University of Nottingham
UK | CHINA | MALAYSIA

COMP 3004 : Designing Intelligent Agents

Designing Intelligent Agents Coursework : Mobile Robot Path Planning for Search and Rescue Operations

Toluwalope Ojo

Student ID : 20042059

Abstract

This study evaluates the effectiveness of Markov Decision Process (MDP) strategies, specifically Policy Iteration and Value Iteration, for improving path planning in Search and Rescue (SAR) operations using autonomous mobile robots. The research explored variations in performance by manipulating environmental configurations and reward placements within a Gridworld simulation. It also assessed the impact of transitioning the operational model from stochastic to deterministic on algorithm performance. The main performance metrics included response times and efficiency under different simulated conditions. The findings indicate performance differences between Policy Iteration and Value Iteration, particularly in their responses to environmental changes and reward adjustments. Modifying the model from stochastic to deterministic significantly affected the policies generated by Value Iteration, presenting challenges in adaptability. Although initial attempts were made to extend the evaluation to a 3D environment using Coppeliasim, time constraints limited the thorough examination to 2D scenarios. This highlights the need for further research to assess these MDP approaches in more complex three-dimensional settings, which could provide insights into their practical application in real-world SAR scenarios.

1.1 Background

Natural disasters such as earthquakes floods, pandemics and man-made disasters from conflict and war have a huge impact on civilians worldwide, leading to severe injuries and the loss of life. In 2023 there was a total of 398 natural disasters recorded worldwide [1]. Moreover, man-made disasters over the past 20 years have made a devastating impact with over 800mn victims worldwide who have been injured or killed from these disasters [2].

These disasters have a huge impact on lives, leading to destroyed environments, disrupting civilians' access to essential resources such as food, water, safety and emergency services. Leaving towns in ruins , collapsed buildings, people trapped under rubble. To aid the victims of these disasters a wide range of initiatives have been implemented one key operation in these are Search and Rescue (SAR) operations occurring on land, sea. SAR operations are vital to locate disaster victims who may be trapped or isolated and relocating them to a safer environment. Existing methods currently use specialised teams with multiple tools such as remote sensing, satellite navigation or thermal imaging. Whilst SAR operations are essential for rescuing victims there are several limitations that impact its efficiency such as the limited resources of rescuers to aid in SAR operations since they encompass large areas with challenging terrains and weathers making unsafe or dangerous for human rescuers. Additionally, these operations are time-critical, and response needs to be prompt to increase the chances of an individual's chances of survival. Furthermore, limitations such as sensors, and limited access to certain environments are an impediment to the effectiveness of SAR operations.

The limitations of existing SAR operations have led to the integration of automated robotic systems to take on these tasks that pose a risk to humans. Around the world robotic systems are in the process of implementation in SAR , various initiatives, and organisations such as the 'Urban Search and Rescue' (USAR) and the 'Internationals Search and Rescue Advisory Group', have supports SAR operations providing robots and tools technologies [3]. An example of its effectiveness was highlighted in Hyrodnalix's Emergency lifesaving Lanyard (EMLIY) this remote-controlled robot was a lifeboat that helped rescue over 240 refugees off the coast of Greece [4].

1.2 Literature Review

Literature and studies in SAR operations have investigated the integration of autonomous agents such as robots to carry out SAR operations. In a study carried out by Niroui F et al, deep reinforcement learning techniques were adopted for the deployment of mobile rescue robots, due to the unpredictable nature of environments in SAR operations. In this study Niroui F et al combined the traditional approach of frontier-based exploration (A3C Network) with deep reinforcement learning to allow robots to explore unknown environments autonomously [5]. A virtual 2D Grid map was used to simulate a sample disaster site to optimise a mobile robots exploration strategy deciding optimal locations for the robot to explore. Experiments from this study revealed the suitability of mobile robots to explore varying environments of varying size which is key for SAR operations [6].

In their reviews, Queralta J.P. et al. and Drew D. explore various strategies for coordinating and planning multi-agent robotic systems, incorporating diverse platforms such as Unmanned Ground Vehicles, Unmanned Aerial Vehicles, Unmanned Underwater Vehicles, and Unmanned Surface Vehicles across different settings including urban, maritime, and wilderness areas. Their findings highlight how leveraging multiple autonomous robots in Search and Rescue (SAR) operations enhances reach, reliability, and reduces risks associated with system redundancies. Despite the promising integration of these robots in SAR, as of 2021, their deployment remains limited due to challenges such as communication, perception, planning, and task allocation, as well as the balance between cost and functionality [6]. This coursework builds on these insights by focusing on the performance of individual agents within multi-agent configurations, assessing their applicability in real-world scenarios where autonomous drones often conduct preliminary surveys before the deployment of ground or maritime robots.

Both studies conducted by Queralta J.P et al and Niroui F et al have analysed multiple approaches for Search and Rescue Applications broadly focusing on the preparedness and the strategy of mobile robots. Search And Rescue operations comprise of many stages of operations and can be broken down into four main phases: mitigation, preparedness, response, and recovery [6]. The response phase is the most critical phase consisting of the initial search and assessment, targeted search, localisation, and identification and finally assistance. Additionally, the level of automation was categorised into three levels from teleoperation, semi-autonomous to fully autonomous. In a study conducted by Chitikena H et al the snake robot's capability to carry out SAR was assessed at different automation levels. Chitikena H et al emphasised the importance of higher autonomy for improved intervention capabilities and addressed the ethical and design considerations for SAR operations [7].

Previous research has compared traditional methods like the A* search combined with Braitenberg collision avoidance algorithms to reinforcement learning techniques, finding the latter more adaptable for autonomous robots in unknown environments. This project focuses specifically on optimizing the navigation and path planning phases of Search and Rescue (SAR) operations using autonomous mobile robots. Studies, including those by Lee M. et al., have demonstrated the effectiveness of learning agents such as the Deep Q-Network (DQN) and

Double Deep Q-Network (DDQN) in indoor navigation without collisions, with the DDQN outperforming the DQN by 5.06% in simulations aimed at reaching target locations [8].

Furthermore, Chen Y. et al. developed a real-time path planning algorithm utilizing a Markov Decision Process, which includes utility and policy update phases based on Bellman's Equation [9]. This algorithm proved to be particularly effective in dynamic environments, surpassing traditional methods like the A* algorithm in speed and adaptability to moving obstacles. Given these findings, this coursework will employ reinforcement learning, specifically the Markov Decision Process, for robot navigation. A notable limitation of prior studies is their restriction to 2D simulations, which overlook factors like rough terrain or obstacle size that affect real-world SAR operations. To address this, this coursework will incorporate 3D simulations to better validate the results in realistic scenarios.

2.0 Aims & Objectives

This project aims to assess the efficacy of path planning algorithms in a search and rescue context, focusing on their performance within both 2D and 3D simulated environments. Specifically, the study utilizes the environment Gridworld for 2D path planning and Coppeliasim for 3D scenarios. The primary objective is to compare the effectiveness of the Markov Decision Process algorithms, namely Policy Iteration and Value Iteration, in optimizing rescue paths. These algorithms will be evaluated based on criteria such as time efficiency, reward value, and convergence rates. The project will be conducted by setting up different difficulty levels within these environments to simulate a variety of search and rescue scenarios. This approach stems from the hypothesis that if an algorithm can efficiently navigate complex paths with multiple obstacles, it would naturally excel in simpler, more straightforward environments. This could potentially negate the need for developing multiple specialized algorithms for varying complexities.
The performance of the algorithms will be tested by comparing how each handles the transition between simple and complex environments within the simulations. By the end of the project, the aim is to determine which algorithm not only performs optimally in varied scenarios but also learns the most effective paths in terms of time and computational resources, thereby enhancing the overall efficiency of search and rescue operations. The insights gained will inform the development of more robust path planning strategies in real-world rescue missions, with a clear understanding of each algorithm's capabilities and limitations in dynamic environments.

3.0 Methodology & Implementation

In this section the environment , autonomous agents and methodology used to investigate this topic are explained. The chosen environment , agent and questions were selected to simulate the localisation and path planning of a ground UGV mobile robot, searching for a human in this experimentation the highest value reward is achieved when this mobile robot reaches the goal location.

3.1 Environment :

```
World = [
        1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
        1 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
        1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
        1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
        1 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 0 0 0 1
        1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 0 0 0 1
        1 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 1
        1 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1
        1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1
        1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1
        1 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1
        1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
        1 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
        1 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
        1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1];
```

Figure 1 : Grid Environment

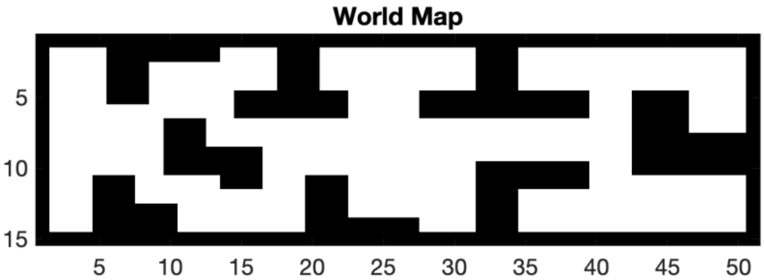

Figure 2 : Generated World Map Environment

For experimentation in this project a grid environment was generated as seen in Figure 1 and 2 . This project was written in C using MATLAB IDE, because of its ability to connect with Coppeliasim via remote API. Two environments were used the first Gridworld is a 2D simulated environment that is comprised of free spaces denoted by 0 and obstacles denoted by 1. For each turn the robot could move in 8 directions or remain stationary. The reward function was given a high value of 100 representing the goal state. The obstacles in this environment represent the real-life obstacles a mobile robot encounters during navigation and path planning. Two maps of varying complexity level were chosen to evaluate the performance of both MDP algorithms.

For final simulations Coppeliasim environment was chosen, this environment enables realistic modelling of search and rescue scenarios in the 3-dimensional field. In addition, with the ability to select from a wide range of mobile robots with assorted shapes wheels and characteristics. The use of Coppeliasim facilitates the understanding of how these mobile robot characteristics impact a robot's ability to navigate in this environment. The use of Coppeliasim mitigated many of the limitations of the 2d simulations. Synchronous mode connection for the remote API service between the MATLAB code and the Coppeliasim environment , as shown in Figure 3.

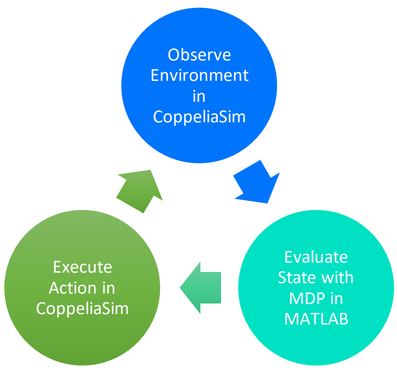

Figure 3: Remote API MATLAB Synchronous mode Connection

3.2 Autonomous Agents:

In the Gridworld 2D simulations , the start point of the algorithm represents the robot and the target goal. The Markov decision process algorithms value iteration and policy iteration. In these simulations the single point represents the robot states orientations can be defined, multiple robot characteristics remain undefined.

In Coppeliasim environment the mobile Pioneer 3dx Robot is used , this is a small lightweight two-wheel two motor differential drive robot. This robot was selected over more complex robots such as the snake robot due to its simplicity, restricted orientations and degrees of freedom that can easily be defined seen in Figure 4 and 5.

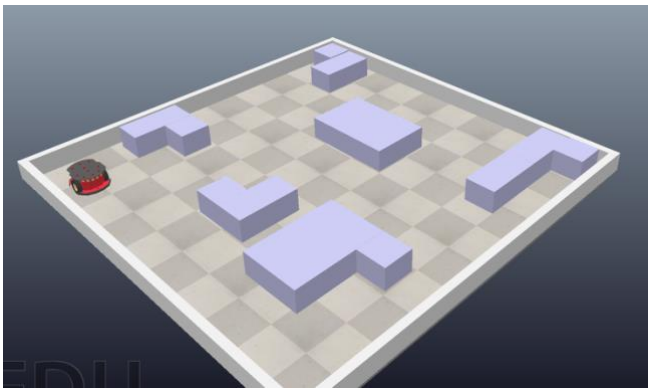

Figure 4 : Coppeliasim sample environment.

Figure 5 :Pioneer 3DX mobile robot

3.3 Markov Decision process

As exhibited in literature , there are many benefits for the use of Markov Decisions process for path planning however its ability to model decisions under uncertainty , handle stochastic environments makes this process ideal for path planning in SAR applications. The Markov decision process (MDP) is a mathematical framework used to model sequential decision-making problems for stochastic environments. The MDP consists of a set of states , actions, a transition model, and a reward. In this experimentation the MDP is used to represent the algorithm deciding the actions, states, and reward of a mobile robot.

State, S : Robots position in space, the environment was discretised into a grid where each cell represents a state.

Action, A(s) : Defines actions of the mobile robot, choices to move forward, turn left, right from a state in addition to obstacle avoidance until finding the goal state.

Rewards R(s, a, s'): the points received when transitioning to a state , this value is dependent on the state for example very low values of rewards for obstacle states ensure that the mobile robot avoids obstacles. In contrast to the target location in the experiment representing a victim of a natural disaster, the reward value was very high for the goal state to encourage the model to choose optimal path that leads to the goal.

Policy (π): This is the set of instruction for the robots defining what actions should be taken in each state. This maps states to optimal actions that maximise the expected rewards.

Model Transition T(s, a, s'): This represented the probability to go from one state 's' to the next s' ,

The overall goal of **Markov Decision Process** is to find the optimal policy and maximise the long-term reward, in contrast to traditional methods that rely on memory, this algorithm ins memoryless and the probability of the transitions to future states depend only on the current state not the history.

Two of the methods used to solve **Markov Decision Processes** that will be compared in this project are Policy Iteration and Value Iteration :

Value Iteration:

Bellman's Equation : $\boldsymbol{U(s) = R(s) + \gamma max_a \sum_{s'} T(s, a, s')U(s')}$

In value iteration we compute the optimal state value function by iteratively updating the estimated value functions based on the value functions of the neighbouring states. This process starts with random value functions U(s) and once the optimal value function is obtained then the optimal policy can be derived. Pseudo code can be seen in Figure 6 .

Algorithm 2: Value Iteration Algorithm

```
Data: θ: a small number
Result: π: a deterministic policy s.t. π ≈ π*
Function ValueIteration is
    /* Initialization                                   */
    Initialize V(s) arbitrarily, except V(terminal);
    V(terminal) ← 0;
    /* Loop until convergence                           */
    Δ ← 0;
    while Δ < θ do
        for each s ∈ S do
            v ← V(s);
            V(s) ← max_a Σ_{s',r'} p(s', r|s, a)[r + γV(s')];
            Δ ← max(Δ, |v − V(s)|);
        end
    end
    /* Return optimal policy                            */
    return π s.t. π(s) = arg max_a Σ_{s',r'} p(s', r|s, a)[r + γV(s')];
end
```

Figure 6 : Value Iteration Pseudo-Code [10]

Policy Iteration:

Equation : $\boldsymbol{U_t(s) = R(s) + \gamma \sum_{s'} T(s, \pi_t(s), s')U_t(s')}$

Policy iteration is a cycle comprising of the two phases : policy improvement and policy evaluation. In policy improvement, an arbitrary policy is chosen based on the policy the value function is calculated and the policy is updated by searching over all the actions using the Bellman Equation. Then the policy is improved by changing the policy of that state. With a finite number of policies this process is finite and eventually converges. Pseudo code is shown in Figure 7.

Algorithm 1: Policy Iteration Algorithm

Data: θ: a small number, η: another small number and $\eta > \theta$
Result: V: a value function s.t. $V \approx v_*$, π: a deterministic policy s.t. $\pi \approx \pi_*$

Function *PolicyIteration* **is**
  /* Initialization */
  Initialize $V(s)$ arbitrarily;
  Randomly initialize policy $\pi(s)$;
  /* Policy Evaluation */
  $\Delta \leftarrow \eta$;
  while $\Delta > \theta$ **do**
    for *each* $s \in S$ **do**
      $v \leftarrow V(s)$;
      $V(s) \leftarrow \sum_{s',r'} p(s', r|s, \pi(s))[r + \gamma V(s')]$;
      $\Delta \leftarrow \max(\Delta, |v - V(s)|)$;
    end
  end
  /* Policy Improvement */
  policy-stable $\leftarrow$ *true*;
  for *each* $s \in S$ **do**
    old-action $\leftarrow \pi(s)$;
    $\pi(s) \leftarrow arg\max_a \sum_{s',r'} p(s', r|s, a)[r + \gamma V(s')]$;
    if *old-action* != $\pi(s)$ **then**
      policy-stable $\leftarrow$ *false*;
    end
  end
  if *policy − stable* **then**
    return $V \approx v_*$ and $\pi \approx \pi_*$;
  else
    go to Policy Evaluation;
  end
end

Figure 7 : Policy Iteration Pseudo Code [10]

Difference between Policy and Value Iteration :

Value Iteration	**Policy Iteration**
Start with initial value functions	**Start with an initial policy**
Update value functions at each iteration	**Update the policy at each iteration**

4 Experimentation

4.1 Questions

To achieve the aims and objectives of this coursework the following questions were the focus for experimentation.

1. How do the performance of the different MDP approaches Policy and Value Iteration compare
2. What is the impact on the performance of both MDP approaches when the grid/ environment changes and the position of the reward changes.
3. The impact of changing the reward location.
4. What is the impact on the Policy generated in value iteration when the nature of the model is changed from stochastic to deterministic.
5. How does this simulation in a 2d environment compare to a 3D environment in Coppeliasim.

4.2 Experiment breakdown

In this project the following experiments were carried out to answer these questions:

E1 : Map 1 MDP Performance Evaluation

To evaluate the performance of value iteration and policy iteration for path planning one identical grid as shown in Figure 8 was used with identical obstacles and reward location. To assess the performance the total computation time, average time per value update and the reward per iteration was compared for policy and value iteration.

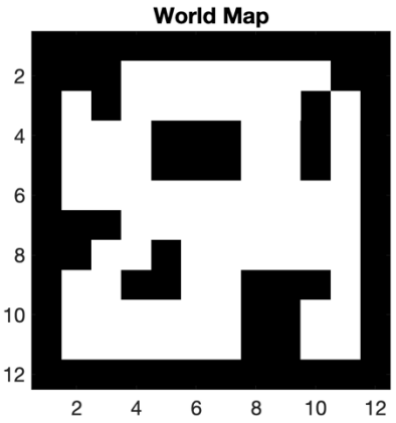

Figure 8 : Map 1

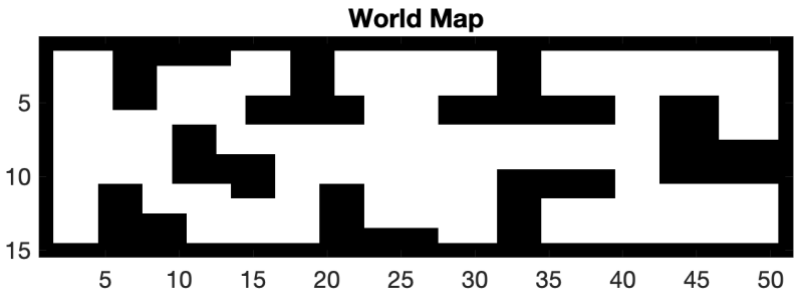

Figure 9 : Map 2

E2 : Map 1 and Map 2 MDP Performance

To evaluate the impact of the grid or environment complexity, the number of obstacles were increased, and the location of the reward/ goal location were varied. Map 1 and Map 2 were shown in figures below. By varying the shape of the environment from symmetrical in map 1 to asymmetrical and rectangular map 2 , the asymmetrical environment forced the agent to adapt to the unexpected creating a partially observable Markov Decision Process.
map 1 : 6 obstacles, reward location 4,11; map 2 : 8 obstacles, reward location 12,49

E3: Reward Location

The impact in the variation of reward location across Map 2 was investigated through comparative analysis of the policies for both locations.

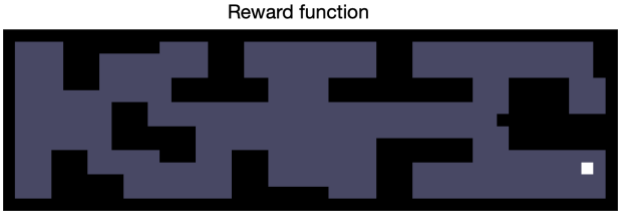

Figure 10 : MAP 2 Reward Location 1 - (12,49

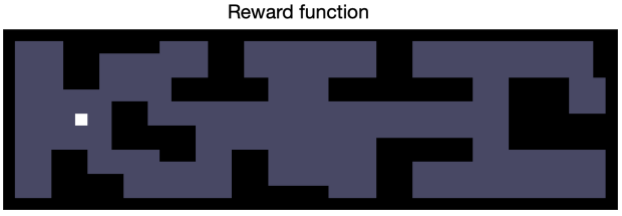

Figure 11 : MAP 2 Reward Location 2 – (6,8)

E4 : Stochastic vs Deterministic

For value iteration experimentation the final policy between both stochastic and deterministic environments were compared to assess which environments are more suitable for dynamic path planning in SAR . In deterministic value iteration the outcome of each action is certain however in stochastic / probabilistic value iteration the outcomes of the action are not certain, this reflects real world uncertainties as shown in Figure 12. These are more complex and realistic and reflect uncertainties in robots' actuators or sensors or the complexity of a SAR environment.

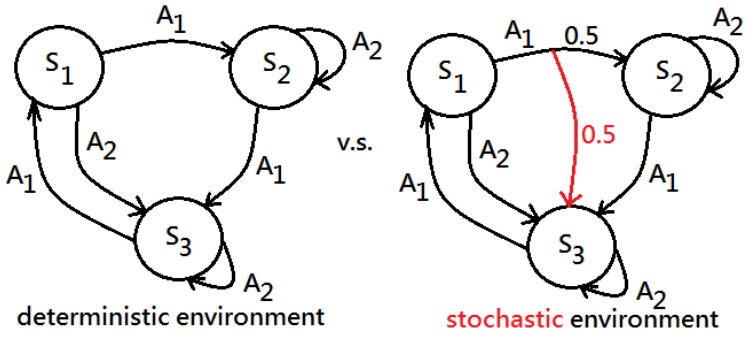

Figure 12: Stochastic and deterministic environment schematic [11]

E4 : Coppeliasim 3D Environment

To prove the applicability of this in the 3D world value iteration was planned to be carried out in MATLAB using the Coppeliasim via Remote API. Limitations in this experiment impeded any results from this experiment hence experimentation did not proceed beyond 2D experimentation and results were not obtained.

5 Results & Discussions

E1 Results :

Figure 13: MAP 1 Heatmap Policy Iteration

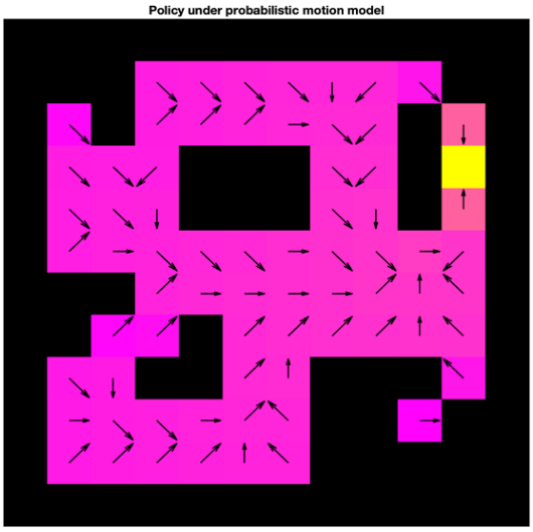

Figure 14: MAP 1 Heatmap Value Iteration

For MAP 1 comparing the performance of Policy and Value Iteration :

These results exhibit the heatmap and policies generated from policy and value iteration. The colour gradient on the heat map denotes the desirability of a state this is defined in terms of its closeness to the goal state at (4,11). For Figure 13 at convergence the policy in policy iterations has shifted away from the left region of the grid however the higher more desirable states are dispersed on the grid indicating that this policy has not converged to an optimal path. Additionally, policies in some cases are directed towards obstacles rather than away. In Figure 14 , for value iteration the optimal path is very clear, a limitation in this comparison is the defined actions for value iteration which were greater than the defined policies of policy iterations.

Total computation time

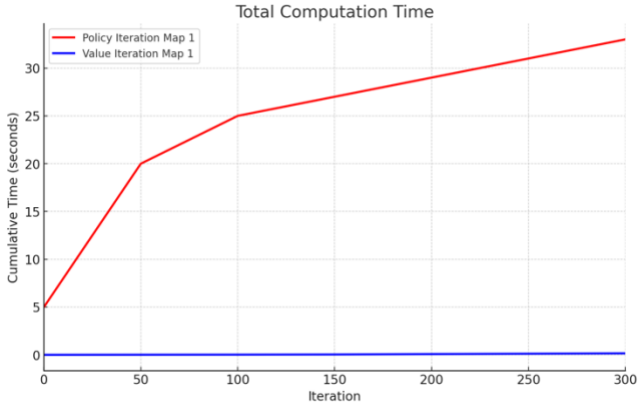

Figure 15: Total Computation time Map 1

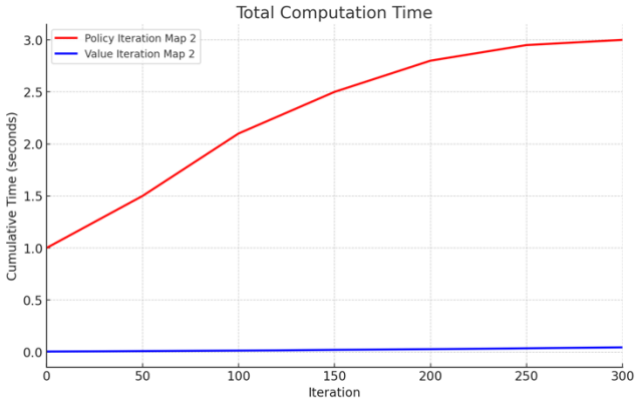

Figure 16: Total Computation time Map 2

For Map 1 (Figure 15) and Map 2 (Figure 16) it is evident from the results that value iteration values are significantly smaller within the range 0.01 to 0.2 seconds , while policy iteration values were significantly larger from 5 to >30 s. When comparing total computation time between Map 1 and Map 2 for policy Value iteration, the gradient of the two maps computation follows the expected trend. Map 2 is more complex and exhibits a steep gradient as the total computation time is greater this is expected because the time to converge is longer with more obstacles and more states. This difference is also seen in policy iteration computation time for map 1 and map 2 however there is a greater discrepancy between the two maps this can allude to the fact that value iteration is more ideal for quick and effective path planning and can mitigate the impact of the added complexity of the environment (from Map 1 to Map 2).

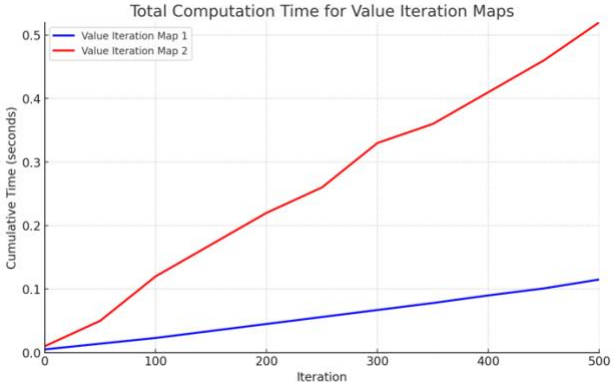

Figure 17: Total Computation time for Value iteration

Policy Iteration and Value iteration time variance box plot

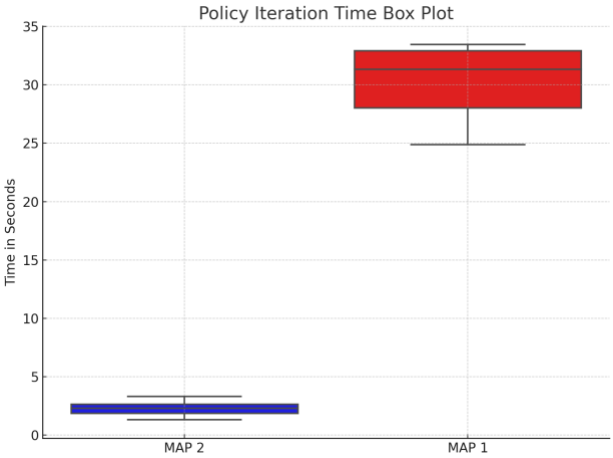

Figure 18: Policy iteration Time Box plot

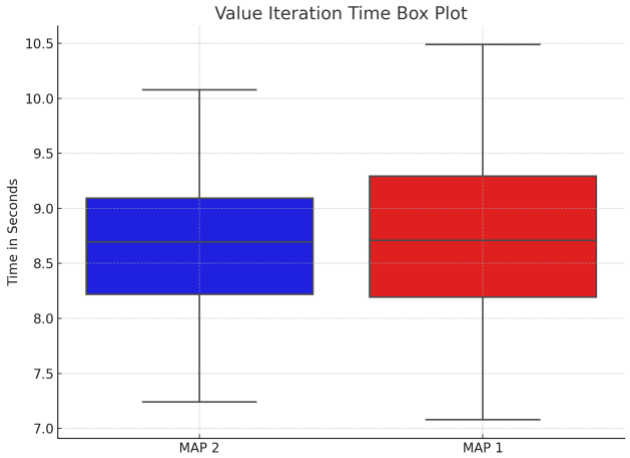

Figure 19: Value iteration Time Box plot

To assess the accuracy of each method the same conditions were repeated ten times and the time for completion was derived for each policy and value iteration. Box plots were generated for policy and value iteration for maps 1 and 2 as shown in Figure 18 and 19. By comparing the two box plots for Policy Iteration and Value Iteration, it could be deduced that value Iteration demonstrates more consistent performance stability. Despite the increased complexity from Map 1 to Map 2, the variance in execution times for Value Iteration remains relatively unchanged. This suggests that Value Iteration is less sensitive to changes in map complexity, indicating a potentially more robust approach under varying environments.

E4 : Policy visualisation comparison

MAP 1 Policy Comparison

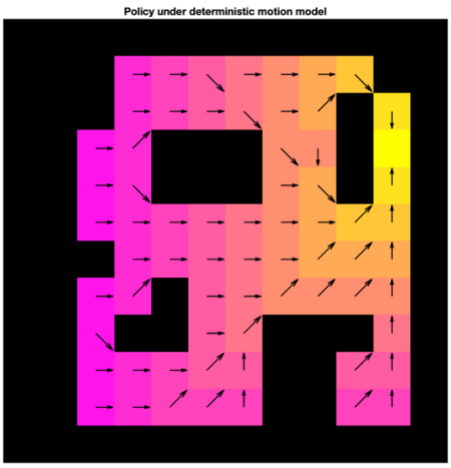

Figure 20: Map 1 Policy under deterministic model

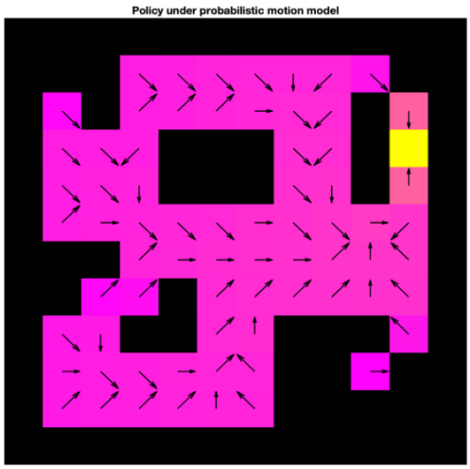

Figure 21: Map 1 Policy under stochastic/probabilistic model

Map 2 Policy Comparison

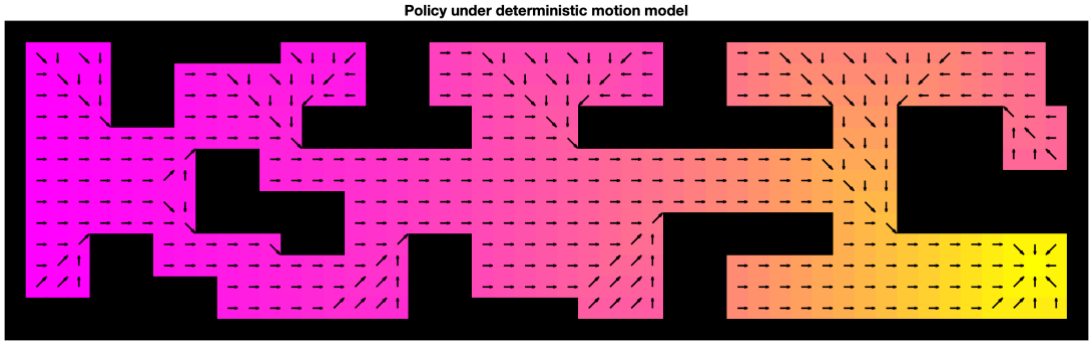

Figure 22: Map 2 Policy under deterministic model

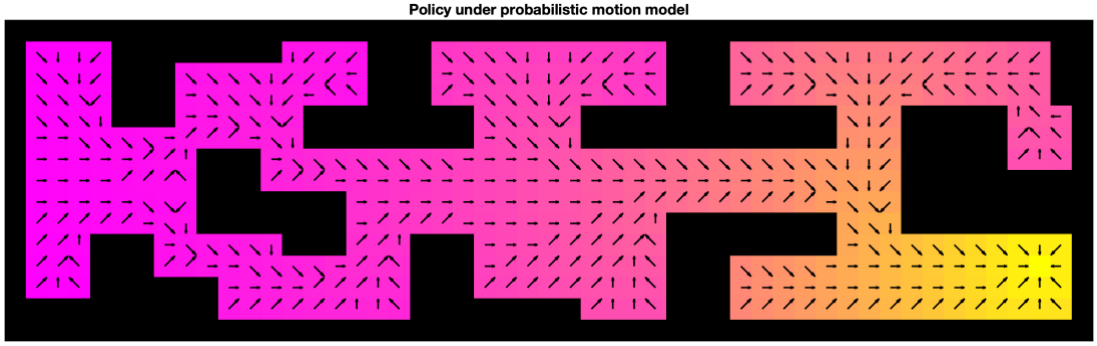

Figure 23: Map 2 Policy under stochastic/probabilistic model

In the deterministic model the robot operates under the assumption that every movement will result in the expected outcome, each state has a specific chosen action the policy function directly maps a state s to an action a, there is no randomness in the action. In Figure 20 and 22 for Map 1 and 2 , the deterministic motion shows arrows following the shortest possible path without variation. The path to the goal is more direct often skirting along obstacles. However, in a probabilistic model , the model accounts for errors in the robots' actions. Hence the results for the probabilistic model (Figure 21 and 23) exhibit paths that less direct with more varied arrow directions additionally the paths chosen are wider avoiding narrow passages. A noticeable difference can be seen in the colour gradient between the simple map 1 between stochastic and deterministic compared to map 2. The colour gradient represents darker colours (pink/purple) are lower values further from the goal and lighter colours like yellow represent higher values as they lie closer to the goal. This implies that the impact of introducing a probabilistic motion model has a greater impact on smaller grid map 1 than map 2 .

Reward locations

When the reward locations were varied for the map 2, this had little impact on the time however had a significant impact on the derived policy as seen in the policy visualisation. When the reward was moved from behind an obscure obstacle policy iteration developed policies with an improved path . This indicates again that value iteration is more suited to navigation and finding hidden objects or trapped victims in SAR operations.

Challenges:

During this project, several technological challenges arose. Initially, the plan was to manually generate transition probabilities; however, it became necessary to automate this process. A function was developed to generate transition probabilities, considering obstacles which presented difficulties. The function had to be adjusted so that the transition matrix did not allow moves into obstacles or outside the designated grid area. Another challenge was the intention to compare Policy Iteration and Value Iteration with Q-Learning, but the extensive time required for simulations limited the feasibility of this comparison. Additionally, difficulties in establishing a remote API connection between MATLAB and CoppeliaSim restricted testing, resulting in only Value Iteration being implemented on Gridworld.

Conclusion

In conclusion results from this experimentation showed improved performance with the use of value iteration, showing more optimal paths to the desired goal. Despite challenges faced throughout simulations results showed a clear mitigation of the impact of added complexity of the maps with value iteration, this was exhibited in time variance, policies and total computation time.

Future Work:

The limitations experienced with CoppeliaSim meant that simulations were restricted to a 2D plane. There was an unintentional bias in the comparison between Policy and Value Iteration due to differences in the range of orientation and possible actions, which could have contributed to the apparently superior performance of Value Iteration. In the configurations used, Value Iteration was defined with eight possible actions, whereas Policy Iteration was limited to just four, suggesting a need for re-evaluation of these approaches with equivalent action possibilities in future studies.

References

[1] Aon Corporation, 'Natural Catastrophes Caused USD 65 Billion Economic Loss in Asia Pacific in 2023, Aon reports', *Aon.com*. [Online]. Available: https://www.aon.com/apac/in-the-press/asia-newsroom/2024/natural-catastrophes-caused-usd-65-billion-economic-loss-in-asia-pacific-in-2023. [Accessed: 17-May-2024].

[2] P. R. Hannah Ritchie, 'Natural Disasters', *Ourworldindata.org*, 12.2022. [Online]. Available: https://ourworldindata.org/natural-disasters. [Accessed: 17-May-2024].

[3] Association for Computing Machinery, *Search and rescue robotics: From theory to practice*. BoD – Books on Demand, 2017.

[4] S. Gossett, '12 examples of rescue robots to know', *Built In*, 25-Jun-2019. [Online]. Available: https://builtin.com/robotics/rescue-robots. [Accessed: 17-May-2024].

[5] F. Niroui, K. Zhang, Z. Kashino, and G. Nejat, 'Deep reinforcement learning robot for search and rescue applications: Exploration in unknown cluttered environments', *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 610–617, 2019.

[6] J. P. Queralta *et al.*, 'Collaborative multi-robot search and rescue: Planning, coordination, perception, and active vision', *IEEE Access*, vol. 8, pp. 191617–191643, 2020.

[7] M.-F. R. Lee and S. H. Yusuf, 'Mobile robot navigation using deep reinforcement learning', *Processes (Basel)*, vol. 10, no. 12, p. 2748, 2022.

[8] H. Chitikena, F. Sanfilippo, and S. Ma, 'Robotics in Search and Rescue (SAR) operations: An ethical and design perspective framework for response phase', *Appl. Sci. (Basel)*, vol. 13, no. 3, p. 1800, 2023.

[9] Y.-J. Chen, B.-G. Jhong, and M.-Y. Chen, 'A real-time path planning algorithm based on the Markov decision process in a dynamic environment for wheeled mobile robots', *Actuators*, vol. 12, no. 4, p. 166, 2023.

[10] M. Aibin, 'Value Iteration vs. Policy Iteration in Reinforcement Learning', *Baeldung.com*, 29-Mar-2024. [Online]. Available: https://www.baeldung.com/cs/ml-value-iteration-vs-policy-iteration. [Accessed: 17-May-2024].

[11] 'Markov decision process by intuition', *Github.io*. [Online]. Available: https://mjtsai1974.github.io/DevBlog/2017/12/01/mdp-markov-decision-process/. [Accessed: 17-May-2024].