A Deep Reinforcement Learning Framework for UAV Navigation in Indoor Environments

Ory Walker
Queensland University of Technology
2 George Street
Brisbane City, QLD, 4000
ory.walker@hdr.qut.edu.au

Fernando Vanegas
Queensland University of Technology
2 George Street
Brisbane City, QLD, 4000
+61 (0) 7 3138 1772
fernando.vanegasalvarez@hdr.qut.edu.au

Felipe Gonzalez
Queensland University of Technology
2 George Street
Brisbane City, QLD, 4000
+61 (0) 7 3138 1363
felipe.gonzalez@qut.edu.au

Sven Koenig
University of Southern California
300 Henry Salvatori Computer Science Center
941 Bloom Walk
Los Angeles, CA 90089-0781 4000
213-740-7285
skoenig@usc.edu

***Abstract*—This paper presents a framework for UAV navigation in indoor environments using a deep reinforcement learning based approach. The implementation models the problem as two seperate problems, a Markov Decision Process (MDP), and a Partially Observable Markov Decision Processes (POMDP), separating the search problem into high-level planning and low-level action under uncertainty. We apply deep learning techniques to this layered problem to produce policies for the framework that allow a UAV to plan, act, and react. The approach is simulated and visualised using Gazebo and is evaluated using policies trained using deep-learning. Using recent deep-learning techniques as the basis of the framework, our results indicate that it is capable of providin smooth navigation for a simulated UAV agent exploring an indoor environment with uncertainty in its position. Once extended to real-world operation, this framework could enable UAVs to be applied in an increasing number of applications, from underground mining and oil refinery surveys and inspections, to search and rescue missions and biosecurity surveys.**

TABLE OF CONTENTS

1. INTRODUCTION

An increasing number of applications for Unmanned Aerial Vehicles (UAV) within society require the capacity for autonomous searching of a known or unknown environment in the presence of both hazards and sensing limitations. Such applications include the broad field of search and action tasks, from search and rescue [24], environmental sampling and data collection [14], pursuit of targets in complex environments [3], even underground mining and surveying of confined spaces [8].

A common requirement for such applications is the ability to search continuous and/or large environments. Representing large environments discretely is one solution to the problem of large environments [15], however one must then consider how a UAV would search each discrete cell.

There are multiple works involving searching or navigating an unknown environment using UAVs. Most solutions however do not allow for operation of the UAV under uncertainty [13] or operation in continuous action spaces, using a discrete list of actions to control the operation of the UAV [25].

Many works regarding planning of robotic agents apply the Partially Observable Markov Decision Process (POMDP) or Markov Decision Process (MDP) framework when formulating a planning problem [6] [2] [1]. The main reason being that POMDPs are especially useful for modelling uncertainty within a system. In the past POMDPs were most effectively solved by classical algorithms and solvers [17] [22] [12]. However, these approaches had a number of limitations, including discrete actions spaces [17] or difficulty modelling continuous actions spaces [17] [22], pauses in operation to calculate the next optimal trajectory [17] [22] in the case of online planners, or long pre-planning times for each new environment.

Such limitations could be detrimental to the completion of any search task with time constraints. Fortunately, recent developments in the field of deep learning have shown promise when applied to POMDP problems [9] [11] [7]. While deep reinforcement learning of a POMDP model can be quite time consuming initially, correct formulation of a model and its training allows for application to a variety of environments, meaning continuous operation and little to no pre-training of the model prior to operation. Solutions are also available for continuous state and action spaces using modern deep reinforcement learning techniques [18].

Deep learning techniques have recently seen applications within the field of UAV operation [20] [5], however the application of such techniques to UAV decision-making remains largely unrepresented, with most efforts being directed

978-1-5386-6854-2/19/$31.00

toward low-level control [16] and image processing based obstacle avoidance [23] .

This paper maintains a focus on the problem space of searching inside a building using UAVs. We present the UAV indoor search problem as two separate problems, a global planning problem, and a local planning problem. The UAV must be capable of searching a local environment, while also being directed between local environments within a global search space. In order to illustrate the concept, a map of the search environment is provided. The UAV is assumed to have an estimation of its velocity, but uncertainty is present in the UAVs knowledge of its location with respect to the environment.

We develop a framework to develop models and test the application of modern deep reinforcement learning approaches to the two concurrent problems of global and local planning.The Deep-RL algorithm, Trust Region Policy Optimization [21] is used for validation and illustration purposes.

The main contributions of this paper are:

1. A scalable formulation of the UAV search problem within large environments as two separate problems, an MDP problem and a POMDP problem. The global MDP using a discretized map of the environment to provide macro actions for the system, while the local POMDP problem utilizes a continuous action and state space to enable a UAV to smoothly search a local environment in 3 degrees of freedom while avoiding any obstacles present despite uncertainty in its measurements. The framework implements the MAVROS-PX4-Gazebo software in the loop functionality to provide low level control of a simulated 3DR Iris UAV within gazebo.

2. A framework for the testing of deep reinforcement learning techniques when applied to the two concurrent POMDP problems of global and local planning. The framework uses a modular architecture that permits variable functionality and testing of the system via software in the loop within the Gazebo simulator.

2. BACKGROUND

POMDP

The POMDP framework is suitable for modelling sequential decision making for robotic agents operating in the real world, as robotic agents rarely if ever have access to measurements and sensing free from uncertainty.

Formally, the POMDP framework can be defined by an 8-tuple (S, A, T, O, Ω, R, b0, γ). Where:

- S is a set of states,
- A is a set of potential actions,
- T is the transition function, defining the probability of transition between states,
- O is a set of observations, Ω is the observation function, defining the probability of observing *o* from state *s* after taking action *a*,
- R is the reward function,
- b0 is the initial belief,
- and γ is the discount factor.

An agent (UAV) in a POMDP formulation does not have knowledge of the true state of the environment. The agent instead maintains a continuous belief *b(s)* over the potential states. It updates this belief after taking an action *a* and making an observation *o* . Consider this belief a distribution of all possible states for the agent given the transition and observation functions T and Ω, with less certain functions (modelled for less certain problems) resulting in a larger belief space, and more uncertainty in the true state of the agent and environment.

Much like the foundational MDP, the objective of a POMDP solver is to identify a sequence of actions that maximize the total expected discounted return of the model. Rather than calculating for a known true state *s* however, the POMDP solver uses an initial belief of potential states b0.

The solution a POMDP solver provides is a policy π that maps belief space to action space and identifies which action will maximize the expected discounted reward given a belief b(s). The optimal policy is then represented by π* which yields the maximum reward value for each belief state, which is denoted by V*. V* (b) is therefore the value function that accounts for the maximum reward value for any given belief *b*.

Deep Reinforcement Learning and TRPO

Solving the POMDP problems of global and local planning involves optimizing a policy with respect to a future discounted reward. The primary goal of deep reinforcement learning within our framework is therefore to learn optimal policies for both problem formulations. Deep reinforcement learning leverages the properties of deep neural networks and reinforcement learning algorithms to produce near-optimal policies for complex problems. For illustrating the framework we opted to use the Deep Reinforcement Learning Algorithm, Trust Region Policy Optimization to solve the global and local planning problems. However, the framework is designed in a modular fashion which permits the use of alternative algorithms to solve either of the local or global planning problems, so that other algorithms can also be tested for performance. A key benefit of applying Deep Reinforcement Learning to the local control problem is the reduced control overhead required when implementing a learned policy and the near-instantaneous response to new information available to the UAV at any given time, without the need for recalculation. This assumes however that an adequate policy has been trained, but with modern Deep-RL techniques improving rapidly, the problem space a useable policy can be produced for like-wise improves.

More classical solutions are more applicable to the global planning problem, where the UAV must visit each node within a graph map and satisfy a local search condition. However, for this paper an MDP model was developed and a policy was trained for the small global planning problem using the same TRPO algorithm that was used for the local problem.

Trust Region Policy Optimization:

Trust Region Policy Optimisation (TRPO) [21] is an on-policy policy gradient method that allows precise control of the policy imporvement during optimization. At each iteration k, the algorithm attempts to solve the constrained optimization problem by optimizing the stochastic policy π_0. The algorithm is able to accomodate both discrete and continuous action spaces by changing the representation of the stochastic policy π_0. A categorical distribution is used when the action space is discrete, and a Guassian distribution is used when the action space is continuous. The algorithm

has been shown to learn complex policies for continuous tasks such as swimming, hopping and walking and further improvements have shown it can be adapted to learn policies for multiple agents in continuous tasks [9]. TRPO is well suited to the local planning problem, being a continuous task, and is also capable of handling the discrete global planning MDP.

3. FRAMEWORK ARCHITECTURE

A modular framework was created that can be broken into four major modules, shown in figure 1. The four modules run in parallel and interface with each other using the Robotic Operating System (ROS). They communicate with each other by publishing and subscribing to ROS topics. Policies of POMDP and MDP models are trained prior to operation and executed by the framework modules. Model formulation and training is outlined further in Section 4.

The four modules are as follows: the *Global Planner* that provides macro actions, the *Map Management* module that maintains up to date information regarding the environment and processes the macro actions of the *Global Planner*, the *Local planner* that provides velocity commands for the UAV, and the *Low-Level Control* module that carries out the commands sent by the *Local Planner* and provides sensor data for the higher level modules. Because the modules run in parallel, each module can have its own update rate.

Global Planner Module

This module executes a pre-trained policy for the global MDP problem. Since the search task has a focus on searching confined or indoor environments (buildings, under the forest canopy, etc.) the global environment map is broken into discrete sub-maps. The *Global Planner* module receives observations from the *Map Management* module. The observations it receives are model dependent and are outlined further in Section 4.

The *Global Planner* module then produces an action (move to a specific room or search the current room), which is processed by the *Map Management* module. This process repeats until such time as the task is completed. Because the policies are pre-trained, the planner can use any policy that shares the same observation and action structure as the planner. This provides flexibility, so that if a collection of policies have been trained for common environment structures and the structure of the global environment is known (as is assumed for this task), the policy best suited to that global environment structure can be selected prior to run-time to maximize performance.

The *Global Planner* module operates at a pseudo-event-driven rate. After it gives an action, the planner is suspended until that action is completed by the system (eg. move, search, etc.). However, this is only one way to implement the global planner and the framework was designed with the intended flexibility to allow any number of *Global Planner* modules to function adequately.

The structure of the MDP formulation for the *Global Planner* is elaborated on further in Section 4.

Map Management Module

The *Map Management* module handles all the logic necessary to enable the *Global Planner* to effectively control the *Local Planner*, it also processes data received from the *Low-Level Control* module.

This module maintains a working map of the entire global environment, as well as sub-maps for each area within the global environment.For illustration purposes each sub-map is divided into 0.5m x 0.5m cells, with each cell representing a space of explorable environment. Currently, for illustrative purposes each sub-map has a maximum size of 10m x 10m (100 square meters), however this can be adjusted within the framework.

An *rtree* [10] index is maintained of all unexplored cells in each sub-map. Figure 2 shows how a map is divided into sub-maps and cells. Whenever a cell is explored by the UAV agent that cell is removed from the *rtree* index. Figure 3 shows a sub-map being explored by a UAV agent.

The *Map Management* module uses these cells to maintain an awareness of the search status of each sub-map for use by the *Global Planner*. It also uses them to produce observations for the *Local Planner* module of nearby areas within a sub-map that remain unexplored, which through the formulation of the local planner POMDP (discussed further in Section 4) causes the UAV to navigate to those regions, subsequently exploring them.

The *Map Management* module also processes move actions from the *Global Planner*. When a move action is received a *transit sub-map* is generated that creates a string of unexplored cells between the current location of the UAV and the target sub-map to move the UAV to the desired area. An example of a transit sub-map can be seen in Figure 3 This map behaves similarly to the regular sub-maps but is created at the instance of a global action. For illustration purposes the l1-path-finder algorithm [19] is used to generate a path of cells between two areas and sets them as unexplored, however other more complex methods can also be used. The UAV then only receives observations from the *transit sub-map*; when the UAV is using the correct policy it will follow the unexplored path of cells to the desired area, where it is switched back to that area's sub-map and can begin exploring again using a search policy.

To maintain knowledge of the UAV's position and produce the correct observations relative to the UAV agent, the *Map Management* module receives the global pose from the *Low-Level Control* module. In the case of the simulated trials outlined in section 5, this pose is the ground truth, however any relatively reliable global pose estimation would suffice. During testing gaussian noise is added to the ground truth position and heading in order to emulate noisy pose estimation.

This module operates at a rate of 20 Hz.

Local Planner Module

Much like the *Global Planner* module, the *Local Planner* module acquires observations (o) and produces actions (a) according to a given pre-trained policy (π).

The *Local Planner* receives observations of the UAV agent's current velocity from the *Low-Level Control* module, and receives observations regarding nearby unexplored areas and hazards from the *Map Management* module. With these observations it then outputs command velocities for the UAV agent in the x, y and *yaw* axes using its pre-trained policy. By using pre-trained policies the performance can be maximized for a given environment, changing search policies on a sub-

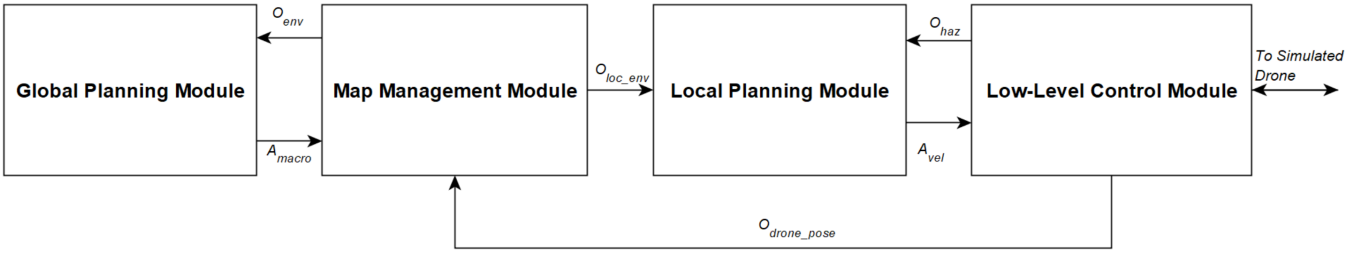


Figure 1: Module Architecture

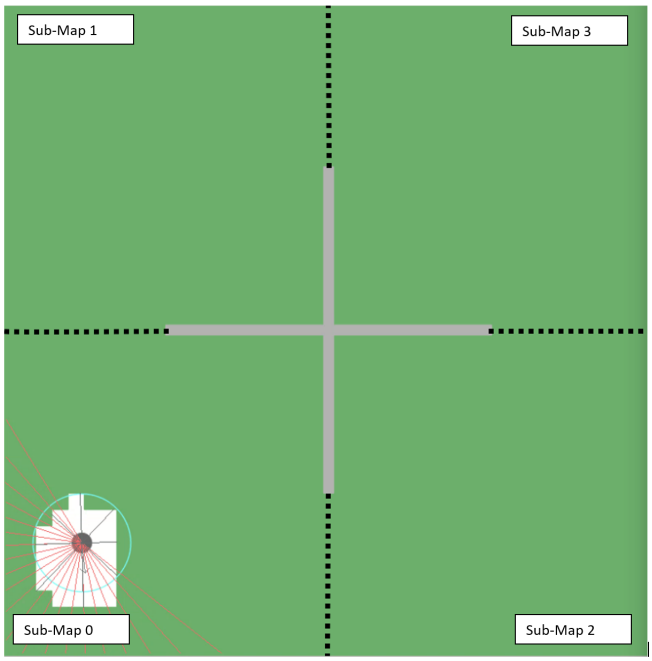


Figure 2: Environment Divided into Sub-Maps and Cells. **Green**: Unexplored Cells, **White**: Explored Cells, **Grey Circle**: UAV, **Blue Circle**: UAV View, **Grey Arrow**:UAV Heading, **Grey Lines**: Exploration Observations, **Red Lines**: Hazard Observations

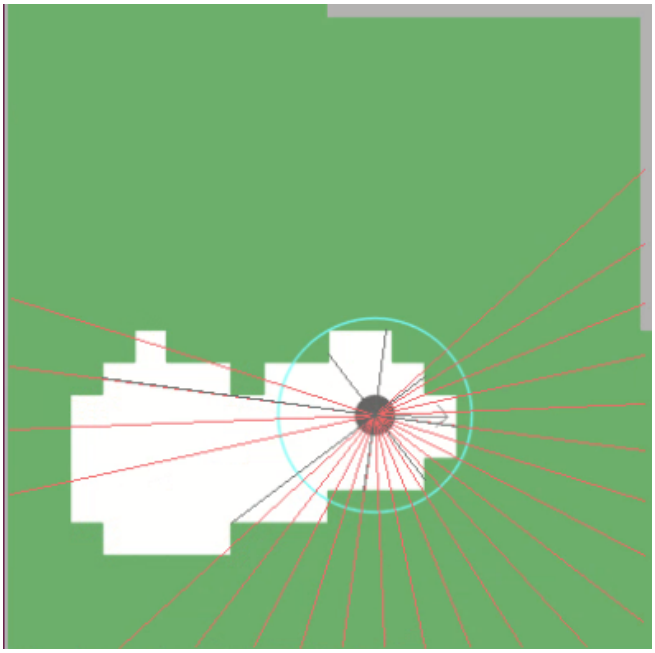


Figure 3: UAV Explores Sub-Map Zero. **Green**: Unexplored Cells, **White**: Explored Cells, **Grey Circle**: UAV, **Blue Circle**: UAV View, **Grey Arrow**:UAV Heading, **Grey Lines**: Exploration Observations, **Red Lines**: Hazard Observations

map by sub-map basis, or a single general policy can be used to navigate any number of environment configurations with reduced performance instead. Multiple POMDPs and policies are created for the *Local Planner* module depending on the problem. The environment searches illustrated by Cases 3a and 3b of Section 5 use two policies generated by two POMDP models. One policy is used for transit between sub-maps, and the other policy is used for searching the sub-maps. Case 3c implements three (3) policies, one (1) policy for transit and two (2) policies for searching each of the two (2) different room types.

A more detailed formulation of the POMDPs for the local planner is outlined in section 4.

This module produces actions at a rate of 10Hz.

Low-Level Control Module

The *Low-Level Control* module converts the *Local Planner* module velocity commands into motion controls for the UAV agent using the MAVROS/PX4 firmware. This firmware is also responsible for producing the velocity estimations for the *Local Planner* module and the global pose observation for the *Map Management* module. This module closes the loop of the framework and allows a properly configured system to search a multi-area environment within Gazebo.

The low-level control module also provides an interface for real-world testing.

4. Problem Formulation and Training

In the proposed framework, the UAV search problem is seperated into the global planning problem and the local planning problem and models are formulated for each. The development of these two models can be further broken down into training and execution.

To allow for rapid learning of a solution, the models must also incorporate approximations of the behaviour of the system during normal operation. This way, each model can be trained independently of any other nodes, decreasing learning times. These approximations are handled during the transition function (T) of each model, and are further outlined in the respective transition function sections for each problem.

Both models, their core parameters, and training parameters can be altered and adapted to tweak execution and training performance or produce solutions for different problem configurations. For illustration purposes the following global planner and local planner model configurations were used.

4.1 Global MDP

The global planning MDP can be defined as an pseudo-event-driven MDP, as it publishes an action and only receives observations and produces actions once the prior action is completed. The global planning MDP is a discrete MDP. The model is illustrated using a four room environment. Creating models that handle different sized environments is simply a matter of scaling the observations and actions as required.

Observations (O):

The set of observations can be broken further into three types:

Observation Type	No. Observations required for map with N rooms
The room map (static)	(2*N)/2 - N/2
UAV location	1
Search status of each room	N

The room map observation set outlined in the table above is only necessary for developing a generalized policy that can handle all unique variations of an N room environment. A model that was only learning one room layout would only need the current location of the UAV and the search status of each room as observations. Figure 4 outlines all the unique, connected, non-directional graph maps for a four room environment. The global MDP was trained using these six configurations to be able to handle any four room configuration.

Actions (A):

The action space for the global planner MDP is a discrete action space whose size is equal to the number of rooms in the environment, in the case of the four room environment the actions available are to search the current room, or move to another room. This can be represented by the following array [0,1,2,3]. Returning an action value from that array equal to the current location of the UAV causes the UAV to execute a search of that room, while returning an the value of a non-occupied room executes a move to that room.

Transition Function (T):

The transition function used for training the global planner does not consider uncertainty or the time an action takes. If a move to another room is requested, the UAV is moved to that room. If a search action is requested, the status of the current room is changed from 0 to 1, to indicate the room has been searched. In real-time operation the global planner is much the same, with the only exception being that rather than transition occuring instantly (as it would when learning), it happens only once the specified action has been completed by the rest of the system and has been reported as completed by the *Map Management* module.

Rewards (R):

The rewards for global planner MDP are as follows:

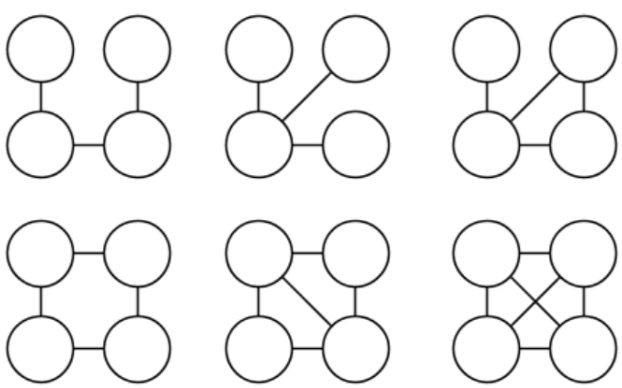


Figure 4: Unique Four Room Combinations

Rewards	
Searching an unsearched room	1
Searching a searched room	-1
Move cost	-0.1

4.2 Local Planner POMDPs

4.2.1 Search POMDP

The local planning search POMDP model is a more complex formulation than the global planning model, as the UAV operates in a continuous state space and has a continuous action space.

Observations (O):

The observations for the search POMDP can be broken into the following types:

Observation Type	No. Observations	Observation Range
Hazard Observations	36	0 to 6
Exploration Observations	8	0 to $submapsize * \sqrt{2}$
Velocities (x,y,yaw)	3	-0.5 to 0.5

Note that the UAV does not ever receive its global or local pose, as without also receiving a map reference these would provide no useful information. In an effort to reduce the state space and create a model that converges on a solution, the UAV only receives measurements of environmental features relative to its current position and heading.

The hazard observations are used in the detection of hazards. This model formulation has the hazard readings spaced at each 10 degree mark relative to the heading of the UAV. The same logic is applied to the exploration observations, spaced at every 45 degrees, as they are not imperative for collision avoidance. These readings give the UAV some idea of the shape of its surrounding environment at any given instant, allowing it to make decisions regarding which unexplored area it should investigate next, and where hazards it must avoid are.

The velocity observations are necessary for giving the UAV an idea of its current trajectory relative to the other observa-

tions and are provided in the local frame of the UAV.

In the case of this illustrative model the range of the UAVs hazard observations is that of an average low-price laser scanner, the range of the exploration observations is from 0 to the length of hypotenuse of a square sub-map to ensure that any unexplored cells are never out of range, and the range of the velocity observations are the limits of the UAV's simulated operating velocities.

Actions (A):

The UAV operates with the continuous action space detailed below:

Action Type	Action Space
Accelerate in the X axis	-0.25 to 0.25
Accelerate in the Y axis	-0.25 to 0.25
Accelerate in the Yaw axis	-0.25 to 0.25

This means that at any given step, the UAV can be given a command to accelerate up to 0.25m/s in any of its three degrees of freedom. However, the simulated UAV does not respond to velocity commands instantaneously and has effective top speeds for each axis. Therefore, a transition function is necessary when modelling the local planner.

Transition Function (T):

The transition function for the local planner approximates the UAVs response to the acceleration actions produced by the planner. First it calculates the difference (Δ) between the current velocity ($v_{current}$) and the new desired velocity ($v_{desired}$) and uses two precalculated transition tables (linear and angular) to determine the response of the system over the 0.1 second time-step. The transition function (T) assumes a linear change of velocity over this time period, taking the average of this new speed and the old speed to find the change in position of UAV in the environment and update the observations, ready for a new action.

The UAV is also subject to velocity limits within the transition function, and any acceleration command given to increase velocity past the maximum or minimum velocity limits (v_limit), sets the desired velocity at those limits. This speed limit also exists in the local planning module, to permit consistent operation across learning and execution.

Calculating the transition tables is not essential to the operation of the framework, however it can reduce sub-optimal performance that can be a result of the policy not learning the approximate physical transition characteristic of the agent, such as overshoot during maneuvers.

The transition tables were calculated as follows:

- First a range of angular and linear response data is collected while the UAV agent operates in Gazebo.
- Secondly the data is loaded into matlab and the System Identification Toolbox is used to generate a discrete state space model for the linear and angular data sets respectively.
- Finally the step response for each state space model is generated, shown in Figure 5 and Figure 6.

The following is performed for each step response to create an approximate transition table:

- Split the step response into approximate logical divisions

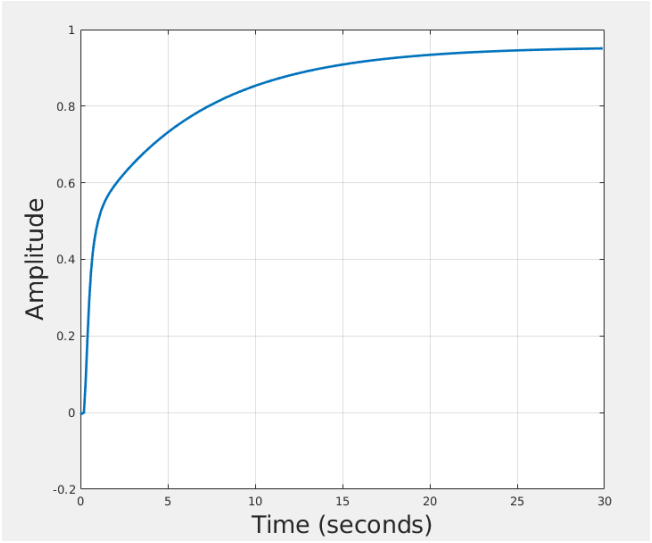


Figure 5: Linear Step Response

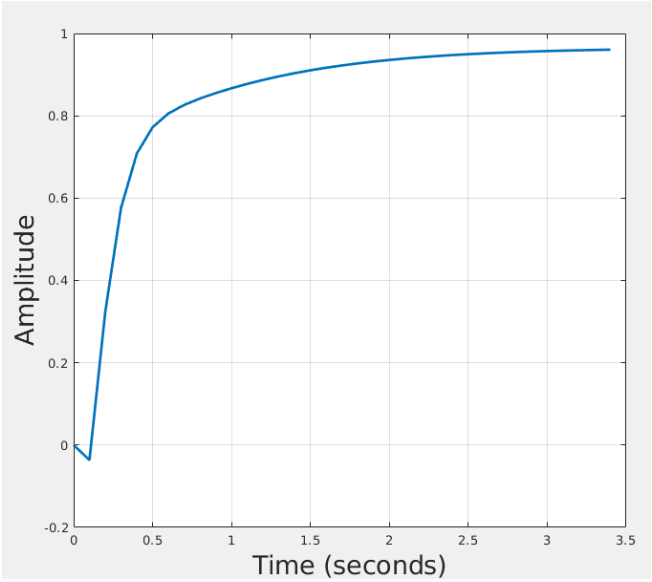


Figure 6: Angular Step Response

based on the delta of the system from the max response (0.8, 0.7, 0.6, 0.55 etc.) and data available.
- Calculate the response coefficient τ at each time step.

$$\tau = v_{\Delta}/v_{initial}$$

- Find the average response coefficient for all time steps between each pre-chosen delta and the next, setting this value as the step response coefficient for the larger of the two.
- Create the transition tables using the average step response coefficients and delta values.

Transition Tables for Simulated 3DR IRIS UAV	
Linear Transition Table	
$v_\Delta(m/s)$	Response Coefficient τ
1	0.1080
0.7	0.1034
0.6	0.07008
0.55	0.04407
0.5	0.02781
0.45	0.01686
0.4	0.01397
0.35	0.01343
0.3	0.01303
0.25	0.01246
0.2	0.01154
0.15	0.00983
0.1	0.00465
0.05	0.00156
Angular Transition Table	
$v_\Delta(rad/s)$	Response Coefficient τ
1	0.3486
0.4	0.3112
0.3	0.1830
0.2	0.0935
0.15	0.0789
0.1	0.0588
0.05	0.02697

The model uses the following to calculate the velocity after each time step $v_{t+0.1}$:

$$v_\Delta = |v_{desired} - v_t|$$

$$v_{t+0.1} = v_t + v_t * coefficient_{response_{v_{delta}}}$$

The transition function is also responsible for calculating hazard and environment exploration observations, rather than receiving them from an external node. Calculating these measurements requires the model to create and maintain an exploration map and a hazard map for each run. In the case of both the exploration and hazard maps, the logic for producing exploration and hazard observations is the same as the logic within the map management module, with the current position of the UAV being used to calculate the distance from the UAV to the unexplored regions and hazards.

Rewards (R):

The rewards for the local planning model are as follows:

Rewards	
Exploring an unexplored cell	10
Colliding with a hazard	-500

4.2.2 Transit POMDP

The *Transit POMDP* is used for controlling the UAV during sub-map transits. It has the same formulation as the *Search POMDP* except it has an altered observation space. All actions, rewards and the transition function remain the same.

Observations (O):

The observations for the *Transit POMDP* can be broken into the following sets:

Observation Type	No. Observations	Observation Range
Hazard Observations	36	0 to 2.5
Nearest Distance to Nearest Transit Cell	8	0 to $submapsize * \sqrt{2}$
Angle to Nearest Transit Cell	1	-π to π
Velocities	3	-0.5 to 0.5

The *Transit POMDP* has the same number of hazard observations, but they are reduced in range, as the UAV agent only cares about hazards when collision is imminent. Also, rather than generate observations for all cells in the transit sub-map the agent only receives the distance and angle to the nearest cell in the sub-map, as transit sub-map consists of a string of cells to the desired location.

4.3 Training

Both the global planning and local planning model have been implemented using the Open AI Gym [4] environment structure, so that any number of learning algorithms can be tested for performance.

For validation purposes we chose to use the deep reinforcement learning algorithm, TRPO, implemented in rllab to train policies for both models. When training both models, the initial starting conditions are randomized. In the case of the global planner, this means selecting 1 of 6 of the available map types and randomizing the starting location of the UAV. In the case of the local planner, the UAV location is randomized within a 10m x 10m sub-map.

Randomizing the initial conditions of the models when training a policy allows for better policy generation. Due to the nature of the observations the local planner receives (relative measurements to environmental features) a policy can be learnt that searches a 10m x 10m environment with no obstacles, and with a single obstacle.

The models also incorporated noise in the pose of the UAV, to simulate uncertain sensing during operation.

Global Planner MDP:

The neural net model for the planner has two (2) hidden layers, each with eight (8) neurons. A 2 layer neural net was selected as the system is not complex enough to warrant any more layers, and the number of neurons was based around the number of input observations and output actions of the model. The TRPO algorithm is also used for developing the control policy (π), with the following training parameters:

Parameter	Value
Batch Size	2000
No. Iterations	500
Discount Factor	0.99
Step Size	0.01
No. Parallel	4

Local Planner POMDPs:

The neural net model for the planner has two (2) hidden layers, each with 32 neurons. The TRPO algorithm is used for developing the control policy (π) and the training parameters

are as follows:

Parameter	Value
Batch Size	2000
No. Iterations	2000
Discount Factor	0.99
Step Size	0.01
No. Parallel	6

These training parameters were used for the *Transit* and all *Search* POMDP models.

The *Search POMDP* was trained in a number of configurations. The configuration can be broken down into *approximate* or *basic*, and then further broken down into *specialised* or *general*.

The *approximate* configurations used the transition tables outlined above for calculating the change in velocity over a time step whereas the *basic* configurations assumed all desired changes in velocity are met over one time step.

The *general* and *specialised* configurations determined the generation of the environment during training. A *general* configuration assumes the agent will be operating in an environment containing one or zero obstacles and generates random environments for training accordingly. A *specialised* configuration assumes the agent will be operating in an environment containing either no obstacles or one obstacle. Between the *approximate* and *basic*, and *generalised* and *specialised* training configuration distinctions, six (6) *Search* policies were trained for testing in Case 1 (Section 5):

Search Policies	
Approximate	Basic
General	General
Specialised: 0 Obstacles	Specialised: 0 Obstacles
Specialised: 1 Obstacle	Specialised: 1 Obstacle

The *Transit* model was trained using only one configuration. The environment was static, but the agent had to follow randomly generated paths (*Transit Sub-Maps*) through the static environment.

Low Batch and Iteration sizes for the model training was a consequence of limited training time available and resulted in policies that could be improved(outlined in section 5).

5. EVALUATION

Case 1: Testing the Search Policies

The first test case was conducted to compare the performance of the six types of trained search policies in a simulated Gazebo environment. This assists in determining the effectiveness of the *approximate/basic* and *general/specialised* configurations.

There are two test environments. An empty 10m x 10m area, and a 10m x 10m area with a single obstacle in the middle. All policies were trained with the following parameters. Only the environment generation and transition function differed between configurations.

Key Model Parameters	
Parameter	Value
Environment Observations	8
Hazard Observations	36
Penalty for collision	500
Reward for explored cell	10
Time Limit	180 Seconds (1800 steps)

Training Parameters	
Parameter	Value
Batch Size	2000
No. Iterations	2000
Discount Factor	0.99
Step Size	0.01
No. Parallel	6

The specialised policies were tested only in their trained environment, while the general policies were tested in both environments. Each policy was tested five (5) times per testing environment to get an idea of policy performance. Each test was run for three (3) minutes (180 seconds) or until the agent collided with a hazard. The percentage of the environment searched before the time limit had been reached or until the agent crashed has been listed along with whether the agent collided with the environment. Full tabulated results for the trials can be found in APPENDIX A. Below are the averages and totals for the environment tests per policy. Policies were tested without noise in the UAV agent's pose, and then tested again with noise present. The UAV is assumed to only have a downward facing camera and the ability to estimate its pose within the environment.

Zero Obstacle Environment Totals		
Approximate General Policy		
Avg Time (Sec)	Avg % Area Searched	Collisions
Clean Pose		
111.5	57.2	3
Noisy Pose		
90.9	61.2	5
Basic General Policy		
Avg Time (Sec)	Avg % Area Searched	Collisions
Clean Pose		
149.316	73.8	1
Noisy Pose		
180	88.2	0
Approximate Zero Obs Specialised Policy		
Avg Time (Sec)	Avg % Area Searched	Collisions
Clean Pose		
168.508	57.8	1
Noisy Pose		
151.038	52	1
Basic Zero Obs Specialised Policy		
Avg Time (Sec)	Avg % Area Searched	Collisions
Clean Pose		
35.8	39.6	5
Noisy Pose		
42.4	42.4	5

Single Obstacle Environment Totals		
Approximate General Policy		
Avg Time	Avg % Area Searched	Collisions
Clean Pose		
111	36.6	2
Noisy Pose		
111.1	26.4	2
Basic General Policy		
Avg Time	Avg % Area Searched	Collisions
Clean Pose		
70	48.6	4
Noisy Pose		
66.5	42.8	4
Approximate Single Obs Specialised Policy		
Avg Time	Avg % Area Searched	Collisions
Clean Pose		
180	69	0
Noisy Pose		
166	64.2	1
Basic Single Obs Specialised Policy		
Avg Time	Avg % Area Searched	Collisions
Clean Pose		
21.2	23	5
Noisy Pose		
24.3	30.6	5

In the zero obstacle test the *Basic General Policy* was the most performant policy, resulting in a single collision and searching on average the most area in the alloted 180 second flight time. The *Approximate Zero Obstacle Specialised Policy* was the next most cautious, colliding twice. However, the cautious behaviour resulted in a lower average area searched over the flight time than the *Basic General Policy*. The *Approximate General Policy* underperformed, with neither positive characteristic of these policies. This is likely due to the the randomisation of the obstacle generation during training and the low batch size. While the *Basic General Policy* had been able to develop comparatively aggressive search behaviour for zero obstacle environments and the ability to avoid collisions the *Approximate General Policy* had not yet developed this performance. The *Basic Zero Obstacle Specialised Policy* also underperformed. The *Basic General Policy* was selected for illustration when searching zero obstacle environments during *Test-Case 3*, as it empirically has the best performance characteristics for that environment type.

The single obstacle environment tests indicate the benefit of specialised policies, with the general policies underperforming in this testing environment. The only standout of the policies was the *Approximate Single Obstacle Specialised Policy*, searching on average over 50% of the single obstacle environment and colliding with hazards only once. The other policies exhibited low search capabilities and inconsistent obstacle avoidance. The *Approximate Single Obstacle Specialised Policy* was selected for illustration when searching single obstacle environments during *Test-Case 3*.

Case 2: Transit Evaluation

The second test case was conducted to determine the functionality of the *Transit Policy*. A four room environment was used, shown in Figure 7. The UAV started in the upper left of the map (Sub-Map 1), and had to traverse to the bottom right of the map (Sub-Map 2). Figure 8 illustrates the initial *transit sub-map* generated by the *Map Management* module.

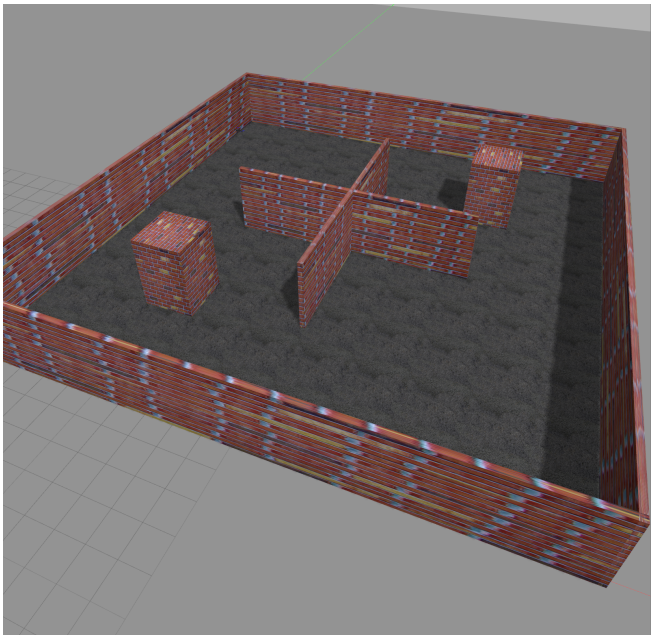


Figure 7: Transit Test Environment

Ten (10) trials were run, five (5) with noisy pose readings and five (5) with clean pose readings, with the UAV traversing from Sub-Map 2 to Sub-Map 1, until the UAV collided with the environment or reached the destination. The results can be seen below:

Transit Test Results		
Clean Pose		
Trial	Time (Sec)	Destination Reached?
1	59.3	Yes
2	83.2	Yes
3	69.6	Yes
4	81.9	Yes
5	57.38	Yes
Noisy Pose		
1	75	Yes
2	77.1	Yes
3	58.7	Yes
4	87.8	Yes
5	80.6	Yes

As can be seen by these results, the transit policy combined with the *Map-Management* module is capable of traversing to desired rooms, enabling the functionality of the *Global Planning* module.

Case 3: Framework Evaluation

The final case was conducted to determine the functionality of the framework, combining the functionality of all modules to perform a searh of multi-room environments. This test involves searching 3 different environments, shown in Figure 9. For illustration purposes, in each environment the UAV has to search a sub-map to at least 50% completion before the *Map Management* module marks it as searched. Once the current action (search or move) has been completed the *Global Planning* module provides another macro action to the *Map Management* module. The *Local Planning* module implements a policy for searching or moving as necessary. When searching a zero obstacle room the *Local Planning*

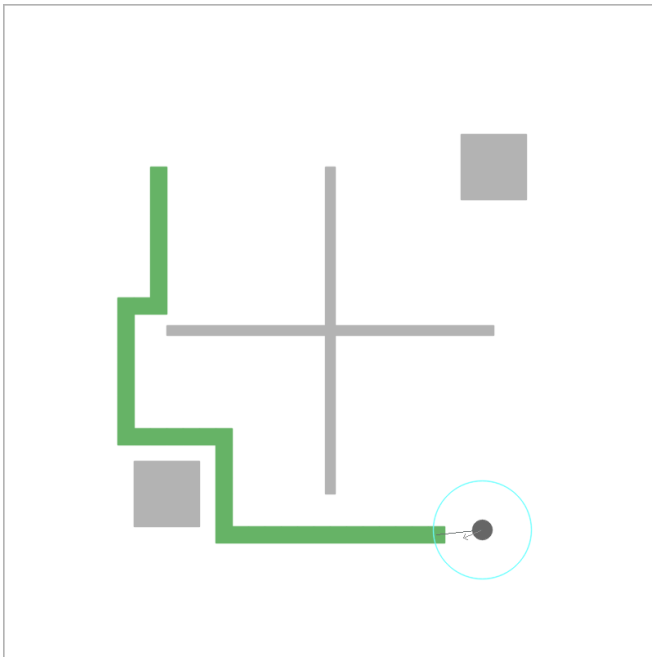


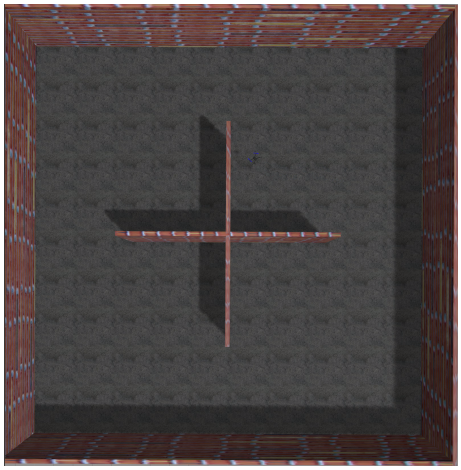
Figure 8: Transit Sub-Map

module uses the *Basic General Policy*, uses the *Approximate Single Obstacle Specialised Policy* when searching single obstacle environments and uses the *Transit Policy* when moving between sub-maps. The framework was validated on the three different environments with clean and noisy pose configurations. The time taken to complete a search of the environment to at least 50 % in each sub-map is listed below.

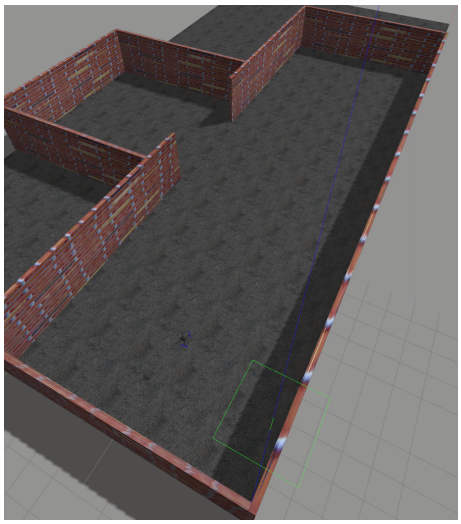
Framework Validation Results	
Four Room Environment with Central Walls	
Time to Completion (Sec)	Noisy Pose?
355	No
305.1	Yes
Hall Environment with Adjoined Room	
Time to Completion (Sec)	Noisy Pose?
298.8	No
341.97	Yes
Four Room Environment with Central Walls and Obstacles	
Time to Completion (Sec)	Noisy Pose?
357.02	No
358.18	Yes

For illustrative purposes Figure 10 outlines the progress of the UAV through the first environment from the perspective of the map-management module. Figure 10(a) shows the UAV at the beginning of the search. All Sub-Maps remain unexplored. Figure 10(b) shows the search progress of the UAV in Sub-Map 0. At this time the search threshold of 50 % is reached and the *Transit Sub-Map* show in Figure 10(c) is generated to move the UAV to Sub-Map 2. The framework then repeats this process, moving the UAV and having it search each Sub-Map until at least half of each Sub-Map has been explored.

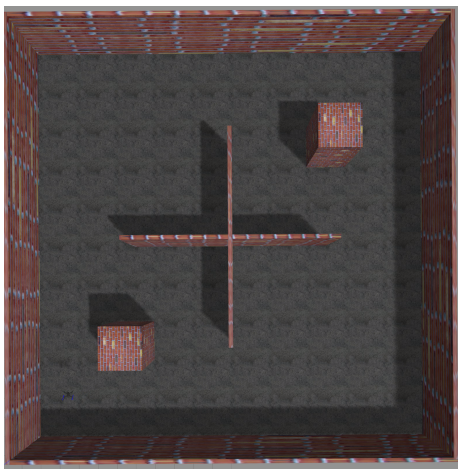
A video of the environment searches is available at the following link: https://youtu.be/V5pFpt8cD1M.



(a) Test Environment One: Four Rooms with Central Walls

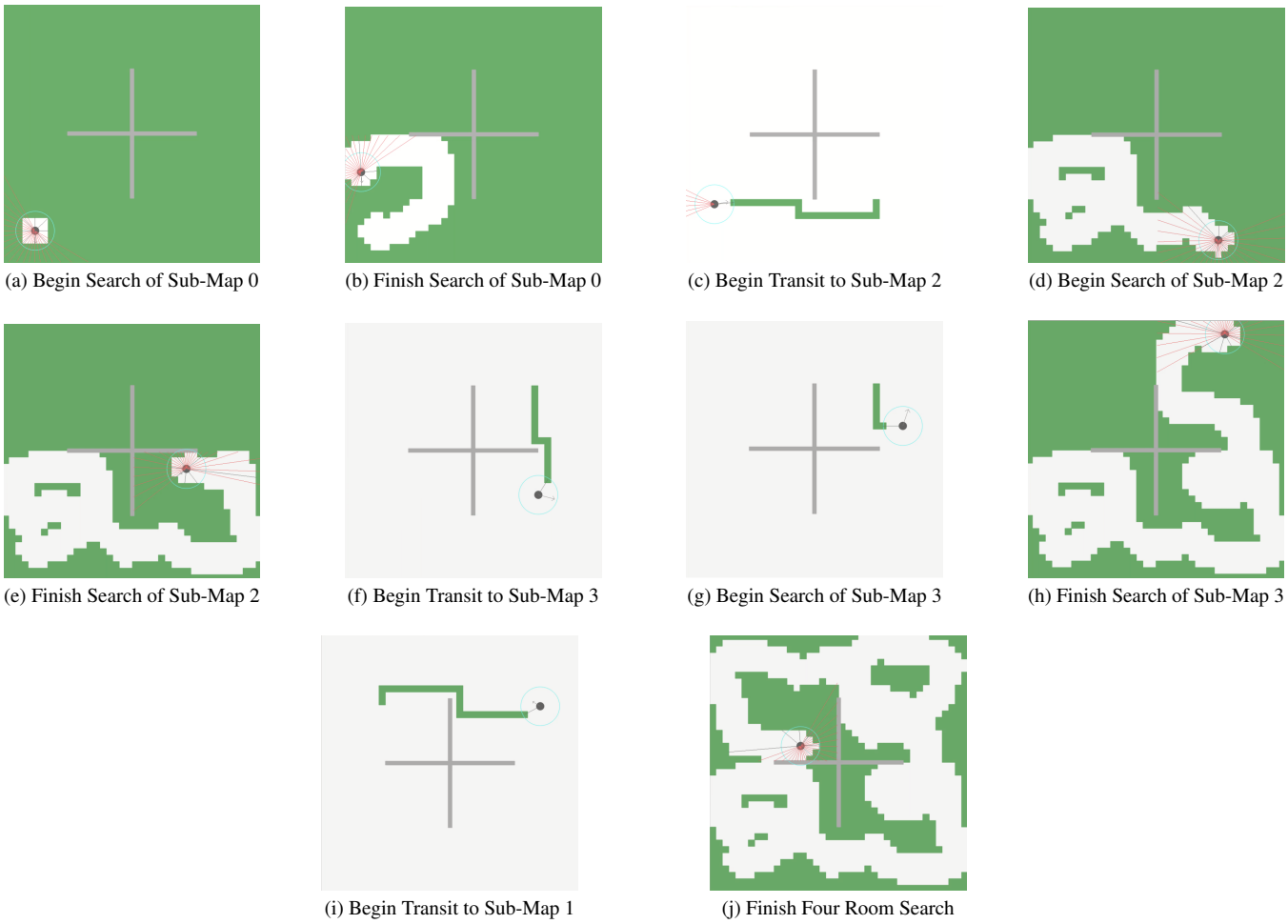


(b) Test Environment Two: Hall with Adjoined Room



(c) Test Environment Three: Four Rooms with Central Walls and Obstacles

Figure 9: Case 3 Test Environments



(a) Begin Search of Sub-Map 0
(b) Finish Search of Sub-Map 0
(c) Begin Transit to Sub-Map 2
(d) Begin Search of Sub-Map 2
(e) Finish Search of Sub-Map 2
(f) Begin Transit to Sub-Map 3
(g) Begin Search of Sub-Map 3
(h) Finish Search of Sub-Map 3
(i) Begin Transit to Sub-Map 1
(j) Finish Four Room Search

Figure 10: Case 3: Four Room Search, Map-Management View

6. CONCLUSIONS

The framework detailed by this paper has been shown to enable a simulated agent to search a multi-room environment using policies generated using deep reinforcement learning. Test Case 1 shows that a briefly trained policies were capable of searching zero and single obstacle environments. Test Case 2 shows that with the correct framework configuration and policy the UAV is able to transit between rooms, and Test Case 3 shows that the framework supports searching multiple room environments by combining high level and low level planning.

Further work will include improving performance of the local planner policy, determining good policy generation practices through testing and investigation of real-time policy switching for increased range of agent operation. Future work will also include testing the framework on different simulated UAV platforms, validation testing of the framework on hardware in real-world environments, increasing the number of agents supported by the global-planner, and testing the search capability of the framework using multiple simulated UAVs.

REFERENCES

[1] Christopher Amato, George Konidaris, Gabriel Cruz, Christopher A Maynor, Jonathan P How, and Leslie P Kaelbling. Planning for decentralized control of multiple robots under uncertainty. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1241–1248. IEEE, 2015.

[2] Haoyu Bai, Shaojun Cai, Nan Ye, David Hsu, and Wee Sun Lee. Intention-aware online pomdp planning for autonomous driving in a crowd. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 454–460. IEEE, 2015.

[3] Richard Borie, Craig Tovey, and Sven Koenig. Algorithms and complexity results for graph-based pursuit evasion. *Autonomous Robots*, 31(4):317, 2011.

[4] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[5] Adrian Carrio, Carlos Sampedro, Alejandro Rodriguez-Ramos, and Pascual Campoy. A review of deep learning methods and applications for unmanned aerial vehicles. *Journal of Sensors*, 2017, 2017.

[6] Min Chen, Emilio Frazzoli, David Hsu, and Wee Sun Lee. Pomdp-lite for robust robot planning under uncertainty. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 5427–5433. IEEE, 2016.

[7] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pages 1329–1338, 2016.

[8] Pascal Gohl, Michael Burri, Sammy Omari, Joern Rehder, Janosch Nikolic, Markus Achtelik, and R Siegwart. Towards autonomous mine inspection. In *Applied Robotics for the Power Industry (CARPI), 2014 3rd International Conference on*, pages 1–6. IEEE, 2014.

[9] Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 66–83. Springer, 2017.

[10] Antonin Guttman. *R-trees: A dynamic index structure for spatial searching*, volume 14. ACM, 1984.

[11] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. *CoRR, abs/1507.06527*, 7(1), 2015.

[12] Ruijie He, Emma Brunskill, and Nicholas Roy. Puma: Planning under uncertainty with macro-actions. In *AAAI*, 2010.

[13] Yufeng He, Qinghua Zeng, Jianye Liu, Guili Xu, and Xiaoyi Deng. Path planning for indoor uav based on ant colony optimization. In *Control and Decision Conference (CCDC), 2013 25th Chinese*, pages 2919–2923. IEEE, 2013.

[14] Amanda Hodgson, Natalie Kelly, and David Peel. Unmanned aerial vehicles (uavs) for surveying marine fauna: a dugong case study. *PloS one*, 8(11):e79556, 2013.

[15] Geoffrey Hollinger, Athanasios Kehagias, and Sanjiv Singh. Probabilistic strategies for pursuit in cluttered environments with multiple robots. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3870–3876. IEEE, 2007.

[16] Klaas Kelchtermans and Tinne Tuytelaars. How hard is it to cross the room?–training (recurrent) neural networks to steer a uav. *arXiv preprint arXiv:1702.07600*, 2017.

[17] Hanna Kurniawati and Vinay Yadav. An online pomdp solver for uncertainty planning in dynamic environment. In *Robotics Research*, pages 611–629. Springer, 2016.

[18] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[19] Mikola Lysenko. l1-path-finder algorithm. `https://mikolalysenko.github.io/l1-path-finder/www/`. Accessed: 2018-09-24.

[20] Huy X Pham, Hung M La, David Feil-Seifer, and Luan V Nguyen. Autonomous uav navigation using reinforcement learning. *arXiv preprint arXiv:1801.05086*, 2018.

[21] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.

[22] David Silver and Joel Veness. Monte-carlo planning in large pomdps. In *Advances in neural information processing systems*, pages 2164–2172, 2010.

[23] Lei Tai, Shaohua Li, and Ming Liu. A deep-network solution towards model-less obstacle avoidance. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 2759–2764. IEEE, 2016.

[24] Teodor Tomic, Korbinian Schmid, Philipp Lutz, Andreas Domel, Michael Kassecker, Elmar Mair, Iris Lynne Grixa, Felix Ruess, Michael Suppa, and Darius Burschka. Toward a fully autonomous uav: Research platform for indoor and outdoor urban search and rescue. *IEEE robotics & automation magazine*, 19(3):46–56, 2012.

[25] Fernando Vanegas, Duncan Campbell, Markus Eich, and Felipe Gonzalez. Uav based target finding and tracking in gps-denied and cluttered environments. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 2307–2313. IEEE, 2016.

7. BIOGRAPHY

Ory Walker *received their B.S. in Aerospace Engineering from Queensland University of Technology in 2015 and is currently a PhD candidate affiliated with the Australian Research Centre for Aerospace Automation (ARCAA) at Queensland University of Technology (QUT). Their research focuses on UAV planning and control problems for single and multiple agents in indoor environments, modelled as POMDPs.*

Fernando Vanegas *is a Mechatronics Engineering from UMNG in 2004. He holds a M.Sc. in Electrical Engineering from Halmstad University in 2008 and a Ph.D. in Aerial Robotics in 2017 from Queensland University of Technology. His current research activities include motion planning for UAV in cluttered and uncertain environments modelled as POMDPs for ecological applications.*

Felipe Gonzalez *is an Associate Professor (UAV) in the Science and Engineering Faculty, QUT and the QUT UAV Remote Sensing Group. He holds a BEng (Mech) and a PhD from the University of Sydney. His research explores bioinspired optimization, uncertainty based UAV path planning and UAV for environmental monitoring. He currently leads the ARC DP project UAV flying under the canopy and urban jungle. Felipe is a Chartered Professional Engineer and member of professional organizations including the RAS,IEEE and AIAA.*

Sven Koenig *is a professor in computer science at the University of Southern California. Most of his research centers around techniques for decision making (planning and learning) that enable single situated agents (such as robots or decision-support systems) and teams of agents to act intelligently in their environments and exhibit goal-directed behavior in real-time, even if they have only incomplete knowledge of their environment, imperfect abilities to manipulate it, limited or noisy perception or insufficient reasoning speed. Additional information about Sven can be found on his webpages: idm-lab.org.*

8. APPENDIX

APPENDIX A - Case 1 Results

Zero Obstacle Environment Results			
Approximate General Policy			
Clean Pose			
Trial	Time (Sec)	% Area Searched	Collided?
1	180	71	No
2	56.96	40	Yes
3	59.67	50	Yes
4	180	70	No
5	81.02	55	Yes
Noisy Pose			
1	92.3	56	Yes
2	119.1	84	Yes
3	113.7	87	Yes
4	33.5	31	Yes
5	95.97	48	Yes
Basic General Policy			
Clean Pose			
Trial	Time (Sec)	% Area Searched	Collided?
1	180	87	No
2	180	86	No
3	180	87	No
4	26.5	30	Yes
5	180	79	No
Noisy Pose			
1	180	86	No
2	180	95	No
3	180	85	No
4	180	87	No
5	180	88	No
Approximate Zero Obs Specialised Policy			
Clean Pose			
Trial	Time (Sec)	% Area Searched	Collided?
1	180	50	No
2	180	66	No
3	180	54	No
4	122.5	61	Yes
5	180	58	No
Noisy Pose			
1	180	51	No
2	35.1	43	Yes
3	180	52	No
4	180	55	No
5	180	59	No
Basic Zero Obs Specialised Policy			
Clean Pose			
Trial	Time (Sec)	% Area Searched	Collided?
1	28.7	33	Yes
2	35.06	40	Yes
3	29.8	34	Yes
4	29	33	Yes
5	56.4	58	Yes
Noisy Pose			
1	42.1	45	Yes
2	28.9	36	Yes
3	29.8	36	Yes
4	30.9	35	Yes
5	52.69	60	Yes

Single Obstacle Environment Results			
Approximate General Policy			
Clean Pose			
Trial	Time (Sec)	% Area Searched	Collided?
1	7.5	15	Yes
2	180	64	No
3	180	56	No
4	180	38	No
5	7.13	10	Yes
Noisy Pose			
1	180	51	No
2	180	20	No
3	7.3	10	Yes
4	180	40	No
5	8.1	11	Yes
Basic General Policy			
Clean Pose			
Trial	Time (Sec)	% Area Searched	Collided?
1	23.3	26	Yes
2	32.9	38	Yes
3	55.7	55	Yes
4	180	70	No
5	57.3	54	Yes
Noisy Pose			
1	22.9	28	Yes
2	32	34	Yes
3	75.4	62	Yes
4	180	63	No
5	22.1	27	Yes
Approximate Single Obs Specialised Policy			
Noisy Pose			
Trial	Time (Sec)	% Area Searched	Collided?
1	180	66	No
2	180	71	No
3	180	63	No
4	180	69	No
5	180	76	No
Noisy Pose			
1	180	55	No
2	180	70	No
3	180	65	No
4	111.4	66	No
5	180	65	No
Basic Single Obs Specialised Policy			
Clean Pose			
Trial	Time (Sec)	% Area Searched	Collided?
1	18.7	23	Yes
2	26	15	Yes
3	22.8	27	Yes
4	21.9	29	Yes
5	16.7	21	Yes
Noisy Pose			
1	31.7	41	Yes
2	21.8	27	Yes
3	21.1	30	Yes
4	24.3	29	Yes
5	22.7	26	Yes