is a problem that can be approached from different perspectives. The presented work presents a methodology that unveils numerous new tasks and problems to be solved. Potential avenues for further research include:

- Further improving and optimizing the multi-objective, Reward-based CCPP algorithm; other key-factors could be studied and be implemented in the Reward function.
- Further studying the effect of the assumed sensor system.
- An experimental context to implement the CCPP algorithm in an autonomous vehicle for a real-world testing of the algorithm's effectiveness.
- Implementing alternative optimization methods to tune the algorithm's parameters such as Bayesian optimization, Black-Box optimization [17], etc.
- Further study on learning models to increase the model's prediction accuracy for the Adaptive version.
- Further studying the multi-objective, Reward-based algorithm's decision making, using Decision Trees, Graphs or other

methods.

- Expand the multi-objective, Reward-based algorithm for various grid sizes; the motion's grid step-size could differ from the grid of the environment
- Expand the multi-objective, Reward-based algorithm for vehicles with different areas of effect; the vehicle's coverage could be expanded to not be limited to one-cell.

ACKNOWLEDGEMENTS

I would like to thank Professor Chamilothoris for the valuable guidance and his contribution to my work. I would also like to thank all of the people that supported me during my studies.

REFERENCES

[1] Faiyaz Ahmed, Jagadish Mohanta, Anupam Keshari, and Pankaj Yadav. Recent advances in unmanned aerial vehicles: A review. *Arabian Journal for Science and Engineering*, 47:1–22, 04 2022.

[2] Haque Nawaz Lashari, Husnain Mansoor Ali, and Shafiq-Ur-Rehman Massan. Applications of unmanned aerial vehicles: a review. 11 2019.

[3] Inyeong Bae and Jungpyo Hong. Survey on the developments of unmanned marine vehicles: Intelligence and cooperation. *Sensors*, 23(10), 2023.

[4] Tauã M. Cabreira, Lisane B. Brisolara, and Paulo R. Ferreira Jr. Survey on coverage path planning with unmanned aerial vehicles. *Drones*, 3(1), 2019.

[5] Krishan Kumar and Neeraj Kumar. Region coverage-aware path planning for unmanned aerial vehicles: A systematic review. *Physical Communication*, 59:102073, 2023.

[6] Jérôme Barraquand and Jean-Claude Latombe. Robot motion planning: A distributed representation approach. *The International Journal of Robotics Research*, 10(6):628–649, 1991.

[7] Mohamed Amine Yakoubi and Mohamed Tayeb Laskri. The path planning of cleaner robot for coverage region using genetic algorithms. *Journal of Innovation in Digital Ecosystems*, 3(1):37–43, 2016. Special issue on Pattern Analysis and Intelligent Systems – With revised selected papers of the PAIS conference.

[8] Sumit Gajjar, Jaydeep Bhadani, Pramit Dutta, and Naveen Rastogi. Complete coverage path planning algorithm for known 2d environment. In *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, pages 963–967, 2017.

[9] Hong Liu, Jiayao Ma, and Weibo Huang. Sensor-based complete coverage path planning in dynamic environment for cleaning robot. *CAAI Transactions on Intelligence Technology*, 3:65–72, 03 2018.

[10] Daqi Zhu, Chen Tian, Bing Sun, and Chaomin Luo. Complete coverage path planning of autonomous underwater vehicle based on gbnn algorithm. *Journal of Intelligent & Robotic Systems*, 94, 04 2019.

[11] Peng-Fei Xu, Yan-Xu Ding, and Jia-Cheng Luo. Complete coverage path planning of an unmanned surface vehicle based on a complete coverage neural network algorithm. *Journal of Marine Science and Engineering*, 9(11), 2021.

[12] M.A. Viraj J. Muthugala, S.M. Bhagya P. Samarakoon, and Mohan Rajesh Elara. Toward energy-efficient online complete coverage path planning of a ship hull maintenance robot based on glasius bio-inspired neural network. *Expert Systems with Applications*, 187:115940, 2022.

[13] Zongyuan Shen, Palash Agrawal, James P. Wilson, Ryan Harvey, and Shalabh Gupta. Cppnet: A coverage path planning network, 2021.

[14] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[15] Ching-Lai Hwang. Multiple objective decision making - methods and applications: A state-of-the-art survey. 1979.

[16] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[17] Stéphane Alarie, Charles Audet, Aïmen E. Gheribi, Michael Kokkolaras, and Sébastien Le Digabel. Two decades of blackbox optimization applications. *EURO Journal on Computational Optimization*, 9:100011, 2021.