Learning based modular framework for environment-adaptive planning in exploration

Sara Misra

CMU-RI-TR-20-27

July, 2020

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Matthew Travers, Co-chair
Howie Choset, Co-chair
Maxim Likachev
Shohin Mukherjee

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Abstract

Autonomous planning, has spawned a number of different solutions in robotics research, using different paradigms and strategies, both generalized and specific to certain problems, representations, and environments.

Thus, we pose the hypothesis that the hyperparameters for best performance are dependent on the type of environment determined by its "clutteredness". Usually, these hyperparameters are manually tuned by trial and error for a general estimate of the type of environments likely to be encountered by the agent, not accounting for different environment types within the same map. While the planner may operate within optimal bounds, the absolute best performance of the planner on every map in the dataset is unguaranteed.

This problem becomes evident and significant in exploration tasks, where the agent could encounter different environments during the same task. For exploration tasks, especially search-and-rescue, efficient navigation is critical, which is to have the highest or best performance in every portion of the map explored while looking for objects of interest. To address this issue this work presents a modular framework to utilize the experience of the underlying structure and the corresponding planning strategies performances of prior environments to predict the unknown map and adapt the planning strategy for maintaining high performance throughout the current exploration task.

In this thesis, we evaluate this framework using A* with the inflation factor hyperparameter being adapted to the environment in the task to see improvement over static hyperparameters. This framework can be extended to other heuristic based planners and even sets of planners where the adaptation is in which planner is being used in the environment rather than the hyperparameters of the planner model.

Acknowledgments

I would like to express my gratitude to everyone who directly or indirectly has supported my master's thesis completion in these challenging times.

I am deeply grateful to my advisors Professor Matthew Travers and Professor Howie Choset for providing the opportunity to dive into academic research from my freshman year and their solid support. These years and this thesis, on the accelerated schedule, were only possible with their unwavering guidance. My special thanks to the committee members, Professor Maxim Likachev and Shohin Mukerjee for providing valuable feedback in our discussions.

I cherish the time I have spent in the Biorobotics Lab, in particular with my CMU-Boeing Collaboration teammates: Lu Li, Haowen Shi, Michael Schwerin, Albert Xu, Daqian Cheng, Jim Picard. And my deep thanks extend to Guillaume Sartoretti, Shuo Yang, Daniel Vedova, Hans Kumar, Raunaq Bhirangi, Benjamin Freed, Michael Tatum, Barbara Fecich, and many others who have worked with, inspired and supported me throughout my journey at Carnegie Mellon University.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Robot autonomy, to solve the problem of path planning, has spawned several different solutions using different paradigms and strategies [17], with intense research focus on improving these tractable algorithms by looking at the worst-case performance guarantee metric. Specifically including metrics such as computational or asymptotic optimality [9] [15], probabilistic completeness [16] of the algorithm or its computational complexity [3].

However, these metrics capture a complex representation of algorithm performance over variations in the environment of the planning task. In this work, we focus on more fundamental metrics, time performance, and path quality. Prior work, Hsu et al. shows that the finite time takes to find a path and the path quality in a simple planning problem is dependant on the connectivity of the state-space, which is affected by various parameters such as obstacle configuration, robot dynamics and start/goal position. Consequently, the effectiveness of the proposed planning algorithms is heavily reliant on the underlying environment structure. With an increase in variability of the type of environments the robot faces, the fluctuation in planner performance is exacerbated.

Considering the variability of encountered real-world environments, there is motivation to ask how we can design a framework to leverage the qualities and performance of the given planner strategies to maximize expected performance, based on the metrics mentioned prior, on the distribution of environments encountered by the robot, or agent.

Machine learning procedures have been used to develop novel algorithms to adapt both sampling and search based planning algorithms to a specific environments [31] [4], where the novel algorithm learns from previously solved planning problems. This thesis leverages this strategy of picking the best element from a static set to design a loosely-coupled framework over a novel algorithm. Thus, in this context we aim to use machine learning methods to learn planner performance over planning problem distributions and their underlying environment structure to adapt the planning strategy for better expected performance over these distributions.

1.1 Motivation

This thesis asks the proposed question in the context of robot autonomy in unknown or partially known environments. Autonomous systems operating in such conditions face a variety of environments, or scenarios within the same mission and vary across different missions. Consider the example of the DARPA Subterranean Challenge, where agents autonomously navigate and explore unknown environments to find multiple goals or objects of interest whose locations are unknown beforehand. The applications can vary from disaster search-and-rescue to mapping of hazardous environments.

The encountered scenario can vary widely and change abruptly within the same mission, especially in the applications mentioned above, going from structured to unstructured, complex to open suddenly. As the navigation environment is partially known due to limited onboard sensing, there is a possibility for sudden changes to the prior belief of the environment state and as such using a singular planning strategy with static hyperparameters becomes infeasible due to the fluctuating navigation performance and the requirement of repeated manual tuning of algorithm hyperparameters for every mission.

Additionally, such robots have complex state spaces and limited computation budgets. Thus it is hard to engineer a general motion planner that has consistent performance across the distribution of planning problems. This leads us to consider an adaptive framework to answer the initial question. That is a framework that changes its search strategy according to the expected performance on the distribution of environments and planning problems.

1.2 Problem Statement

The path planning problem being attempted is simplified in the control space of the system by assuming a point robot in 2D space. Then let $\mathcal{X} \in \mathcal{R}^2$ be the state space of the robot in euclidean space. Given the objective is to find a trajectory σ for the robot from start to goal.

Let us define the planning problem composed of the environment, start and goal as Φ, and a planner as p then the trajectory $\sigma = p(\Phi)$. Each σ has a computation and path cost to it which will form the basis of our performance metrics.

With these definitions we can describe the distribution of planning problems $P(\Phi)$ by:

1. Drawing samples of environments from a generative distribution N times to form a series of $\{\Phi\}_{i=1}^{N}$ planning problems
2. Classify series of input maps and sample start and goal positions on each to form a series of $\{\Phi\}_{i=1}^{N}$ planning problems

The expected performance is calculated as an expectation of the cost of the planner (defined by the performance metric) over the planning problem distribution $P(\Phi)$.

1.3 Contribution

We summarize the main contributions of this work and the novel connections between the planning and learning open-loop control.

1. **Adaptive prediction framework over eclectic environments**: A designed framework to leverage learning prior experiences of the agent to maximize expected performance over the distribution of planning environments, given a set of strategies using both planner performance and map prediction.
2. **Loosely-coupled Modular system design**: a modular framework for plug-and-play application over multiple onboard sensor configurations, different exploration tasks and adaptive over different planner(s) or hyperparameter(s) of one planner for their standard implementations as shown in Figure 1.1

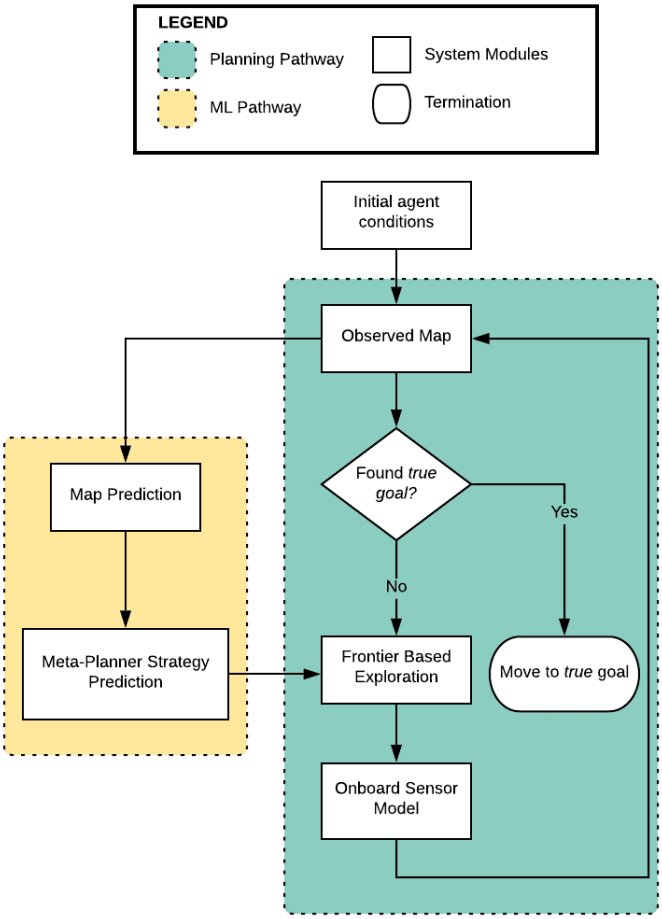

Figure 1.1: Visualization of the system framework

Chapter 2

Related Work

In this section we detail out prior methods and work done in map representation, adaptive motion planning, map prediction and frontier-based exploration.

2.1 Map Representation

In the scope of this thesis, the agent operates in a static 2D map is standardly represented in robotics literature [32] as a discrete or an occupancy grid where each cell contains an occupancy probability. We take the cell probability of one to mean occupied space (wall), of zero to mean free space (traversable), and of 0.5 to mean unknown or unexplored area throughout this paper.
Occupancy grid maps are useful representations for sensor mapping and combining different sensor modalities. So an occupancy grid map representation for the proposed system enables modularity concerning the sensor systems carried by the agent.

2.1.1 Occupancy Grid Mapping

Now that we have an occupancy grid map to navigate in and an onboard sensing array, we need an algorithm to transmute the sensor measurements into the observed map. The agent initially starts with a 2D pose $x_{t=0} = [x, y, \theta]$ describing the coordinates and angle of the agent. Note in the scenario of our research we assume a 2D point holonomic robot with 360° coverage as such θ becomes redundant.
For the occupancy map m containing cells $\mathbf{m_i}$, the goal is to find the posterior occupancy probability of the cell, that is, $p(\mathbf{m}_i|z_{1:t}, x_{1:t})$ given the prior occupancy$p(\mathbf{m}_i)$ and sensor measurements z_t. We use the binary Bayes filter to find the posterior probability and update the occupancy grid map [32].
We transform the occupancy map into it's log-odds form for each of calculation as shown in Equations 2.1. The pseudocode is shown in Algorithm 1

$$
\begin{aligned}
l_{t,i} &= \log \frac{p(\mathbf{m}_i|z_{1:t}, x_{1:t})}{1 - p(\mathbf{m}_i|z_{1:t}, x_{1:t})} \\
l_0 &= \log \frac{p(\mathbf{m}_i)}{1 - p(\mathbf{m}_i)} \\
\text{INVERSESENSORMODEL}(\mathbf{m}_i, x_t, z_t) &= \log \frac{p(\mathbf{m}_i|z_t, x_t)}{1 - p(\mathbf{m}_i|z_t, x_t)}
\end{aligned} \tag{2.1}
$$

The INPERCEPTUALFIELD function uses the Breshnam line algorithm to evaluate whether the cell is touched by any of the sensor beams of the model. If so, then the cell is said to be in the perceptual field of the The INVERSESENSORMODEL function is defined in Section 3.1.1.

Algorithm 1 Occupancy Grid Mapping

Input: $\{l_{t-1,i}\}$ is the occupancy log-odds for cell $\mathbf{m}_i$, x_t is the pose of the agent and z_t is the array of sensor measurements

function OCCUPANCYGRIDMAPPING($\{l_{t-1,i}\}, x_t, z_t$)
  for all cells $\mathbf{m}_i$ **do**
    if INPERCEPTUALFEILD($\mathbf{m}_i, z_t$) **then**
      $l_{t,i} = l_{t-1,i} - l_0$ + INVERSESENSORMODEL($\mathbf{m}_i, x_t, z_t$)
    else
      $l_{t,i} = l_{t-1,i}$
    end if
  end for
  return $l_{t,i}$
end function

2.2 Adaptive Motion Planning

As the autonomous agent encounters a wide range of scenarios in the real world, they are expected to have a constant performance according to some performance metric, usually path quality or search time. Since even general planners Choudhury [6] 2018 proposes that the planning module for the robot must adapt its search strategy to achieve real-time performance across a distribution of planning scenarios. There are two methodologies proposed visualized in Figure 2.1:

1. Black-box adaptive motion planner
2. White-box adaptive motion planner

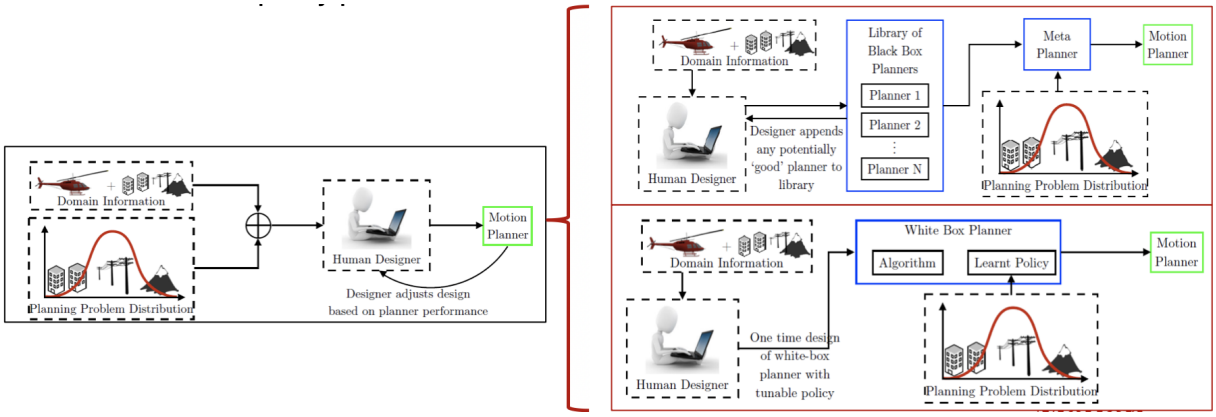

Figure 2.1: The above image visualizes the initial planning problem(Left) and the two proposed systems of blackbox (Top) and whitebox (Bottom) adaptive planners

2.2.1 Black Box Adaptive Planner

Since we know that the planner performance is dependant on the obstacle configuration and we have a series of planners that leverage different assumptions on different environments, the question to ask naturally becomes: can we build a meta-planner that selects a planning strategy from a set of potential strategies, such that we can maximize it's expected performance on the context of the planning problem?

The proposed algorithm by Choudhury, visualized in Figure 2.2, uses list prediction [30] [31] to train a series of predictors over an planning problem distribution to predict seed trajectories for trajectory optimizers, heuristics for search-based planning and planner prediction for 2D and autonomous helicopter planning which are either fixed for all time steps or dynamically changing. The focus of this work is on known maps, goals, and start positions of the agent. Even in the tests of planner prediction with helicopter planning in the case of partially unknown maps, the goal position is known [31]. The body of work extends into exception handling using an online exception planner. That is in case of failure to find a path the agent queries an optimal planner with which has been trained on a multitude of planning problems.

This body of work is closest in idea to the work in this thesis; however, it differs in being a tightly coupled framework with limited modularity, and further, the scope of this thesis specifically focuses on exploration tasks where the goal location is unknown.

2.2.2 White Box Adaptive Planner

Another method to adapt the planning strategy to the environment while conserving the effort to extract context is to adapt the search strategy during a planning cycle by taking a sequential approach. Now the planning problem is seen as a planning policy that maps a sequence of actions and states as visualized in Figure 2.2.

A novel imitation learning-based framework is proposed [7] to train adaptive search heuristics for information and search based planning.

The scope of this thesis, while leveraging ideas about data-driven learning from the proposed novel algorithm for exploration, we focus on providing a modular framework to leverage the qualities of existing planning strategies to improve expected performance during exploration tasks.

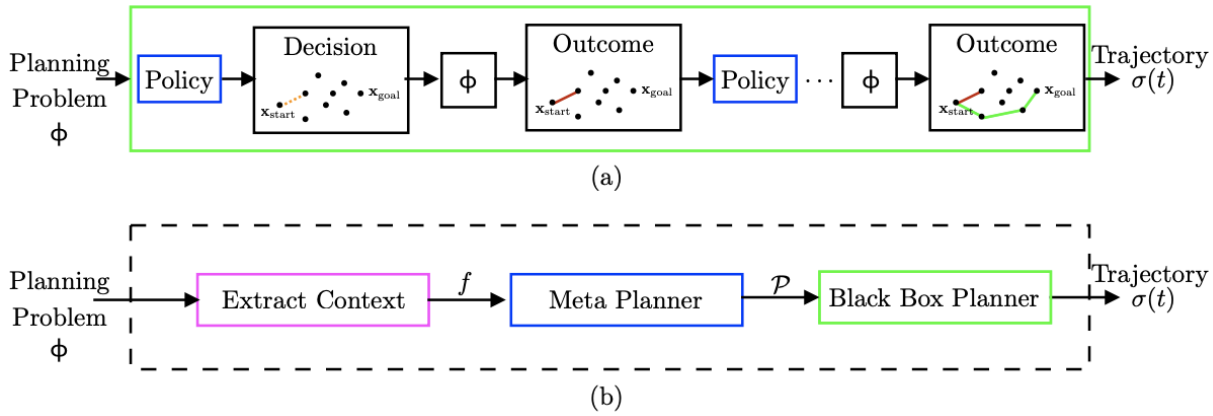

Figure 2.2: (a) In the white-box paradigm, the adaptive aspects happens inside the planner, similar to reinforcement learning. The adaptation is the policy that looks at history of decisions made and outcomes received in order to make the next decision. (b) In the black-box paradigm, the planner expends effort to extract context from the problem to select a black box planner to be executed.

2.3 Map Prediction

As the agent explores the map, searching for the goal(s), the planner set element chosen for path planning has access to the local map explored till that time-step. This means the choice of the planner set element chosen for path planning is biased by the past explored map, over the current map. For example, when the agent passes through a transition between different types of environments, for example and open space and a maze, then until the maze has been explored to the point where the maze portion of the map dominates over the open space the system will choose the planner set element biased to the open space.

To cut down on this transition period where the system works at sub-optimal performance, we aim to use map prediction to predict the future map, based on the local map explored. The paradigm applied is that the usually encountered environments some underlying structure which can be exploited to predict a belief over the unexplored region.

Map prediction literature focus on many applications of this concepts, in the scope of planning and navigation, from leveraging different techniques and assumptions about the environment to machine learning techniques aiming to improve performance. In the scenario of the research, we look at machine learning approaches as they are primed to model the map distribution over the variety of environment. Specifically deep neural network architectures have been used for solving this problem. Caley et al., for example utilizes the CNN architecture to predict exits of a building, given 2D floor plan images, In the scope of path planning, Elhafsi et al. uses deep learning architecture combined with A* frontier exploration for adaptive motion planning. This body of work demonstrates the benefits using deep neural networks for map prediction over a variety of maps. Specifically looking at CNN architectures, we use the U-Net CNN architecture with Map In-Painting losses [27] [18]

2.3.1 CNN Model: U-Net

The U-Net is most suitable CNN architecture as it used for both image classification and segmentation tasks. Segmentation provides the location of elements in an image, in other words, solving a localization tasks of image elements which is a necessary component of map prediction. The U-Net is comprised of a downsampling pathway, of a standard CNN, and an upsampling pathway of deconvolutions that allows the network to perform these tasks. The localization of the elements in particular occurs when a upsampling layer or decoder layer is concatenated with the feature map of the corresponding downsampling layer, improving the output in the next layer. The U-Net is flexible in accepting any input size as the architecture is void of both dense and fully connected layers, thus specifying the segmentation to use the valid portions of the convolution.

2.3.2 InPainting Loss

Image in-painting is the task of filling in holes of images, that is filling in unknown areas based on context of the known area which is analogous to map prediction. As such we leverage the loss used in this method for the map prediction for smoothing out the predicted map into the surrounding. The overall loss function is a weighted average of of multiple loss functions that focus on different metrics for this problem, that are discussed in this section. For the calculation of all these loss functions, an ImageNet trained VGG-16 is used to extract features and project them into a higher feature space

In the equations described below we use the following notations, $\mathbf{I}_{gt}$ is the ground truth image, $\mathbf{I}_{in}$ is the image with the hole that is sent into the network, $\mathbf{I}_{comp}$ [14] is the image that contains the ground truth of the explored area, or known areas in $\mathbf{I}_{in}$, $\mathbf{I}_{out}$ is the predicted map and M is the binary mask sent to the network, that is it contains the sent in image $\mathbf{I}_{in}$ it along with the walls, or occupied space discovered by the exploration. The binary map with the occupied space encoding is used for better understanding for traversable areas, improving map prediction.
The per-pixel losses$\mathcal{L}_{hole}$ and $\mathcal{L}_{hole}$ are defined below. These losses form the metric for understanding the pixel-level difference between the ground truth and the predicted image. Specifically $\mathcal{L}_{\langle \updownarrow]}$ and $\mathcal{L}_{\sqsubseteq \dashv \updownarrow \rangle [}$ are the L1 losses for the hole and non-hole pixels.

$$\mathcal{L}_{hole} = \|(1 - \mathbf{M}) \times (\mathbf{I}_{out} - \mathbf{I}_{gt})\|_1 \tag{2.2}$$

$$\mathcal{L}_{valid} = \|\mathbf{M} \times (\mathbf{I}_{out} - \mathbf{I}_{gt})\|_1 \tag{2.3}$$

Since we also want to be aware of spatial features, $\mathbf{L}_{perceptual}$ and $\mathbf{L}_{style}$ are used.
The perceptual loss $\mathbf{L}_{perceptual}$ on the other hand measures the L1-Norm of the per-pixel difference between ground truth and the image, and in this context it combines the perceptual loss of the predicted image and $\mathbf{I}_{comp}$. This method take a looser approach to the per-pixel difference, avoiding grid-artifacts from recreation. Thus, ideally the weight of this loss should be low.

$$\mathcal{L}_{perceptual} = \sum_{k=0}^{K-1} \left\| \Psi_k^{\mathbf{I}_{out}} - \Psi_k^{\mathbf{I}_{gt}} \right\|_1 + \sum_{k=0}^{K-1} \left\| \Psi_k^{\mathbf{I}_{comp}} - \Psi_k^{\mathbf{I}_{gt}} \right\|_1 \tag{2.4}$$

For computing the other spatial features metric $\mathcal{L}_{style}$, L1-loss is computed after applying an auto correlation, by taking the L1 norm after applying the Gram matrix $C_k \times C_k$, where C_k is the number of channels of the high level features $\Psi(x)_k$, and the normalization factor $K_k = \frac{1}{C_k H_k W_k}$, where H_k, W_k is the image shape, on the k^{th} layer.

$$\mathcal{L}_{style_{out}} = \sum_{k=0}^{K-1} \left\| K_k \left(\left(\Psi_k^{\mathbf{I}_{out}} \right)^\intercal \left(\Psi_k^{\mathbf{I}_{out}} \right) - \left(\Psi_k^{\mathbf{I}_{gt}} \right)^\intercal \left(\Psi_k^{\mathbf{I}_{gt}} \right) \right) \right\|_1 \tag{2.5}$$

$$\mathcal{L}_{style_{comp}} = \sum_{k=0}^{K-1} \left\| K_k \left(\left(\Psi_k^{\mathbf{I}_{comp}} \right)^\intercal \left(\Psi_k^{\mathbf{I}_{comp}} \right) - \left(\Psi_k^{\mathbf{I}_{gt}} \right)^\intercal \left(\Psi_k^{\mathbf{I}_{gt}} \right) \right) \right\|_1 \tag{2.6}$$

The final loss function we need to consider is $\mathcal{L}_{tv}$, which is the total variation loss which is the smoothing penalty. It leads to neighbourhood smoothing by averaging, or taking the L1 norm of the cell and its neighbours in the x, y directions.

$$\mathcal{L}_{tv} = \sum_{(i,j) \in A, (i,j+1) \in A} \left\| \mathbf{I}_{comp}^{i,j+1} - \mathbf{I}_{comp}^{i,j} \right\| + \sum_{(i,j) \in A, (i+1,j) \in A} \left\| \mathbf{I}_{comp}^{i+1,j} - \mathbf{I}_{comp}^{i,j} \right\| \tag{2.7}$$

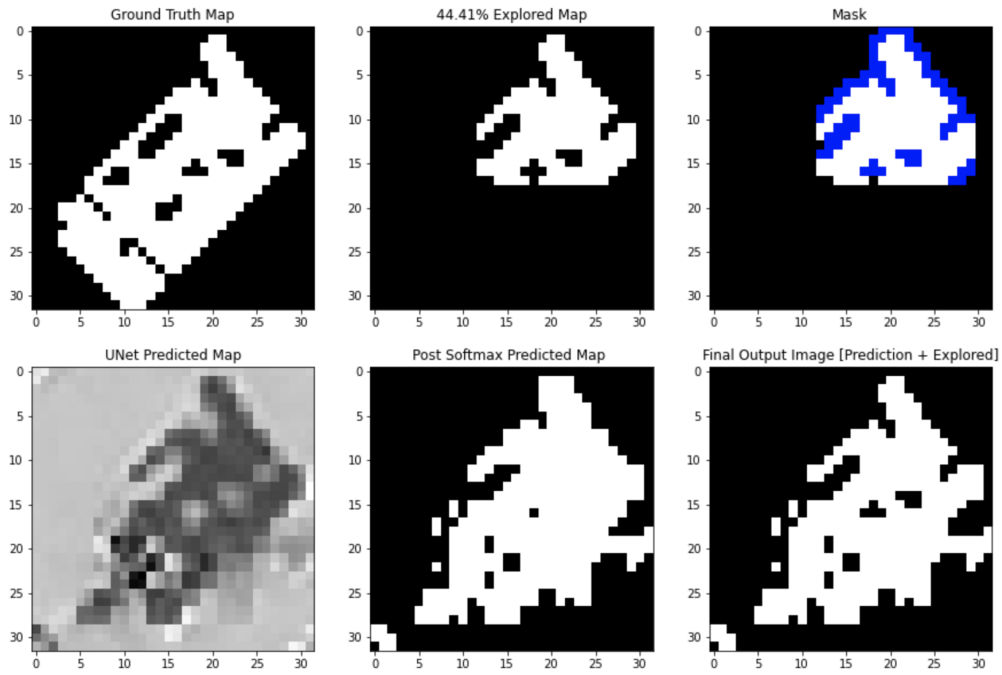

Figure 2.3: The ground truth image $\mathbf{I}_{gt}$ in the top left, the top middle is the image input to the U-Net $\mathbf{I}_{gt}$, on the top right the binary mask $\mathbf{M}$. On the bottom left is the output from the U-Net using soft-max is in the bottom middle and to the bottom right is the predicted output map $\mathbf{I}_{out}$ which is added to the bottom middle

2.4 Goal Estimation

2.4.1 Frontier-Based Exploration

For the agent to find the goal(s), the paradigm is that they must exist in the areas unexplored by the agent. Exploiting this paradigm: the frontier-based exploration algorithm proposed by Yamauchi also guarantees complete exploration and will be used in this work to provide temporary goals to the planner set.
The map representation is a belief space of occupancy, that is, a probability map of occupancy where each cell holds a probability that the corresponding region in space is occupied.

In this paper, a value of 0.5 defines the probability of an unknown or unexplored cell. By that definition, open cells have an occupancy probability of less than 0.5 and occupied have an occupancy probability greater than 0.5.

The algorithm defines frontiers as the transitions between open space and unknown areas and frontier regions as distance-based clusters of the frontiers pruned for computation. The centroids of these regions act as temporary destinations for the agent to increase map knowledge until the complete map or the goal(s) is found. A representation of the terms defined above is shown in Figure 3.11

Chapter 3

Approach

This section describes the map representation used in the for the proposed system and subsequently discusses the mapping strategy and specific sensor models that are used in this research. We discuss the CNN frameworks used for predicting the best performance epsilon in a map and for map prediction. Detailing out the frontier based approach of temporary goal estimation until the true goal(s) is found. Finally we describe the proposed system framework.

3.1 Mapping

As the agent navigates the map with onboard sensing it needs to maintain a map of its surrounding. Since we are using the occupancy grid map representation of the environment, in this section we discuss the mapping methodology for occupancy maps and the simulated sensor model for the agent.

3.1.1 Laser RangeFinder Sensor Model

Laser rangefinders are simple, common and inexpensive sensors, as such modeling them is also simple. For occupancy grid mapping the inverse sensor model [32] is implemented as in the pseudocode in Algorithm 2.

The INVERSESENSORMODEL function given the map cell $\mathbf{m}_i$ in the agent's perceptual view determines the log-odds form of the occupancy probability of the cell, conditioned on the position of the agent and the measurement the measurement of the k^{th} laser beam that passes through it. If the cell is out of range of the laser beam, lies behind the measurement $z_t^k + \frac{\alpha}{2}$, where α is the thickness of the walls or is out of the width of the sensor beam (β) then simply return the prior otherwise update the cell probability. The paradigm used is that if the beam passes through the cell, that is the distance from robot pose to the cell is less than the sensor measurement then the cell occupancy probability is lower, otherwise if the beam endpoint lies with the cell then the cell occupancy probability is higher.

Since, the testing and evaluation is simulated, we simulated the measurements of laser range

Algorithm 2 Inverse Sensor Model

Input: $\mathbf{m}_i$ is the cell position, x_t is the pose of the agent and z_t is the array of sensor measurements

function INVERSESENSORMODEL($\mathbf{m}_i, x_t, z_t$)
    Let x_i, y_ibe the center-of-mass of $\mathbf{m}_i$
    $r \leftarrow \sqrt{(x_i - x)^2 + (y_i - y)^2}$
    $\phi \leftarrow \text{atan2}(y_i - y, x_i - x) - \theta$
    $k \leftarrow \arg\min_j |\phi - \theta_{j,sens}|$
    if $r > \min(z_{max}, z_t^k + \frac{\alpha}{2})$ or $|\phi - \theta_{k,sens}| > \frac{\beta}{2}$ **then**
        return l_0
    end if
    if $z_t^k < z_{max}$ and $|r - z_t^k| < \frac{\alpha}{2}$ **then**
        return l_{occ}
    end if
    if $r \leq z_t^k$ **then**
        return l_{free}
    end if
end function

finder array of 36 lasers uniformly placed, with 360° coverage. The measurements are simulated from the environment by using raycasting to find the occupied cell colliding with the laser beam, which then becomes the measurement for that laser beam. find the collision point of an occupied cell and the laser bean in the environment.
The measurements have white noise sampled from a Gaussian $\mathcal{N}(\tilde{0}, \sigma)$ [13]. Note that as we have a 360° field of view, the angle of the agent is redundant to occupancy grid mapping algorithm.

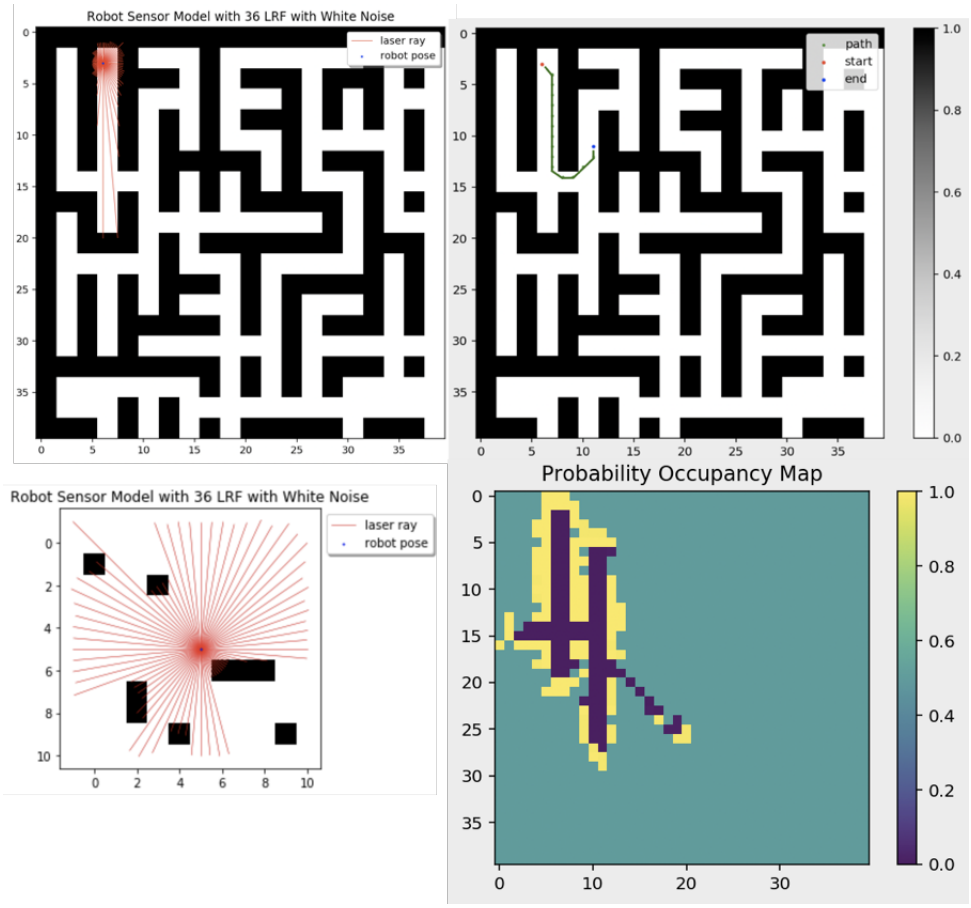

Figure 3.1: (Left) The simulated laser rangefinder array is shown in the a random obstacle and maze map. (Top Right) The groundtruth occupancy map and the robot path (green) is shown, note the colorbar shows black for occupied and white for free cells. (Bottom Right) The cumulative occupancy mapping of the robot using the sensor model along the path is shown. The colorbar shows the associated color to the probability of occupancy, for example yellow is occupied cells, $p(\mathbf{m}_i)$= 1, blue is unknown $p(\mathbf{m}_i)$= 0.5, and purple is free space, $p(\mathbf{m}_i)$= 0.

3.2 Environment Datasets

In this section we discuss the 2D environment datasets used to form the planning problem distribution $P(\Phi)$.
The maps are of eclectic shapes ranging from (32x32) to (1024x1024). The environments displayed below are full maps for each dataset however since the planner only sees a local map of approximately (32x32) we split these maps into local areas of (32x32) for evaluation.

3.2.1 Map Generation

Mazes are a standard testing environment for path planning problems, as such we use them as one of the environment datasets for evaluating our system. The maze generation algorithms used

graph-based were: Eller's, Wilson's, Kruskal's, Randomized Prim's and Aldous-Broder's maze generation algorithms [23]
We use a number of maze generation algorithms to downweigh any inherent bias of the algorithms to prefer to certain elements of the planner set. The above algorithms generate 'perfect mazes', that is any point in the maze can be reached by any other point which makes the analysis in Section 3.4 easier as there are no absolutely unreachable targets in these maps.

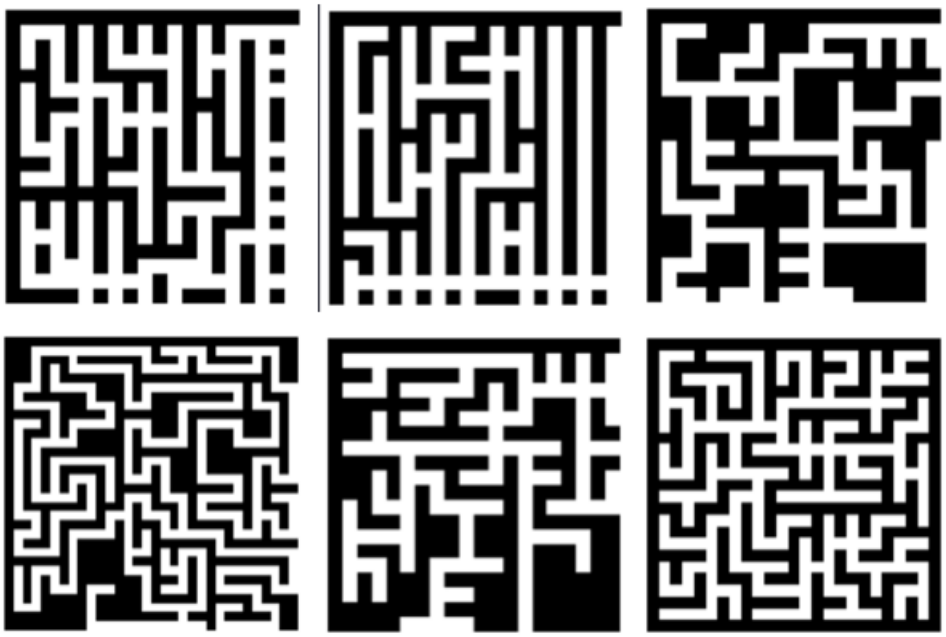

Figure 3.2: (Clockwise) The maze maps shown above showcase different maze generation algorithms with different hyperparameters

3.2.2 PathFinding Benchmark Datasets

While mazes are a good benchmark for testing path planning algorithms, they are not representative of map environments potentially experienced by field robots. To increase the variety of maps in our database, we look at Nathan Sturtevant's Moving AI Lab's 2D PathFinding Benchmark maps [29]. These are generated from a variety of sources, such as multiple video game dungeon maps as well as real city street maps, as shown in Figure 3.3. The dungeon maps resemble cave or tunnels environments, depending on the video game, that a field robot could experience as well as real-world street maps, thus forming a dataset similar to the real-world for evaluating the performance of the proposed system.
From the raw map file format the environments are converted into binary maps where zeros represent free cells and ones represent occupied cells.

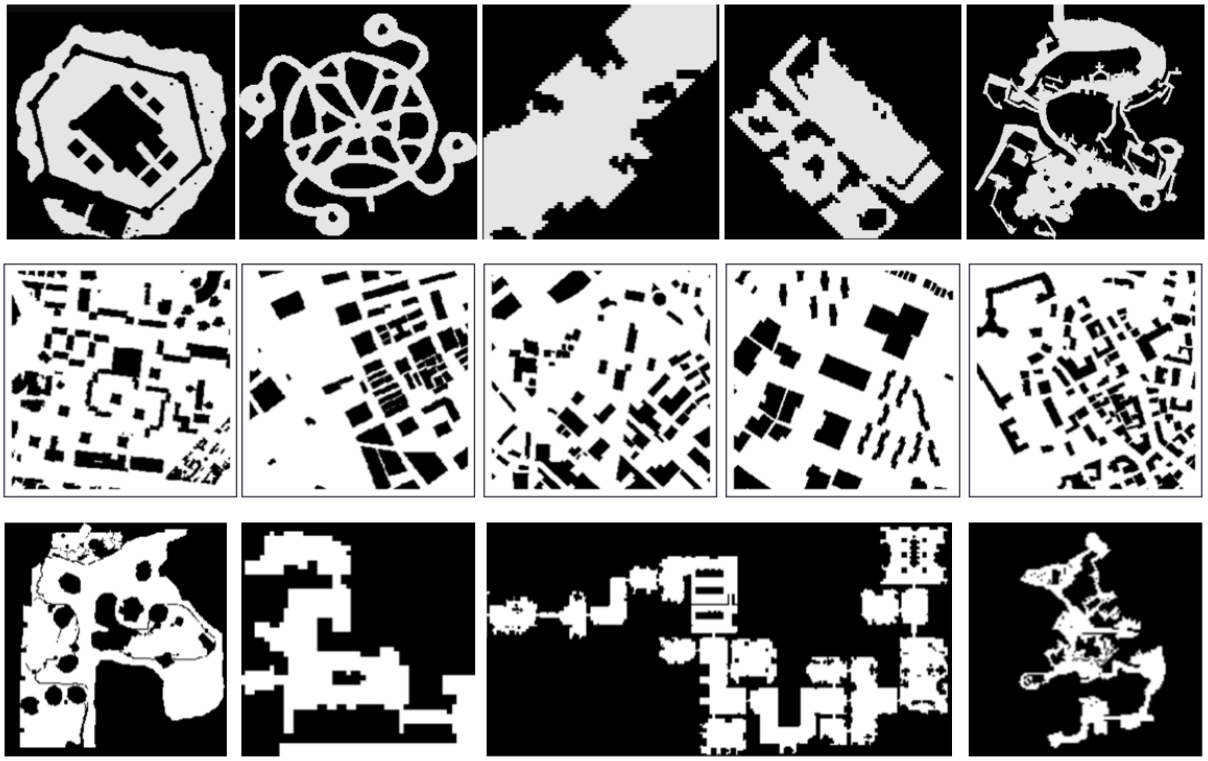

Figure 3.3: (Top to Bottom) The top and bottom row showcases dungeons maps from Baldur's Gate and Dragon Age: Origins video games respectively, the middle row showcases street maps from around the world

3.2.3 Greedy Agnostic Maps

Generally heuristic-based planner, for example A* with an euclidean heuristic behaves like a 'best first' or greedy planner when inflation factor $\epsilon > 1$. 'Best First' planner tend to have low performance in case of obstacle placement that causes them to plan suboptimal paths to navigate around the obstacle. Such maps are called greedy agnostic maps, and we integrate the dataset of such maps, described in [22], into the database of maps being used.

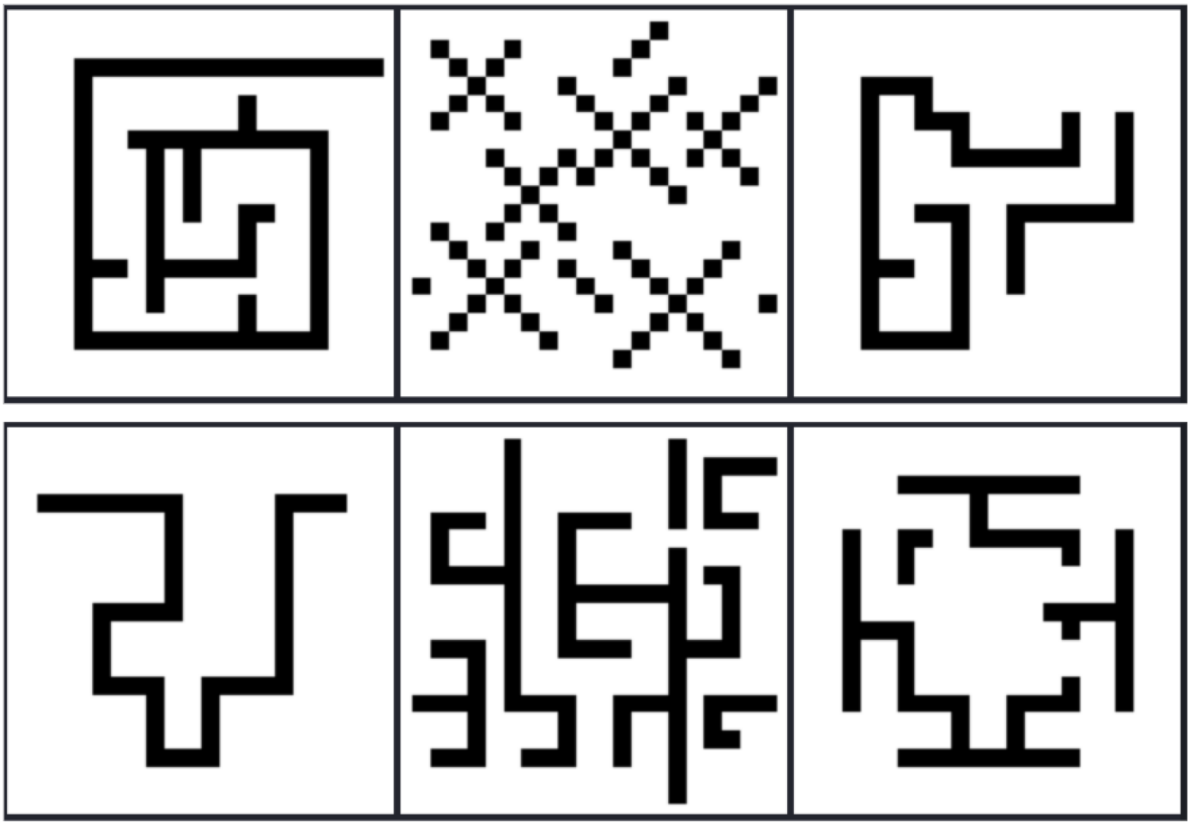

Figure 3.4: Subset of maps contained in the Greedy Agnostic map dataset.

3.3 System Design Overview

We consider a single agent exploring an initially unknown static 2D map M using the laser ranger-finder array system detailed in Section 3.1.1. The initial assumptions of this framework are:

1. The shape, or size of the environment $(m \times n)$ is available to the agent
2. The agent's starting position r_p is in a free cell whose position is known

The framework pseudocode and visualization are shown below in Algorithm 3 and Figure 3.5 respectively. It is comprised of multiple modular components following these steps: given the starting agent pose r_p, while the true goal(s) g_{true} are unknown:

1. The agent uses the SENSORMODEL of the onboard laser rangefinder sensing array to observe the true environment M_{true} and simulate the sensor measurements z_T from the current pose r_p.
2. Update the observed map M_{obs}, with the sensor measurements z_t using UPDATEMAP

Algorithm 3 System Framework

function SYSTEM($r_p, (m, n)$)
  $M_{obs} \leftarrow [0.5]_{m \times n}$
  while $True$ **do**
    $z_t \leftarrow$ SENSORMODEL(r_p)
    $M_{obs} \leftarrow$ UPDATEMAP(M_{obs}, z_t)
    if FOUNDGOAL(M_{obs}) **then**
      break
    end if
    $M_{pred} \leftarrow$ PREDICTMAP(M_{obs})
    $g_{front} \leftarrow$ YAMAUCHI(M_{obs})
    $p \leftarrow$ PREDICTPLANNERELEMENT(M_{pred})
    $r_{p*} \leftarrow p(r_p, g_{front})$
    $r_p \leftarrow r_{p*}$
  end while
end function

which is wrapper for Algorithm 1 .

3. The agent uses Algorithm 4 to get a temporary goal g_{front} of the nearest frontier region centroid from YAMAUCHI.
4. The agent based on the local explored map, uses PREDICTMAP to predict a window of the unexplored area, which forms M_{pred}
5. Using the predicted map PREDICTPLANNERELEMENT predict the best planner set element $p \in P_N$, where $P_N = \{p_1, p_2 ... p_N\}$ is the planner set.
6. Using the planner set element p for navigation to g_{front}, the agent arrives at the next pose r_{p*} and repeats from (1).

Figure 3.5, shown below is a visualization of this framework.

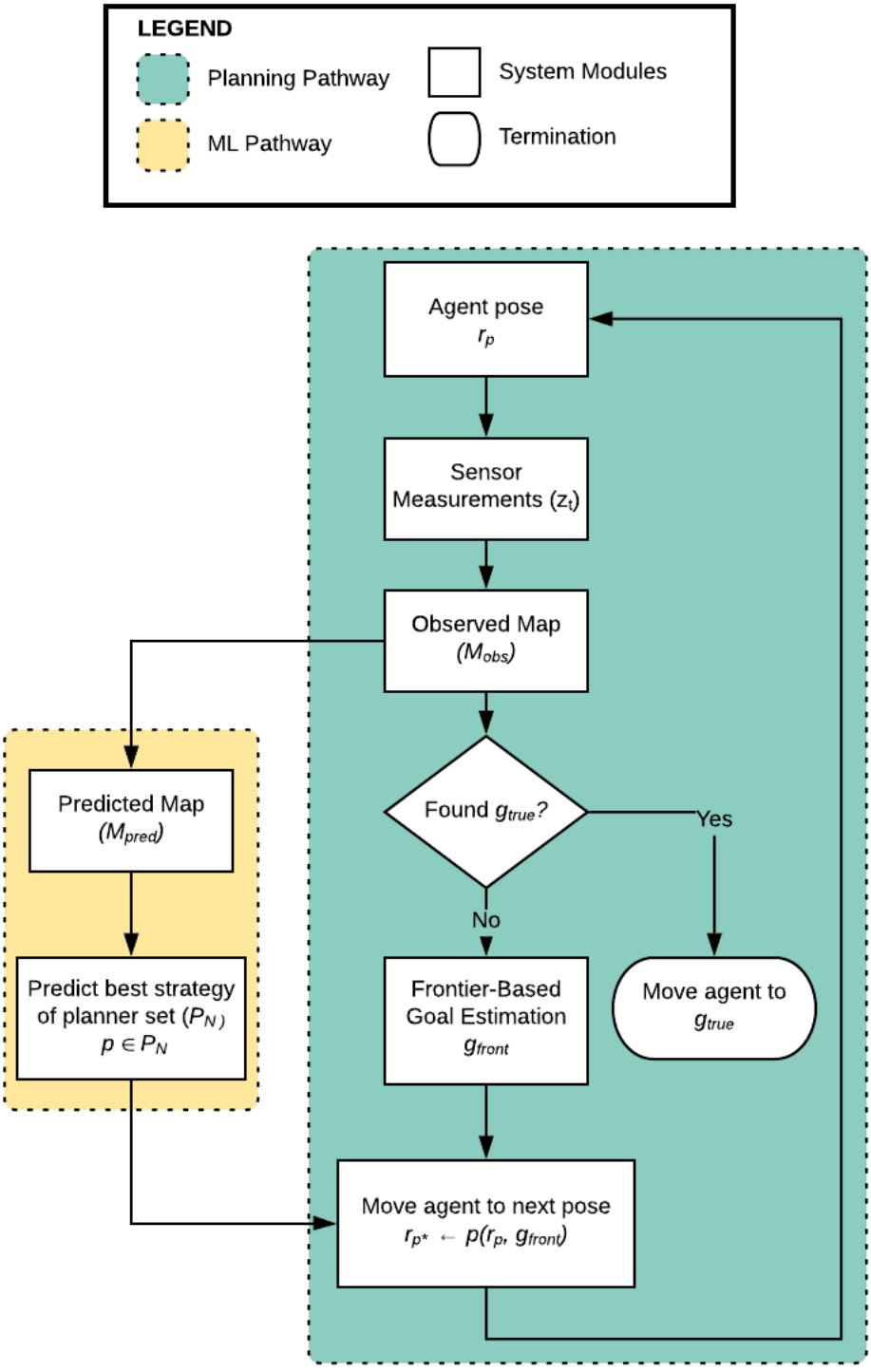

Figure 3.5: Visualization of the system framework

3.4 Meta-Planner Strategy Prediction

To utilize prior performance information of planner strategies over the distribution of planning problems, we need a learning model to learn the association between the best performing planning strategy and the respective context. In this scenario, the context is the local map sampled from the planning problem distribution. Note that the local map refers to a (32x32) window of the observed map around the agent position extended using map prediction. This task is equivalent to image classification where the image is the binary map of the environment and the label is the planner set element with best performance.

To learn the association between the map and the planner set element with the best performance in it, we decided to use a CNN architecture, due to high performance in similar image classification tasks, high scalability and does not rely on preprocessing of images for feature selection.

In the context of this thesis, the map is sent as an image to the CNN architecture; the corresponding label is the planner set element with the best performance in that map.
To associate the map to label, we take a Monte Carlo Expectation approach of generating random start and destinations for the agent to traverse through the map, and evaluating the performance of all the planner strategies in planner set, that is $p_i \in P_n$. Using the performance criterion on the performance expectation on the environment for each planner, the best planning strategy p_{best} is chosen as the label for the respective environment. Figure 3.6 provides a visualization of this process.

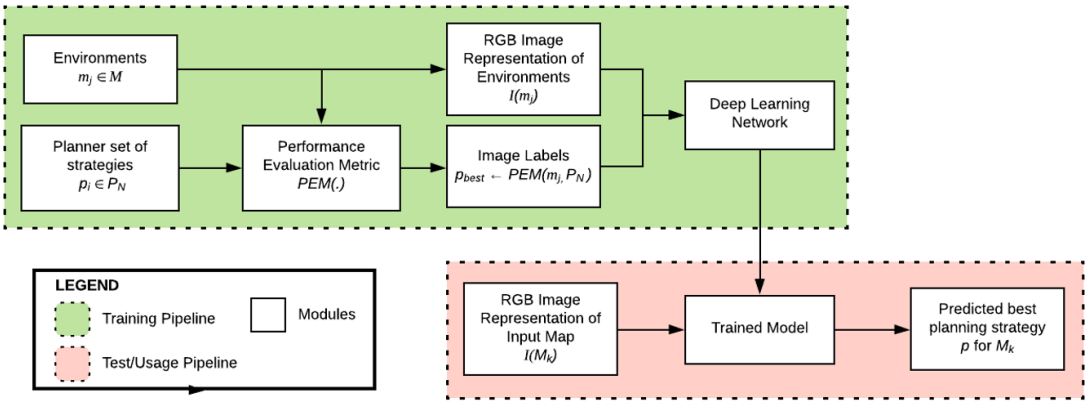

Figure 3.6: Model Training and Usage Pipeline

3.4.1 CNN Model: VGG16

The CNN architecture used in this scenario is the VGG16 CNN architecture, in particular for its high performance in image classification and its usage for high-level feature representation for maps in the datasets.

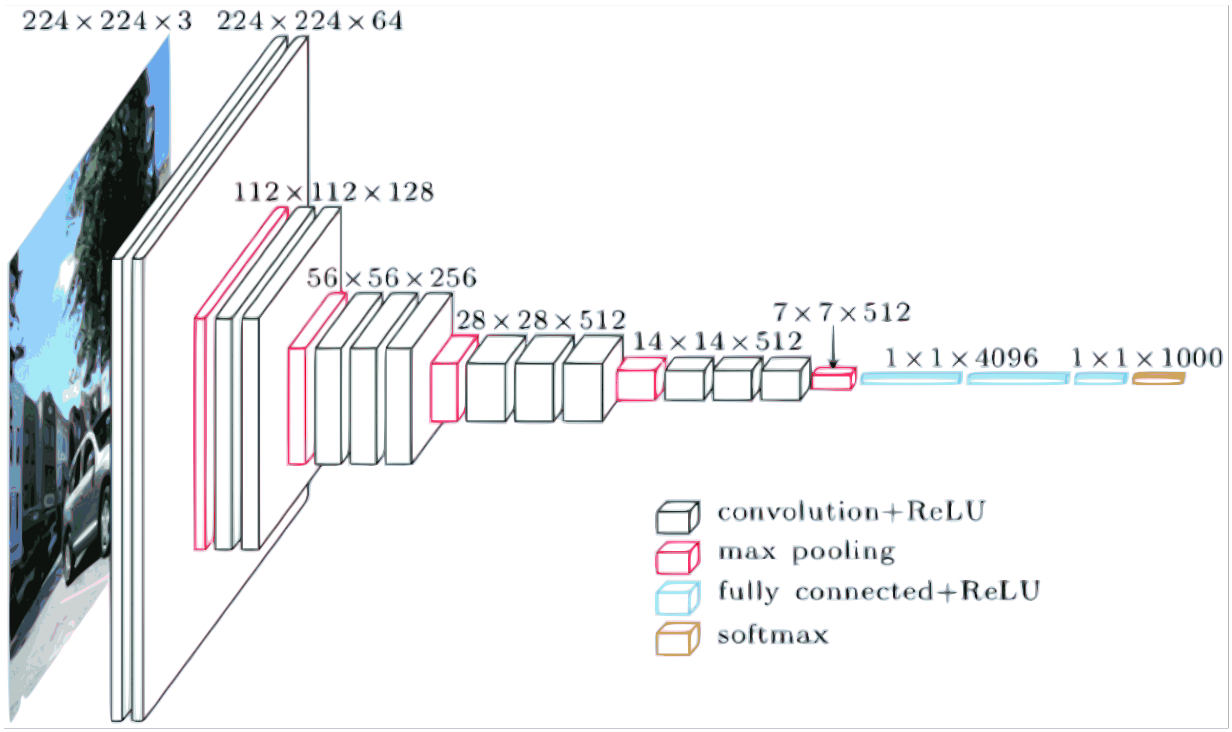

Figure 3.7: VGG16 architecture for an image size of 224x224x3 pixels. The encoder (left) aims to capture spatial context and the decoder (right) accurately localise these features

The VGG16 architecture is comprised solely of a downsampling pathway (encoder) to extract the image features. The initial convolution layers use 3x3 kernel convolutions with a stride of 1, followed by 2x2 kernel max pooling layers with a stride of 2. Following the initial stack of convolutional layers, which are adjusted to the specific image size, are three Fully-Connected (FC) layers: the first two have 4096 channels each and the third performs 1000- way ILSVRC classification. This is the standard format of the VGG16 network, and for our purposes we change the last layer to the size of the number of elements in our planner set.
All the hidden layers are composed with Rectified Linear Units (ReLU). The function, $F = \max\{0, x\}$ introduces non-linearity into the model reducing vanishing gradients and has superior computation speed compared to other activation functions such as the sigmoid or hyperbolic tangent function.
The dataset sent to this network is composed of the local map of (32x32) converted to RGB images of size (224x224x3) with the soft labels of the planner set element. We use soft labels as the best performance planner set element is determined as an expected value that has variance across the other elements as well. Since we use soft labels we use softmax loss or cross-categorical

entropy loss to predict a probability over the classes.

Figure 3.8: Input maps of size (32x32) resized and colored to (224x224x3) for the VGG16 network. The yellow color represents open or traversable space and black represents occupied space

3.5 Map Prediction

Map prediction utilizes a combination of the U-Net CNN architecture and image inpainting, that given a percentage of the map that is known, we can predict the unknown map. The U-Net CNN architecture used assumes a pre-defined image size of $H \times W$ pixels or cells, where $H = W = 32$, forming the window of the area of map-prediction. We choose this image size due to the limitations of the onboard sensing and having a sizeable explored area for the U-Net to confidently predict the remaining area without biasing towards the previous type of environment the agent was in. Note that this thesis uses map prediction to extend the perception of the robot in terms of the observed local map and predict the unknown regions in the window. Since the global map can have varying complexity, and the agent operates in the local map it doesn't effectively utilize the map prediction to
The network predicts an occupancy probability for each cell in the map based on training which is incorporated with the explored map for as input for further modules in the system.

3.5.1 CNN Model: UNet

The multilayer CNN used for map prediction specifically uses the U-Net architecture, based on [27], chosen for high levels of performance in generalizability for image processing contexts and it's speed.

The U-Net architecture is a special form of CNN as in addition to the encoder path, or the contracting path of downsampling layers, on the left it is comprised of the decoder path or the expansive path of upsampling layers, on the right in Figure 3.10.
The encoder path follows a traditional CNN architecture that repeatedly applies 3x3 kernel convolutions followed by a Rectified Linear Unit (ReLU) activation function. The ReLU function,

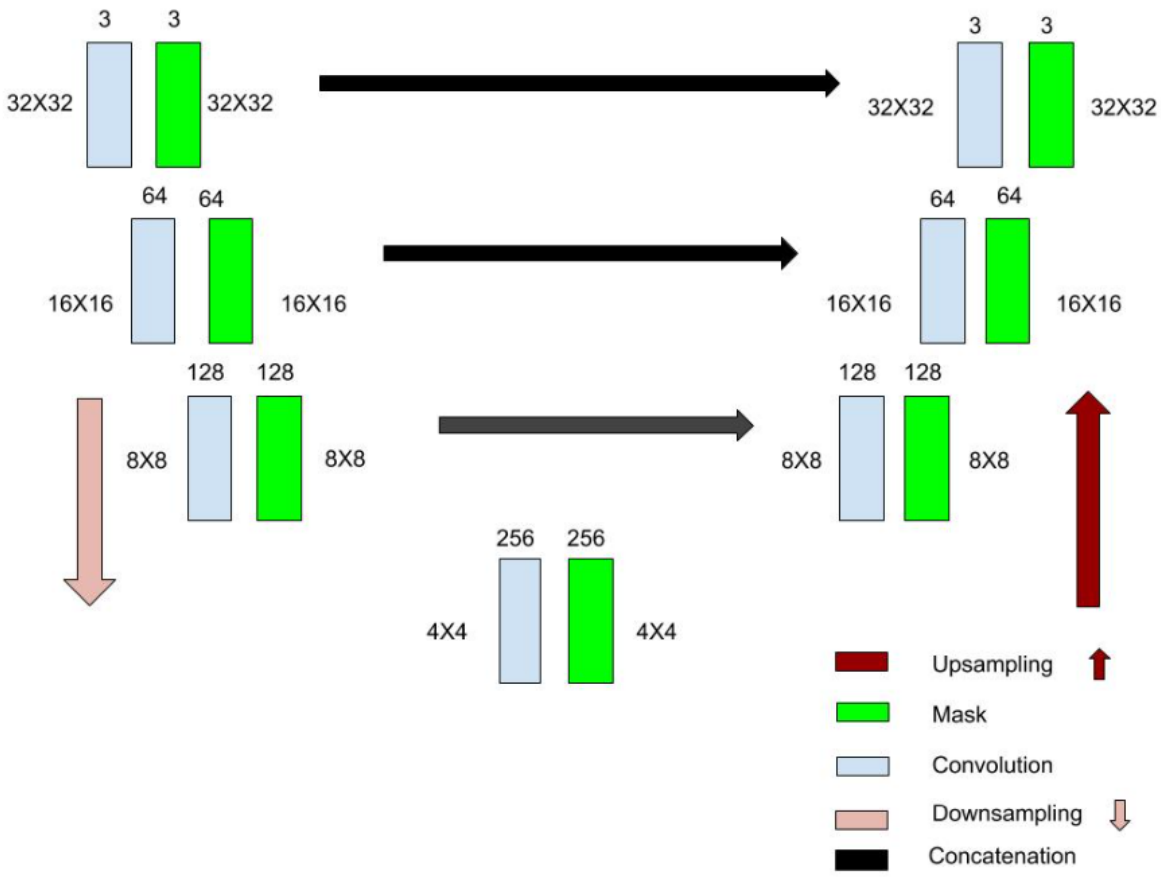

Figure 3.9: U-Net architecture for an image size of 3232 pixels. The encoder (left) aims to capture spatial context and the decoder (right) accurately localizes these features

$F = \max\{0, x\}$ introduces non-linearity into the model reducing vanishing gradients and has superior computation speed compared to other activation functions such as the sigmoid or hyperbolic tangent function. After passing through the ReLU, a 2x2 kernel max pooling operation with a stride of 2 is applied, doubling the number of features at each layer.
The first layer of the encoder is unique as the 3-channel RGB image is mapped to 64 feature channels.
The U-Net decoder, composed of upsampling layers, captures the locations of the features from the encoder. In addition to the upsampling at each decoder layer using 2x2 transposed convolutions, the tensor is concatenated to the corresponding cropped feature map from the encoder path for localizing the features in the spatial context followed by a 3x3 convolution and ReLU.

Dataset Generation

The dataset shape is $(N, 3, x, y)$, where N is the number of samples in the dataset, $x \times y$ is the size of the image. For each sample, three input images: Ground Truth, Image and Mask as in Section 2.3.2.
The Ground Truth is the true map of the environment with zeros representing free cells and ones representing occupied cells, and is used for calculating the losses. The Image ($\mathbf{I}_{in}$) contains the explored free cells encoded as zeros and the unexplored cells as ones. For training, we randomly chose a ratio between 0.1 and 0.7. The Mask (M) is equivalent to the Image with the addition of using ones to encode the walls that have been discovered by the exploration. We split the dataset for training, testing and validation in the ratios of 0.8, 0.1, 0.1 respectively.

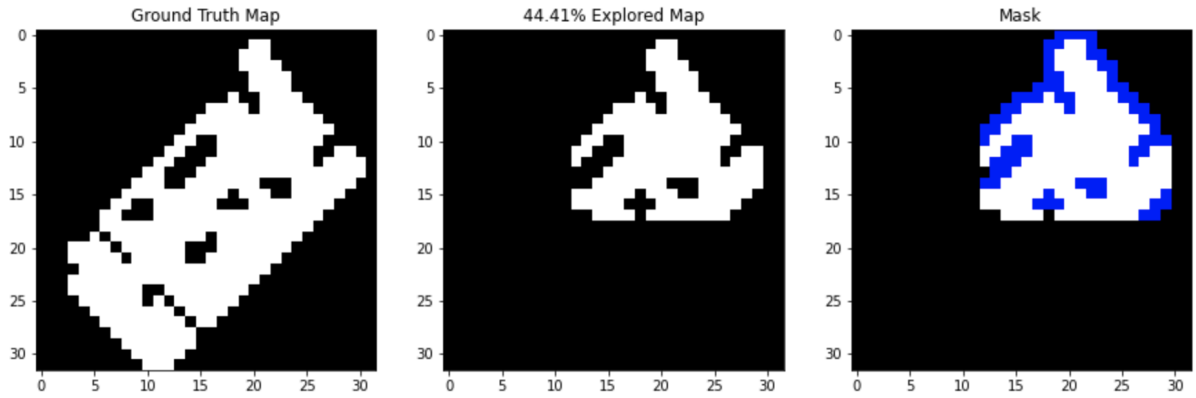

Figure 3.10: (Left to right) Ground truth map, Image and Mask. The Image contains the portion of the map explored and the Mask encodes the known walls in the explored region

3.5.2 InPainting Loss

According to the formulae and the loss terms discussed in Section 2.3.2 we calculate the defined losses, as we assumed the image, or map of size $H_p \times W_p = H \times W$ to be predicted. Specifically for the scenario proposed of this thesis the $\mathcal{L}_{total}$ is defined as the weighted average of the other spatial losses given in Equation 3.1 with weights a, b, c, d, e.

$$\mathcal{L}_{total} = a\mathcal{L}_{valid} + b\mathcal{L}_{hole} + c\mathcal{L}_{perceptual} + d\left(\mathcal{L}_{style_{out}} + \mathcal{L}_{style_{comp}}\right) + e\mathcal{L}_{tv} \tag{3.1}$$

3.6 Frontier-Based Goal Estimation

This research uses the Yamauchi frontier-based exploration for a single agent to estimate temporary goal(s) while the true goal(s) are unknown as mentioned in Section 2.4.1. In our implementation of the algorithm, whose outline is shown in Algorithm 4, we use the A* algorithm to find the closest frontier region centroid and the chosen planner set element to navigate to this

frontier. The ISFRONTIERCELL function determines that if the cell is: unvisited by the agent and is an open(free) cell with at least one adjacent unknown cell, then it is a frontier cell. The CLUSTER function takes the frontier and computes the centroids of frontier regions, clustered using connected-component labelling. The DIST function uses the A* algorithm to find the distance between the current robot position and the frontier region centroids. The closest cluster centroid f now becomes the temporary goal (g_{est}) for the planner set element to navigate the agent to while searching for the true goal (g_{true}). The algorithm terminates once there are no frontier found or when all the cells have been visited.

Algorithm 4 Yamauchi's Frontier Based Exploration

```
1:  function YAMAUCHI(M)
2:      f ← [0, 0]                                          ▷ Initialization
3:      frontiers ← ∅
4:      for all cells c in M do
5:          if ISFRONTIERCELL(c) then
6:              frontiers.append(c)
7:          end if
8:      end for
9:      clusterCentroids ← CLUSTER(frontiers)
10:     f ← min (DIST(clusterCentroids))
11: return f
12: end function
```

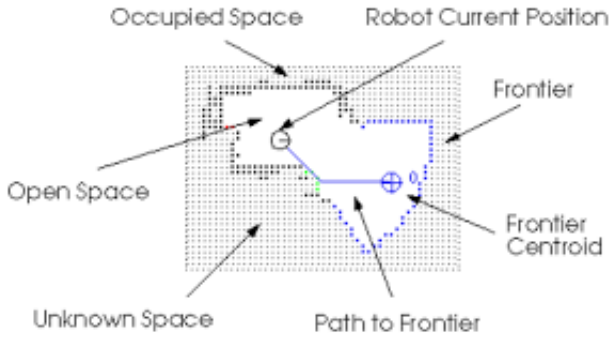

Figure 3.11: The above image visualizes the concepts of open(free space), occupied space, unknown areas and frontiers

Chapter 4

Experiments and Results

To demonstrate the effectiveness and trade-offs of the proposed system design, we developed simulation environments for 2-D grid worlds to test and evaluate the performance. This section demonstrates the results of testing the system on the datasets described in 3.2

4.1 Simulation

The simulation was run on an Intel Core i7-8750H CPU. The evaluations presented in this section are focused on adapting the hyperparameters of the given planning method to the environment based on the defined performance metric.

4.1.1 Planning Problems Distribution

Since we are using a series of already known inputs maps for the distribution we need a method to classify them. In this thesis we use two criteria:

1. **Global Connectivity**: The global connectivity of the map is defined based on it's graph representation connectivity. That is, it is a normalized measure of the valid edges of the graph representation of the map.
2. **Local Distance Field**: The local connectivity is defined as local distance field, that is a averaged distance between each point on the user-defined sized local lattice between start and goal locations on the map. A user-defined sized local lattice is formed by using the straight line joining the start and goal positions and considering a fixed area around the line.

These two metrics encapsulate the planning problems encountered in each map and obstacle configuration, estimating the obstacle configuration and density. An analysis of these metrics are shown in Figure 4.1.

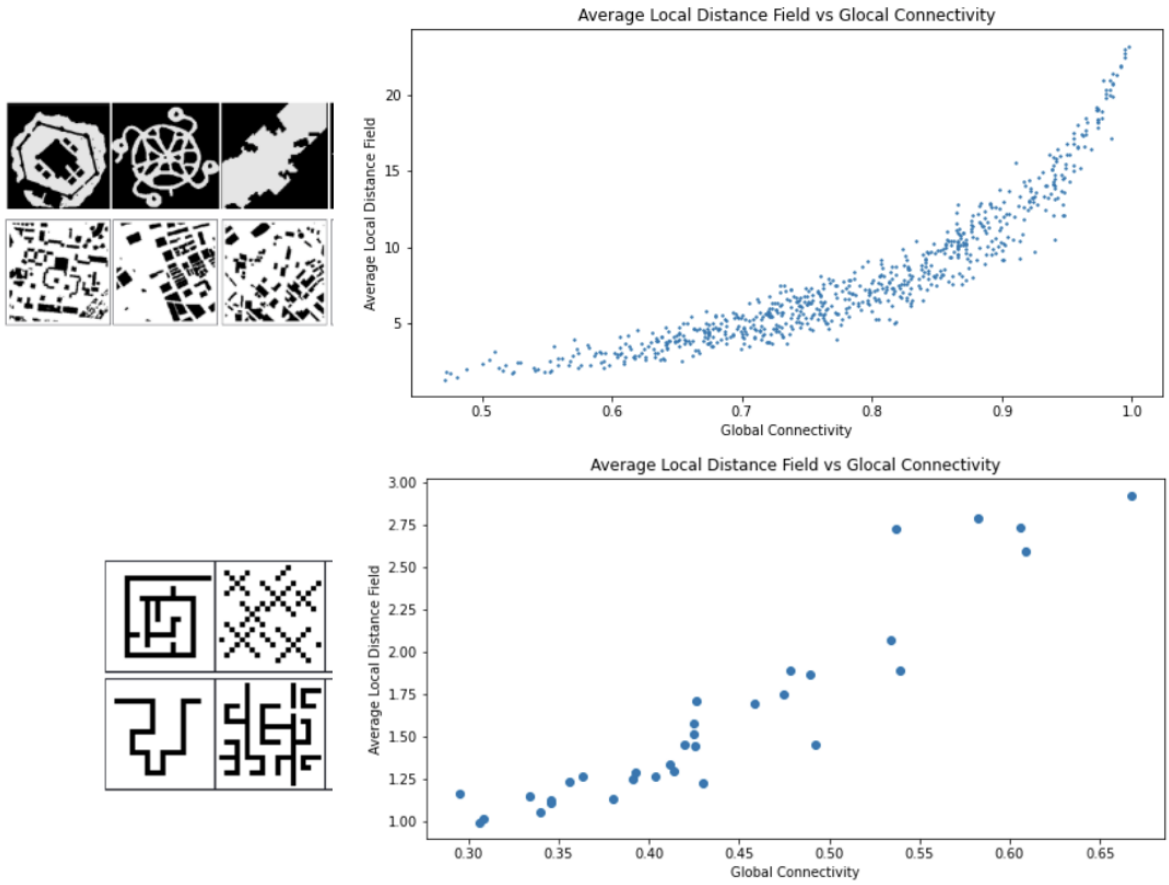

Figure 4.1: (Top) We have the metric analysis on the Video game maps, as sample of which is shown to the left, and (Bottom) of the greedy agnostic maps, as sample of which is shown to the left.

4.2 Search Based Planning Evaluation: Weighted A*

We test the system on a combination of different maps from the datasets defined in Section 3.2. In the context of this thesis, we work with A*, controlled by the heuristic inflation factor, ϵ. Pairs of starting position of the agent and the true goal (unknown to the agent) are chosen randomly on the map to evaluate the expected performance of the adaptive heuristic. For each of these start and goal positions, the proposed planner set is a set of n inflation factor values (m values sampled from $\epsilon < 1$ and $n - m$ values sampled from $\epsilon > 1$) for the A* Manhattan distance heuristic on these 8-connected grids. From prior knowledge, we are aware that as $\epsilon < 1$, the search behaves more like Dijkstra's shortest path algorithm and as $\epsilon > 1$, the search behaves more like a Greedy best-first search.

4.2.1 Performance Criteria

The performance criteria defined for the task, is the search time t to find a path σ, normalized by the path quality of the found path from start to goal. The path quality is assessed as the path cost (c) of the trajectory calculated from the cost map of the maze sent to the planner. The cost map encodes the obstacle configuration with a high cost to ensure they are untraversable, and the neighbouring cells are encoded with a clearance cost. An example is show in Figure 4.2.

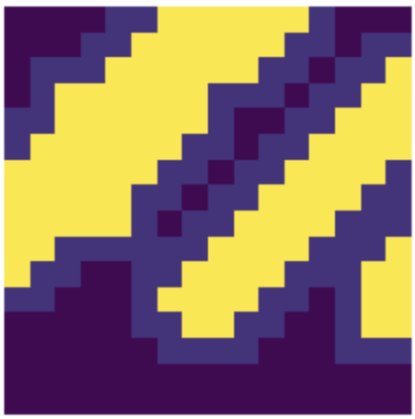

Figure 4.2: The yellow area represents the obstacle configuration with a high cost, the light purple represents the clearance cost cells, and the dark purple represents the free space

The performance score for a strategy p, in a map is calculated as the expectation of the criteria applied on the strategy over a set of N start and goal pairs in the map as shown in Equation 4.1

$$\text{score}(p, M) = \mathbb{E}\left[\frac{t(\sigma_i)}{c(\sigma_i, M)}\Bigg|_{i=0}^{N}\right] \tag{4.1}$$

Note, that by this definition the score is inversely proportional to the performance.

4.2.2 System Evaluation

For expected performance evaluation we take three maps, of different combinations of the environments existing in the database defined in Section 3.2 as shown in Figure. 4.3 and samples using the planning problem distribution classification defined in Section 4.1.1. The planning strategies are the values for the inflation factor ϵ. We create this set of strategies by sampling 10 values in the interval $[0, 1)$ for $\epsilon < 1$ and 10 values in the interval $[20, 25]$ for $\epsilon > 1$.
The evaluation is carried out by a uniform random selection of 100 start and goal pairs in the environment. Over these pairs, the performance of the adaptive system on the heuristic hyperparameter is calculated by running the exploration task. Then for comparison of performance, the exploration task is run over those same start and goal positions, however now for each start and goal a heuristic inflation factor is randomly sampled from the set and kept static throughout the procedure.

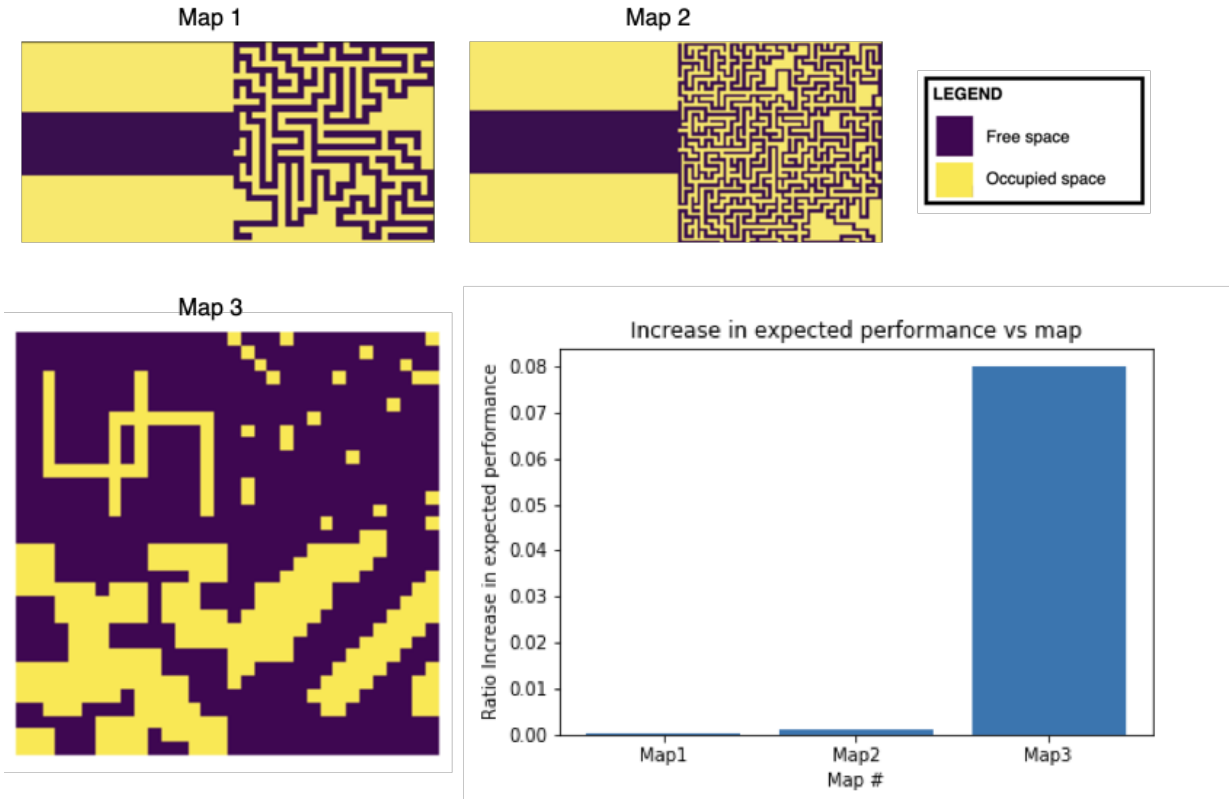

Figure 4.3: On the left, from top to bottom are sets of combined maps that this proposed adaptive system was tested upon, on the bottom right are the results of the increase in expected performance on these maps calculated from Table 4.1. In the maps, the yellow areas represent occupied space and purple areas represent unoccupied(free) space.

Environment	Average Performance Scores ($\times 10^{-3}$)		Avg. Perf. Increase Ratio
	Random Static Planner	Adaptive Planning System	
Map 1	39.705	39.685	0.0001
Map 2	54.970	54.915	0.005
Map 3	49.764	45.833	0.08

Table 4.1: Table of averaged performance scores on the environments, or planning problem distributions, in Figure 4.3 over 100 start-goal point pairs. It also contains ratio of performance increase when using the proposed adaptive system over a random-static planner.

The calculated expected performance of the adaptive system is compared to that of the randomly chosen static planner by the percentage increase of the expected performance of the adaptive system over the random-static system shown in Figure 4.3 and the data in Tabel 4.1. We can see that an adaptive heuristic hyper-parameter has a higher expected performance on adap-

tive heuristics in the context of the planning problem distribution over the planning benchmark, randomized and greedy agnostic datasets compared to the maze dataset. This could be due to the high level of confined structure of the maze which constrains the perception of the robot, and even with the addition of map prediction to extend the local perception of the agent the adaptive system's expected performance is almost comparable to using a random-static heuristic hyperparameter. However, as we relax this tight constraint in Map 3 of the results, we can see the expected performance of the system is much better than using a random-static heuristic hyperparameter. Also note that Map 3 has a higher number of different environments sampled from the planning problem distribution, compared to the maze datasets.

4.3 CNN Models

4.3.1 Meta-Planner Strategy Prediction

The VGG16 architecture is fed with training images as detailed in Section 3.4 and soft labels of the classes, that is over the set of planning strategies, which in this case refer to the inflation values. We leverage transfer learning to pre-train the VGG16 model with ImageNet weights, similar to the strategy for the feature extraction in map prediction, so that training on our dataset of 300 images with respective labels could be computationally and timewise efficient.

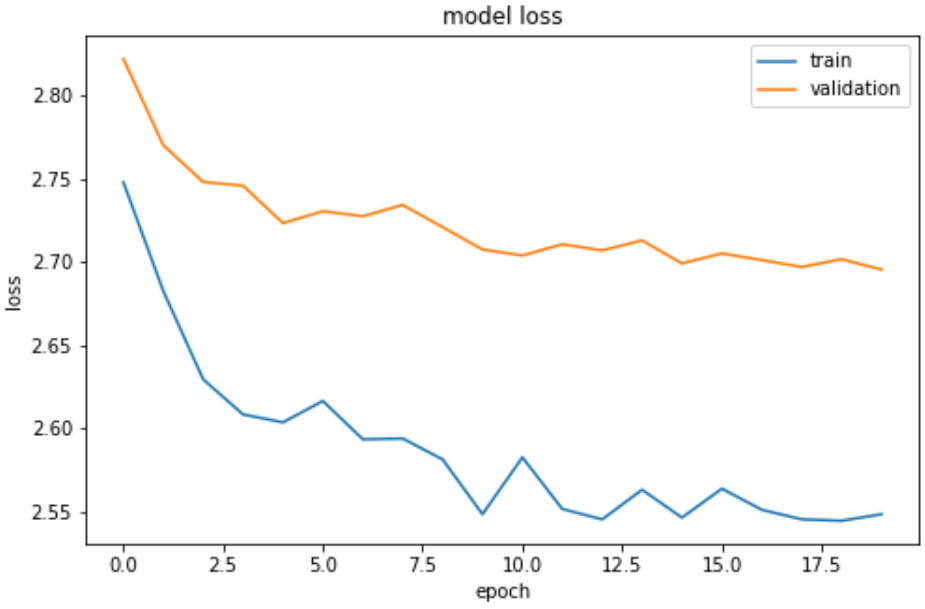

Figure 4.4: Visualization of Meta-Planner Strategy Prediction loss

The model after training and validation has an accuracy of 72% and 61% respectively, however,

looking at the training loss and the short amount of epochs it takes to the plateau it could be a case of over-fitting to the dataset. Thus, while this predictor will work well on the type of distributions it has already seen, it is likely to fail on distributions it hasn't seen before.
As a part of future work, one solution to this problem is to use the planning problem distribution classification metrics as features and replace the VGG16 model with the Random Forests model for the classification of those features with the labels of the planning strategies.

4.3.2 Map Prediction

The UNet Architecture is fed with training images as detailed in Section 3.5 of size (32x32). The network is trained for 1.8×10^6. After 1.02×10^6 iterations as seen in Figure 4.5 given a percentage explored area over varied types and distributions of maps the map prediction network is able to predict the complexity of the unknown map to an extent over two different distributions. Since the goal for the map prediction is to inform the meta-planning strategy selection by extending the boundaries of the explored local region and filling in the unknown spaces, this level of prediction over the entire database serves the purpose of the map prediction module.

Training and Validation Losses

In this section we look at training results in Figure 4.6 which shows the losses of training the map prediction network over a combined dataset from Section 3.5 for 1.8×10^6 training epochs. This includes the individual and combined inpainting loss. The weights for the combined inpainting loss equation as defined in Section 3.5.2, are calculated by a hyperparameter search on 100 validation images as per [18] [27]. They are inflated to prevent overfitting, and tuned on the considered environment database. The final values are as below:

$$a = 120, b = 20, c = 0.05, d = 120, e = 2 \tag{4.2}$$

For the dataset we can see that the losses except $\mathcal{L}_{hole}$ start to plateau around 600,000 training iterations, and the validation loss doesn't differ too much. However there is a large amount of variance per-pixel of the map since the dataset was generated using several different algorithms and datasets and is of different topologies and clutteredness, as such $\mathcal{L}_{hole}$ and $\mathcal{L}_{valid}$'s behaviour and the noise in the other losses is expected. After multiple tests the weights referenced in Section 3.5.2 give the best losses.
Since there is high variability in the maps of the dataset, there is a large enough amount of variance per-pixel that it is reflected in the L1 norm, as such $\mathcal{L}_{hole}$ and $\mathcal{L}_{perceptual}$'s behaviour is expected. Since we are not expecting a perfect map from the prediction, but rather an understanding of the complexity or clutteredness of the environment, these results fit the requirements.

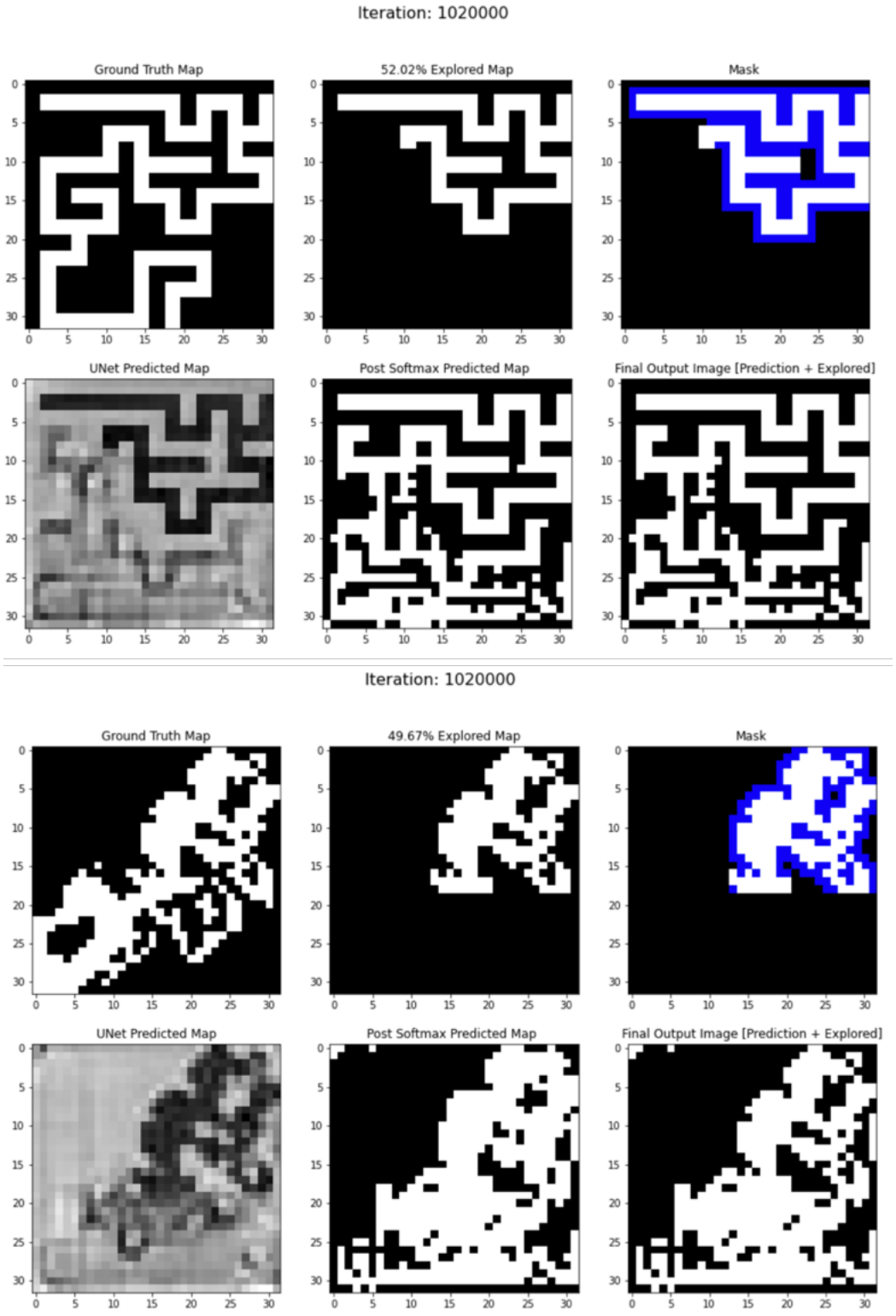

Figure 4.5: Visualization of Map Prediction results after training on combined datasets

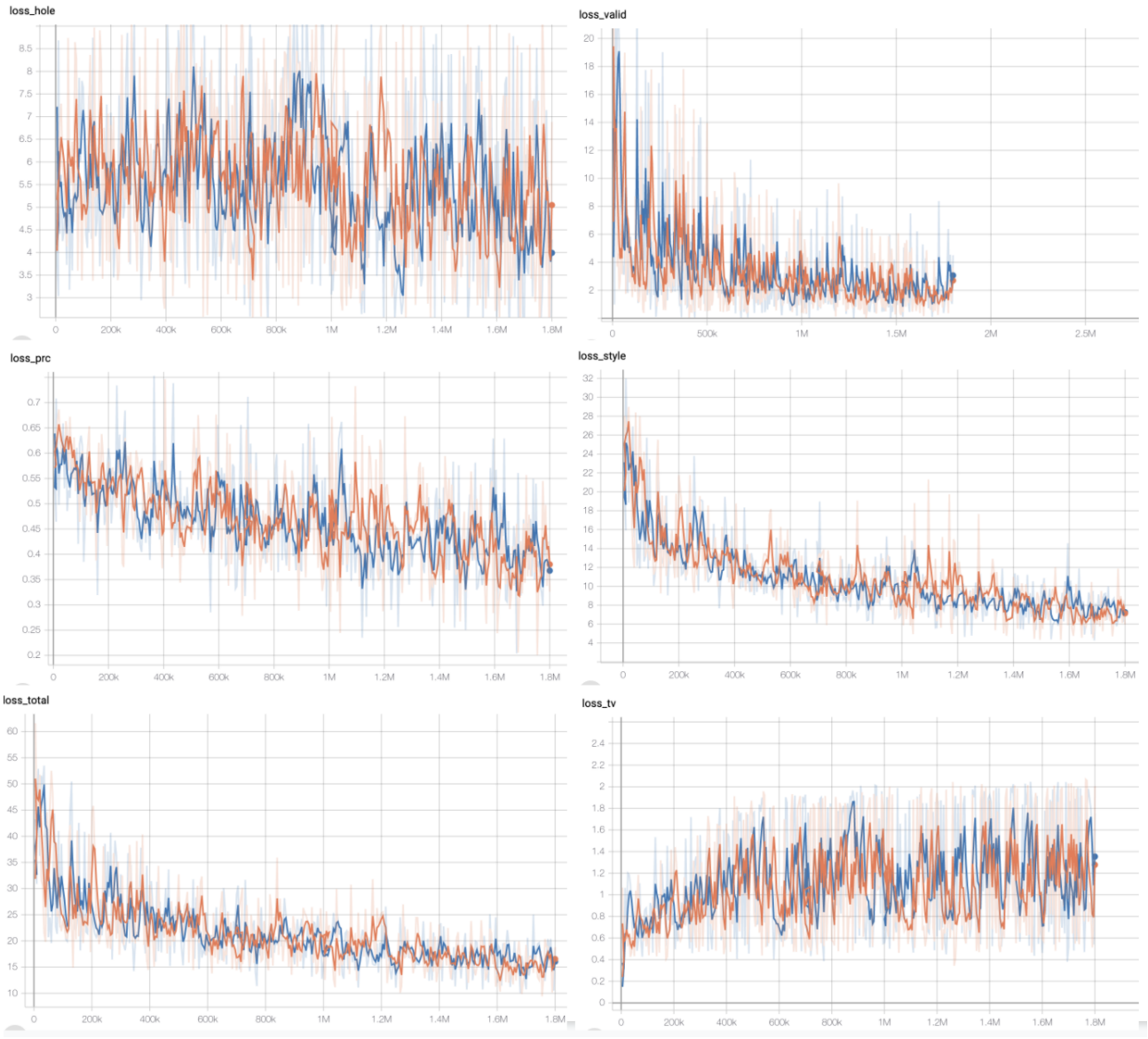

Figure 4.6: Losses for training data (orange) and validation data (blue) over 1.8×10^6 iterations over 8000 training and 1000 validation maps from the Maze datasets

Chapter 5

Conclusions and Future Work

5.1 Conclusions

The results of the simulations show that the proposed adaptive planning system for path planning strategies has an expected performance better than that of a randomly-chosen static path planning strategy to perform exploration task over an environment composed of a distribution of planning problems using A* with the inflation factor as the path planning strategy.

The proposed adaptive approach leverages the observation that planners can employ different planning strategies that can leverage the context of the planning problem, that is in this case the obstacle configuration. However, mapping from obstacle configuration to effective planning strategy is non-trivial. The proposed framework uses machine learning methods in meta-planner strategy prediction and map prediction to provide an understanding of the obstacle configuration, or planning problem context, that can be leveraged to adapt prior implemented planning strategies in contexts where they have the higher performance, thus on average the performance of the agent remains high and consistent across the distribution of planning problems $P(\Phi)$ encountered during exploration tasks without need for regular tuning or novel adaptations from the human designer. This in turn means for applications such as search-and-recuse missions we can have high consistent performance autonomous agents to carry out essential tasks of finding survivors and other objects of interest in disaster zones without operator intervening for adapting to changes in environments.

5.2 Future Work

Currently, the system is focused on the navigation aspect in the scope of exploration, the next steps would be to improve the goal estimation. For example, switching from frontier based exploration to information-gain based exploration, as the objective of the exploration is two-fold, to navigate efficiently and find the goal(s) as quickly as possible. While the latter is not covered by the scope of this thesis, it is a natural extension to the problem.

Goal estimation is another area of interest to explore within this framework. Currently, in the

exploration task, we have no prior information about the goal and assume a uniform probability of goal location in unknown space until it is found. It would be interesting to explore how adding a prior on the goal location could be leveraged by modules of the current framework. For example, we receive the local area map of the goal, then using map prediction we can estimate the goal location and drive our exploration towards those areas.

The current map representation is of an occupancy grid representation that adds the constraint of a fixed size map onto the problem. However, in real-life, map sizes are usually unknown or grow with exploration. It would be beneficial to this research to consider graphs instead of grid maps for environment representation as graphs are easily scalable. For simplicity, consider a graph representation of a grid map where each cell is a vertex and is only connected to its neighbors. Then we can expand the graph as needed. Map prediction and meta-planner strategy prediction would need some tweaks in their loss function and feature representation. Since we tend to overfit over images using the current VGG16 framework for the meta-planner strategy prediction, an extension would be to replace the model with Random Forests, or Decision Trees based on the planner distribution metrics define in Section: 4.1.1. One solution to capture the situation where the start and goal are in different connected components of the free space is to use the search-based planner's expanded area to find a path from the start and goal as a metric.

For testing, in the scope of this thesis we focused primarily on A* as the base planner and the planner set comprised with a range of inflation factor values, the next step would be to integrate and test with:

- Heuristic planners with a planner set of multiple heuristics or hyperparameter values that are dependant on the environment.
- Planner set of multiple planners with separable performance in different environments
- Tuning hyperparameter of a different set of planner, for example RRT from sample-based planning

In terms of the whole system, an interesting method to learn by experience could be by self-learning from past mistakes or less optimal planner strategy selections made in the past. Similar to how AlphaGo trains itself, an extension of this framework would be to train its adaptivity through prior experience.

Bibliography

[1] Jeffrey A Caley. Deep learning for robotic exploration. 2019.

[2] Jeffrey A Caley, Nicholas RJ Lawrance, and Geoffrey A Hollinger. Deep learning of structured environments for robot search. *Autonomous Robots*, 43(7):1695–1714, 2019. 2.3

[3] John Canny. *The complexity of robot motion planning*. MIT press, 1988. 1

[4] S. Choudhury, S. Arora, and S. Scherer. The planner ensemble: Motion planning by executing diverse algorithms. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2389–2395, 2015. 1

[5] S. Choudhury, A. Kapoor, G. Ranade, and D. Dey. Learning to gather information via imitation. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 908–915, 2017.

[6] Sanjiban Choudhury. *Adaptive Motion Planning*. PhD thesis, Pittsburgh, PA, February 2018. 2.2, 2.2.1

[7] Sanjiban Choudhury, Mohak Bhardwaj, Sankalp Arora, Ashish Kapoor, Gireeja Ranade, Sebastian Scherer, and Debadeepta Dey. Data-driven planning via imitation learning, 2017. 2.2.2

[8] Sanjiban Choudhury, Ashish Kapoor, Gireeja Ranade, and Debadeepta Dey. Learning to gather information via imitation. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 908–915. IEEE, 2017.

[9] Rina Dechter and Judea Pearl. Generalized best-first search strategies and the optimality of a. *Journal of the ACM (JACM)*, 32(3):505–536, 1985. 1

[10] Debadeepta Dey, Tian Yu Liu, Martial Hebert, and J Andrew Bagnell. Contextual sequence prediction with application to control library optimization. *Proceedings of robotics: Science and systems VIII*, 2013.

[11] Amine Elhafsi, Boris Ivanovic, Lucas Janson, and Marco Pavone. Map-predictive motion planning in unknown environments. *arXiv preprint arXiv:1910.08184*, 2019. 2.3

[12] David Hsu, J-C Latombe, and Rajeev Motwani. Path planning in expansive configuration spaces. In *Proceedings of International Conference on Robotics and Automation*, volume 3, pages 2719–2726. IEEE, 1997. 1

[13] Shikha Jain, S Nandy, G Chakraborty, CS Kumar, R Ray, and SN Shome. Error modeling of laser range finder for robotic application using time domain technique. In *2011 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*,

pages 1–5. IEEE, 2011. 3.1.1

[14] Xinyue Kan, Hanzhe Teng, and Konstantinos Karydis. Online exploration and coverage planning in unknown obstacle-cluttered environments, 2020. 2.3.2

[15] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011. 1

[16] Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning. 1998. 1

[17] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006. 1

[18] Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 85–100, 2018. 2.3, 4.3.2

[19] Matteo Luperto, Luca Fochetta, and Francesco Amigoni. Exploration of indoor environments predicting the layout of partially observed rooms. *arXiv preprint arXiv:2004.06967*, 2020.

[20] Ratnesh Madaan, Sam Zeng, Brian Okorn, and Sebastian Scherer. Learning adaptive sampling distributions for motion planning by self-imitation.

[21] Marco Morales, Lydia Tapia, Roger Pearce, Samuel Rodriguez, and Nancy Amato. *A Machine Learning Approach for Feature-Sensitive Motion Planning*, volume 17, pages 361–376. 10 2005. doi: 10.1007/10991541_25.

[22] Aleksandr I Panov, Konstantin S Yakovlev, and Roman Suvorov. Grid path planning with deep reinforcement learning: Preliminary results. *Procedia computer science*, 123:347–353, 2018. 3.2.3

[23] Walter D. Pullen. Perfect maze creation algorithms, 2015. URL `http://www.astrolog.org/labyrnth/algrithm.htm`. 3.2.1

[24] Barak Raveh, Angela Enosh, and Dan Halperin. A little more, a lot better: Improving path quality by a path-merging algorithm. *IEEE Transactions on Robotics*, 27(2):365–371, 2011.

[25] Charles Richter, William Vega-Brown, and Nicholas Roy. Bayesian learning for safe high-speed navigation in unknown environments. In *Robotics Research*, pages 325–341. Springer, 2018.

[26] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[27] Manish Saroya, Graeme Best, and Geoffrey A. Hollinger. Online exploration of tunnel networks leveraging topological cnn-based world predictions. In *Under Review*, 2020. 2.3, 3.5.1, 4.3.2

[28] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[29] N. Sturtevant. Benchmarks for grid-based pathfinding. *Transactions on Computational Intelligence and AI in Games*, 4(2):144 – 148, 2012. URL `http://web.cs.du.edu/`

~sturtevant/papers/benchmarks.pdf. 3.2.2

[30] Abhijeet Tallavajhula and Sanjiban Choudhury. List prediction for motion planning: Case studies. *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-15-25*, 2015. 2.2.1

[31] Abhijeet Tallavajhula, Sanjiban Choudhury, Sebastian Scherer, and Alonzo Kelly. List prediction applied to motion planning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 213–220. IEEE, 2016. 1, 2.2.1

[32] Sebastian Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002. 2.1, 2.1.1, 3.1.1

[33] Brian Yamauchi. A frontier-based approach for autonomous exploration. In *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97.'Towards New Computational Principles for Robotics and Automation'*, pages 146–151. IEEE, 1997.