Communications Coverage in Unknown Underground Environments

Michael Tatum

CMU-RI-TR-20-19

May 2020

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Thesis Committee:
Matthew Travers, Chair
Sebastian Scherer
Micah Corah

Thesis proposal submitted in partial fulfillment of the requirements for the degree of Master's of Science in Robotics

Abstract

In field robotics, maintaining communications between the user at a stationary basestation and all deployed robots is essential. This task's difficulty increases when the test environment is underground and the environment is unknown to the operator and robots. A common approach is to place a breadcrumb trail of communications nodes, or radio transceivers, from the deployed robots back to the basestation, autonomously if possible. Current communications network creation approaches fail to implement a near-optimal, if not optimal, approach to communications coverage in unknown environments and fail to maximize communications coverage when the number of radios is limited. We propose a node placement method that takes advantage of robot observations and existing sensor coverage solutions to optimize node location in unknown environments under sensor range and robot connectivity restrictions. In addition, this novel method for communications coverage is based on a method that approaches optimal coverage within $1 - \frac{1}{e}$ of the true optimal coverage in polynomial time, with the added complexity of obeying the parameters of the environment and maintaining communications between the robot and basestation.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank all those who have made the past two years at Carnegie Mellon University possible. I am thankful God has given me the opportunity to study and work at this prestigious university with people at the top of their field. Thank you to Matt Travers for giving me the opportunity to play an important role with Team Explorer in the DARPA Subterranean Challenge. Thank you Sebastian Scherer for giving me my first tasks in the Subterranean Challenge that allowed me to spearhead our communications efforts. To Micah Corah, thank you for continually challenging me in my thesis work, encouraging me to dive deeper into my research.

In Team Explorer, I want to specifically thank Rohit Garg, Katarina Cujic, Chao Cao, Ryan Darnley, Ian Higgins, Anthony Rowe, Bill Drozd, Graeme Best, Bob Debortolli, Nora Kazour, Manish Saroya, John Keller, Matthew Dworman, Vasu Agrawal, Mohammad Mousaie, and Vai Viswanathan, to name a few. You have all helped me on my journey, put up with my antics, and made the past two years more than simply another line on my resume. To all my housemates over the past two years, Dhruv Saxena, Thomas Weng, Tim Mueller-Sim, Jon King, Hunter Goforth, Scott Moore, Joseph Aroh, and Sam Nelson, thank you for being friendly faces and people to share life with. Thank you to the Zappas, the Schnecks, Rachel Weisz-Kaufman, Katie Kuhn, Keith Evans, Ashlyn Landrum, Madison Moitoso, Adam Prather, Don Gilbreath, Howie Choset, Sara Misra, Seth and Josh Austin, and the Sweeneys for being a constant encouragement and support. You all reminded me why I am here.

Thank you Kevin Pluckter for introducing me to the Robotics Institute and being a rewarding role model through this journey. Without you, the #kevmike-travelblog would not have been possible. Finally, thank you to my family, Mike Tatum, Tamara Tatum, Madelyn Tatum, Megan Tatum Miller, and Ben Miller. You all saw me through the ups and downs of the past two years and never gave up on me. Without you, this thesis would not be possible.

Chapter 1

Introduction

In field robotics applications, the goal is often to sense the environment, plan a course of action, and then take act. These tasks are often treated disparately, but ideally they would all occur together for a fully integrated system. This work focuses on one specific aspect of the problem of creating a fully integrated system: communications. In field robotics applications that are above-ground and localized to a specific area, communications are often taken for granted because radio transceivers and stationary networks can be fixed in one location and reliably reach all necessary agents, not to mention access to a global positioning system (GPS) above-ground. In applications underground with a stationary operator and basestation and agents that can traverse complex tunnel, urban, and cave environments, communications are more challenging. This work addresses this challenge with the added practical application of Carnegie Mellon University's current involvement in the Defense Advanced Research Projects Agency (DARPA) Subterranean Challenge [3], competing as Team Explorer. Specifically, we investigate how different types of prior observations can be leveraged to solve sensor coverage in a priori unknown environments.

Motivation

In light of recent events, such as the Chilean mining accident in 2010 and the Tham Luang cave rescue in 2018, field robotics research initiatives, such as the DARPA Subterranean Challenge, are putting increased focus on mapping, navigating, and searching underground environments autonomously. Specifically, the DARPA Subterranen Challenge calls competitors to deploy teams of unmanned vehicles in tunnel, urban, and cave environments to identify and localize specified artifacts, such as backpacks, fire extinguishers, and survivors, to name a few. The DARPA Subterranean Challenge consists of three preliminary Circuit events (Tunnel, Urban, Cave) and a final Grand Challenge that combines all three. At each Circuit, teams compete in one-hour long full-scale deployments in two separate courses representative of the environment type being tested to map the environment and identify and localize as many artifacts as

possible. Sending out teams of mobile robots in underground environments is a major engineering achievement, but operator interaction and, therefore, mission success is nearly impossible without effective communications from the operator to all deployed robots. Effective communication is dependent upon sensor coverage in these unknown environments, specifically using communications nodes, or radio transceivers, to send and receive messages. Current approaches to solving this problem have been seen in the DARPA Subterranean Challenge's Tunnel Circuit in Summer 2019 and Urban Circuit in Winter 2020. These approaches include tethering the user's ground robot to their basestation to keep constant wired communications, using a separate robot to act as a data mule between the deployed robots and the basestation, and dropping communications nodes throughout the environment to create a mobile ad-hoc network (MANET) from the basestation to all deployed robots.

Background

Team Explorer is using the MANET approach, as it, if implemented properly, will allow for constant communication between the basestation and all deployed robots within range of these dropped nodes. Communications through tethering and data muling have their own challenges, but this work focuses on the challenge of dynamically constructing a network, requiring a method for determining where to place a limited number of nodes. Team Explorer places nodes in their testing environments by dropping them out of the node droppers on the back of their unmanned ground vehicles (UGVs), pictured in Figure 1.1. Currently Team Explorer's two largest ground robots are limited to carrying nine nodes each. Therefore, dropping nodes optimally is essential to successful communications and ensuring the longevity of a successful test. Placing nodes sub-optimally in tight corners and against walls leading to dead ends can severely limit communications, limiting the operator's interaction with the robots throughout the test. Placing nodes optimally, in free spaces that lead to unexplored areas, allows operator interaction throughout testing and ensures all artifact detections return to the basestation, leading to better overall performance.

(a) Team Explorer's UGV and UAV

(b) Node Dropper

Figure 1.1: Team Explorer's robot hardware used in the Urban Circuit.

For all events thus far in the DARPA Subterranean Challenge, the SubT Integration Exercise (STIX), the Tunnel Circuit, and the Urban Circuit, Team Explorer's node dropping approach has been as follows: a node is dropped from a robot if the Received Signal Strength Indicator (RSSI) between the robot and it's nearest dropped node is below a threshold or if the robot is not within line of sight of a dropped node and a stricter RSSI threshold is not met. RSSI is the estimated power level measure between a radio frequency (RF) client device and an access point. RSSI decreases as the distance between radio transceivers increases, as shown by Figures 1.3 and 1.4, which display the data from a field test in which no communications nodes were dropped and the robot, connected to a node, drove from a separate node at the origin until the robot was no longer in communications range with the node at the origin. Team Explorer's approach, that will henceforth be referred to as the Naive Approach, was sub-optimal for three main reasons. First, the line of sight restriction, measured using ray casting, caused nodes to pile in tight areas with many corners, since any sudden turns would cause the robot to lose line of sight to any previously dropped nodes. An example of this is shown in 1.2. In addition, radio transceivers are not strictly limited to line of sight connectivity, as shown by the RSSI improvement in Figure 1.3 between time steps 522 and 600, even though the robot is no longer in line of sight with the node at the origin, as shown in comparing Figures 1.4a and 1.4b. In response to this, Team Explorer added the condition that the robot could be out of line of sight within a stricter target RSSI range. Due to the rapid variability of RSSI measurements, as shown by the noise in Figure 1.3, this still proved sub-optimal, resulting in piling of nodes in complex environments.

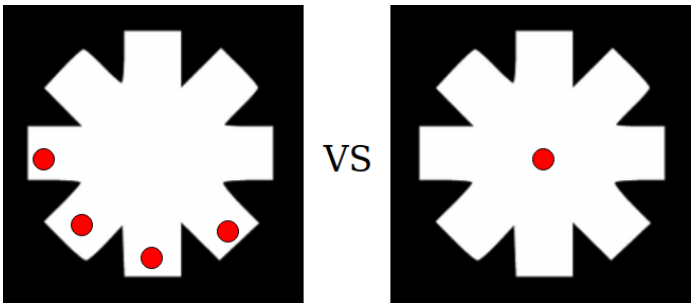


Figure 1.2: The free area to be covered in communications is white and occupied area is black. If the agent is dropping nodes whenever it is out of line of with any other dropped node, represented by the red dot, nodes will just as easily be "piled" in dead ends (left) as they will be placed in the center (right). The left is an example of sub-optimal node placement from piling.

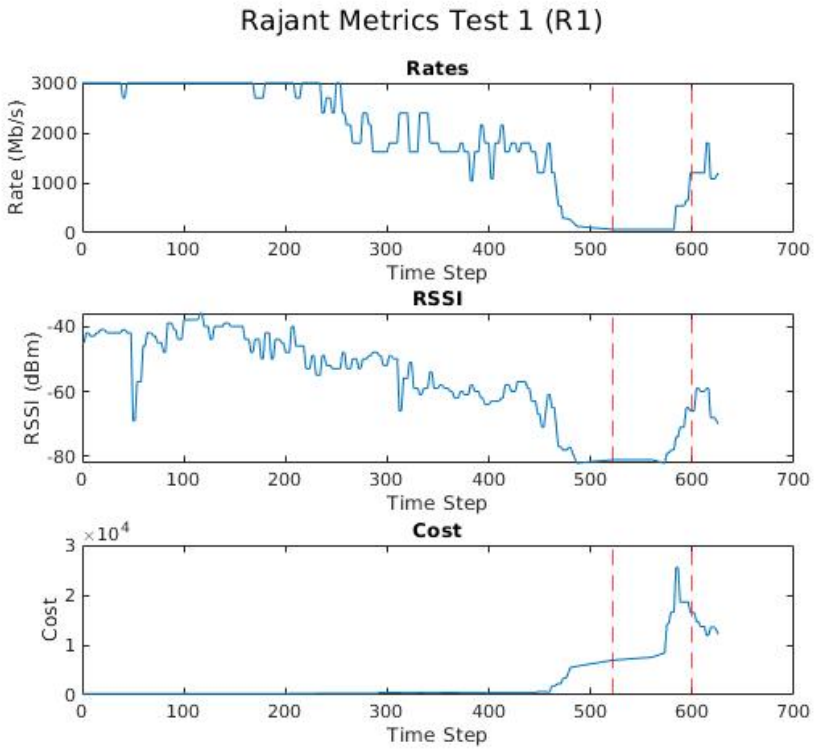
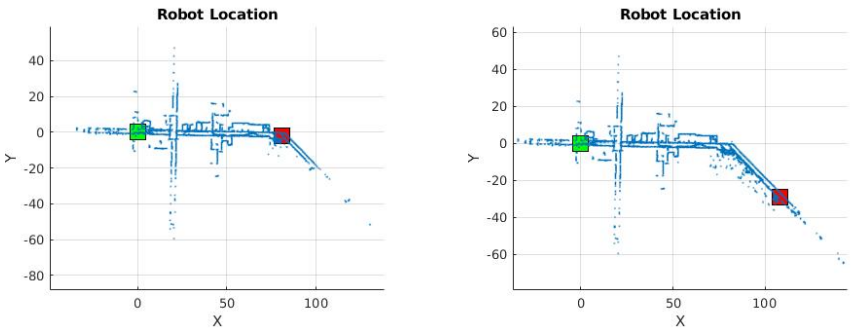


Figure 1.3: Rajant radio transceiver (Section 3.1) data from field testing Team Explorer's robot (R1) in an urban environment. Nodes are dropped based on RSSI values. As RSSI and throughput (Rates) drop, Cost, a weighted combination of RSSI, Rate, and other measurements, rises. Vertical red dashed lines indicate the time steps at which the snapshots in Figures 1.4a and 1.4b are taken, at time steps 522 and 600, respectively. The RSSI values at these time steps are -81 dBm and -66 dBm, respectively.



(a) Time Step 522, -81 dBm (b) Time Step 600, -66 dBm

Figure 1.4: Robot locations (meters) during field testing to measure Rajant radio transceiver metrics in an urban environment. The green square indicates the robot's origin at time step zero and the red square indicates its position at the current time step. All blue dots make up the point cloud of the map created by the robot.

The second reason the Naive Approach was sub-optimal was because nodes were dropped with no regard to their placement in the global environment. In testing, a node was dropped in a tight dead-end corner of the environment just as easily as it was dropped in a free hallway leading to many areas of potential future coverage. Finally, the Naive Approach was based solely on the map known to the robot, only dropping nodes in traversed areas, not considering the potential value of placing nodes in unexplored areas of the environment. These causes of sub-optimal dropping and the desire to create a robust MANET in any environment, known or unknown, inspired this research. This work will present our novel method of communications coverage in unknown underground environments for maximum connected coverage with a fixed number of sensor nodes in polynomial time. Our goal is to optimize the current state of the art in autonomous sensor coverage to allow Team Explorer to autonomously deploy communications nodes in an unknown environment in the DARPA Subterranean Challenge to maximize communications coverage.

Challenges & Contributions

Without a priori knowledge of the environments, dropping nodes poses two main challenges that must be addressed. First, the environment's map is unknown except for areas explored by the agent. This leads to the challenge of predicting the map accurately based on the agent's observations. Our contribution is to apply existing methods for map prediction, using Convolutional Neural Nets (CNNs), to our unique application environments: tunnel, urban, and cave. Once the unexplored environment is predicted, the agent must solve for placement loca-

tions of the available nodes that ensures there will be communications between the agent's start position and every node dropped, either directly or through a chain of nodes. We address this challenge by solving for greedy connected coverage, explained in Section 2.1.3, and modifying the agent's exploration strategy to place nodes as necessary. Finally, our contributions are evaluated by experimental analysis of the combination of our node placement approaches on different map types with different CNNs and experimental comparison of this research's newfound node placement approach with that currently used by Team Explorer. The block diagram in Figure 1.5 demonstrates a high-level overview of the node dropping process employed by this work.

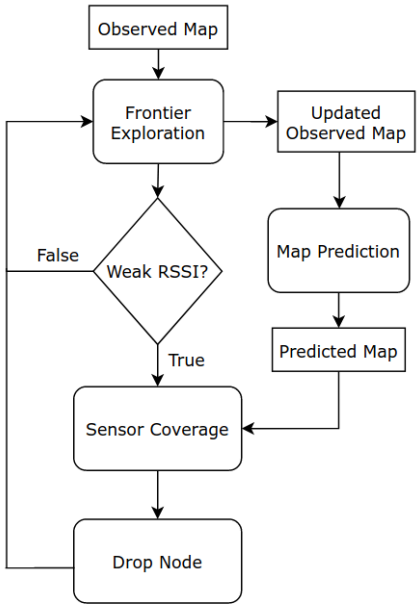


Figure 1.5: An initial Observed Map of the agent's environment is used to calculate the agent's next goal destination with Frontier Exploration. As the agent moves with Frontier Exploration, the Observed Map is updated, which is used for Map Prediction. If the agent's RSSI to all dropped nodes is weak, sensor coverage is solved for potential node placements using the predicted map. A node is dropped at the calculated location nearest to the agent. If there is not weak RSSI or the agent just dropped a node, the agent continues exploring with Frontier Exploration.

Chapter 2

Related Work

This section will discuss previous methods of sensor coverage, map prediction, and frontier exploration that have inspired the proposed node dropping method.

2.1 Sensor Coverage

Currently, the state of the art in sensor coverage is for known environments. Approaches, such as polygon division and the Art Gallery Problem (AGP), are geometry-based sensor coverage solutions. Kazazakis et al. proposed a polygon division method of sensor coverage for limited-visibility guards, or sensors, in 2D areas, but the map is a required input to solving the placement[13]. This approach can also result in sub-optimal placement where neighboring sensors are closer than the given radius because global improvement is not considered in the search for complete coverage. Sub-optimal placement results in dropping more nodes than necessary, costly to an application with limited nodes. Another proposed solution is the AGP, a computational geometry problem which takes a known environment and places the minimum number of guards G, or communications nodes for our purposes, such that there is a line segment between every point p within the polygon P and every guard $g \in G$ that does not leave P. This approach guarantees the minimal number of nodes will be dropped, but does not work for our application because it does not consider guards with limited range. The Art Gallery Problem with Fading (AGPF) [6] addresses this issue by treating each guard as a light source where the coverage of the light fades over distance. While this would result in optimal line of sight placement of nodes, placing nodes based solely on if placement is line of sight to another dropped node has resulted in sub-optimal coverage and communications for Team Explorer in the DARPA Subterranean Challenge thus far, as explained in Section 1. Therefore, developing a placement algorithm for coverage of an unknown area with limited-range sensors that need to all be connected explicitly or implicitly through a chain without necessarily being line of sight is essential to creating a reliable communications network online in an unknown

underground environment.

2.1.1 Set Cover Problem

Transitioning to a coverage approach based on sensors with fixed range brought us to the set cover problem. The set cover problem is an NP-complete problem where, given a universe $\mathcal{U}$ of elements e and a collection of sets $\mathcal{S}$ from $\mathcal{U}$ whose union comprises the entire universe, one must find the smallest set cover S, the smallest subset of $\mathcal{S}$ whose union comprises the universe [12]. According to [20], the optimal set cover solution is an integer linear program defined as follows:

$$\begin{aligned}
&\min \sum_{S \in \mathcal{S}} x_S \quad \text{(minimize number of sets chosen)} \\
&\text{s.t.} \sum_{S: e \in S} x_S \geq 1 \;\; \forall e \in \mathcal{U} \quad \text{(cover every } e \text{ in } \mathcal{U}) \\
&x_S \in \{0, 1\} \;\; \forall S \in \mathcal{S} \quad \text{(every set } x_S \text{ is either in } S \text{ or not)}
\end{aligned}$$

Since this problem is NP-hard, it cannot be solved in polynomial time unless variable conditions are relaxed to

$$x_S \in [0, 1] \;\; \forall S \in \mathcal{S}.$$

Team Explorer has a limited number of sensors to drop. Therefore, the set cover problem was deemed too broad since, as maps grow, computation time and the number of potential sensor placements will as well. In addition, if the world the agent traverses is large enough, it will not have enough nodes to solve the set cover problem. This leads us to a solution that will maximize coverage with a fixed number of sets, the maximum coverage problem.

2.1.2 Maximum Coverage Problem

Using a finite number of sensors with fixed range to cover the maximum area possible can be described as an instance of the maximum coverage problem [2]. Given a universe $\mathcal{U}$ and a set system S within the collection of sets $\mathcal{S}$ whose union comprises $\mathcal{U}$, maximum coverage, according to [2], is:

$$\max_{X \subset S, |X| \leq k} |Y| \text{ (maximize number of covered elements)}$$

$$\text{where } Y = \bigcup_{x \in X} x \text{ (union of all sets chosen)}$$

where k is the maximum number of chosen sets. For our purposes, a set is a node and the entire traversable area covered by it is its range. An example of a solution to the maximum coverage problem is given in Figure 2.1 [22]. This problem is NP-hard, thus making it infeasible for a real-world application, leading us to the greedy maximum coverage algorithm.

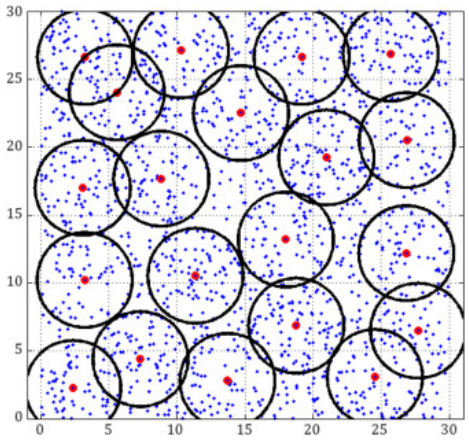


Figure 2.1: This sample solution to the maximum coverage problem shows a fixed number of 20 sets, represented by red dots with fixed black circle range, maximally covering as many blue dots as possible.

2.1.3 Greedy Maximum Coverage Algorithm

The greedy maximum coverage algorithm chooses the set with the maximum incremental reward. At each iteration, the set covering the most uncovered elements is chosen until k sets are chosen. This polynomial time greedy solution to the maximum coverage problem constructively approximates the optimal maximum coverage [7]. For our purposes, polynomial time is ideal since the algorithm is run at least as many times as there are nodes to drop. The greedy maximum coverage algorithm constructively approximates optimal maximum coverage within a minimum ratio of $1 - \frac{1}{e} \simeq 0.632$ [7]. Feige et al. [7] prove this as follows:

Proof. The optimal maximum coverage solution consists of set $S' \subset S$. S' covers n' elements. Since it is optimal, S' can be covered by k sets. Therefore,

$$n_i \geq \frac{n' - \sum_{j=1}^{i-1} n_j}{k}$$

where n_i is the number of elements covered by the ith set from the greedy algorithm. Furthermore,

$$\Sigma_{j=1}^{i} n_j \geq n' - n' \left(1 - \frac{1}{k}\right)^i$$

$$\sum_{i=1}^{k} n_i \geq n' - n' \left(1 - \frac{1}{k}\right)^k \geq n' \left(1 - \frac{1}{e}\right)$$

□

Our sensor coverage method is based on the greedy maximum coverage algorithm, with modifications made for our node placement application to create a connected network. We will refer to the modified algorithm we use as greedy connected coverage, represented by Sensor Coverage in Figure 1.5.

2.2 Map Prediction

Map prediction has been implemented by various methods for numerous applications. Richter et al. used partially observable Markov decision processes (POMDPs) to determine the speed at which a robot should approach a frontier to safely navigate a map as fast as possible [16]. While using a POMDP could be beneficial for local path planning decisions, it is currently not clear how they could be used for a globally optimal node placement strategy of an entire map, leading us to consider CNNs. CNNs are a class of deep neural networks which assigns weights and biases that can be learned to aspects of and objects in an input image for differentiation, leading to classification. Caley et al. used a CNN to predict exit locations in buildings based on 2D images of the building floor plans [1]. The agent is given the floor plan of the environment, unlike in our applications. Even still, this work demonstrates the value of using CNNs trained on a large dataset of image input for map prediction applications. This led us to use the U-Net CNN[17] and image inpainting [14], based on the work of Saroya et al. in [18], represented by Map Prediction in Figure 1.5.

2.2.1 U-Net

U-Net was chosen because it can accept an input image of any size, valuable since the map sizes in the DARPA Subterranean Challenge have varied from 90 meters (m) $\times$ 90 m to 400 m $\times$ 400 m thus far. This is because U-Net is a fully convolutional network made up of only convolutional layers, no dense layers, allowing us to work with map inputs of different sizes. U-Net is used for image classification and localization tasks, where the contents of an image and the locations of those contents are identified, respectively. In addition, U-Net is known for its symmetrical downsampling (encoding) and upsampling (decoding) methods. In a regular convolutional network with only pooling and dense layers, we can classify an image but there is no upsampling of the image, preventing us from localizing the information of interest. U-Net performs upsampling using transposed convolution, also known as deconvolution. Localization occurs when high resolution feature maps from the encoder are concatenated to the transposed convolutional layers in the decoder at the same level. This combination of information helps create a more precise output in the following convolution layer. Furthermore, the U-Net network is without any fully connected layers and the segmentation map only includes the pixels from the valid part of each convolution, allowing segmentation of larger images using the same network. The border region of the image consists of the pixels outside of the valid part of each convolution, also known as the receptive field or context. The overlap-

tile strategy extrapolates these border pixels by mirroring the input image[17]. Figure 2.2 displays the output of a trained U-Net network applied to a partial map.

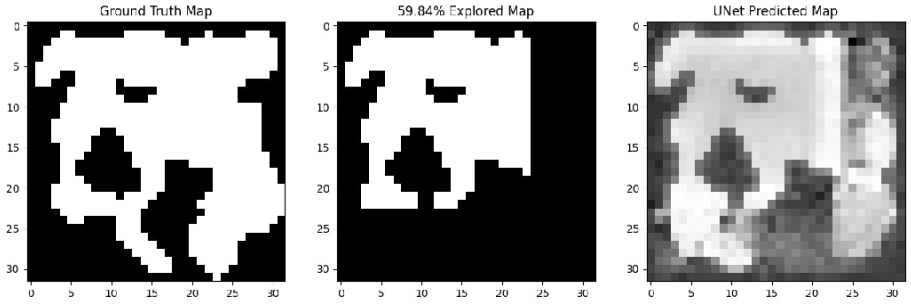


Figure 2.2: U-Net map prediction example. The left image is the ground truth map of a cave environment, where white cells are free and black cells are occupied. The middle image is 59.84% of the ground truth map, representing the amount the map has been explored and observed. The right map is the U-Net output of the full map after the 59.84% Explored Map has passed through the network. Lighter cells represent a higher probability of being a free cell.

2.2.2 Inpainting Loss

Once an image is passes through the U-Net CNN, image inpainting is used to calculate the losses used for training. Image inpainting, often used to fill holes in images, is used in this research for pixel-based reconstruction and composition of maps to ensure predicted cells, whether free or occupied, transition smoothly into their surrounding cells [14]. Image inpainting's losses are the per-pixel losses, $\mathcal{L}_{hole}$ and $\mathcal{L}_{valid}$, spatial feature losses, $\mathcal{L}_{perceptual}$ and $\mathcal{L}_{style}$, and the total variation loss $\mathcal{L}_{tv}$. $\mathcal{L}_{hole}$ and $\mathcal{L}_{valid}$ are used to better predict the ground truth image $\mathbf{I}_{gt}$ from the input image with a hole $\mathbf{I}_{in}$. $\mathcal{L}_{hole}$ is the L1 loss between $(1 - \mathbf{M}) \times \mathbf{I}_{out}$ and $(1 - \mathbf{M}) \times \mathbf{I}_{gt}$, where $\mathbf{M}$ is the initial binary mask and $\mathbf{I}_{out}$ is the predicted map. $\mathbf{M}$ is equivalent to $\mathbf{I}_{in}$ with added ones, or occupied cells, representing the walls that surround the known free cells added to the binary array. Knowledge of where untraversable walls are located allows for better prediction of where potential free areas exist and where they do not, resulting in better overall map predictions. $\mathcal{L}_{hole}$ improves the per-pixel accuracy of predictions for unexplored regions. $\mathcal{L}_{valid}$ is the L1 loss between $\mathbf{M} \times \mathbf{I}_{out}$ and $\mathbf{M} \times \mathbf{I}_{gt}$. $\mathcal{L}_{valid}$ improves the per-pixel accuracy of predictions for explored regions and heavily increases when known pixels are changed, encouraging compliance to the ground truth. To ensure not only per-pixel accuracy is maintained, but also spatial features, inpainting loss includes $\mathcal{L}_{perceptual}$ and $\mathcal{L}_{style}$. These losses use the ImageNet-pretrained VGG-16 network for feature extraction to project the images into a higher level feature space [19]. $\mathcal{L}_{perceptual}$

is defined as

$$\mathcal{L}_{perceptual} = \sum_{p=0}^{P-1} \frac{\left\| \Psi_p^{\mathbf{I}_{\text{out}}} - \Psi_p^{\mathbf{I}_{gt}} \right\|_1}{N_{\Psi_p^{\mathrm{I}_{gt}}}} + \sum_{p=0}^{P-1} \frac{\left\| \Psi_p^{\mathbf{I}_{comp}} - \Psi_p^{\mathbf{I}_{gt}} \right\|_1}{N_{\Psi_p^{\mathrm{I}_{gt}}}}$$

where $\Psi_p^{\mathbf{I}_*}$ is the activation map from the pth layer of P total layers from the VGG-16 network for input $\mathbf{I}_*$. $N_{\Psi_p^{\mathrm{I}} g_t}$ is the number of elements in $\Psi_p^{\mathrm{I}_{gt}}$. In addition, $\mathbf{I}_{comp}$ is $\mathbf{I}_{out}$ with known, or explored, pixels set to ground truth [19]. $\mathcal{L}_{perceptual}$ is shown to decrease the recreation of grid-shaped artifacts [14], which is why we keep its weight low. When applying this method to real-world maps, as opposed to gridworld maps, it may be valuable to increase the weight of $\mathcal{L}_{perceptual}$ to improve the smoothness of the predictions. To compute $\mathcal{L}_{style}$, we apply an autocorrelation, or Gram matrix, to each feature, and then compute the L1 loss. In

$$\mathcal{L}_{style_{out}} = \sum_{p=0}^{P-1} \frac{1}{C_p C_p} \left\| K_p \left(\left(\Psi_p^{\mathbf{I}_{out}}\right)^\top \left(\Psi_p^{\mathbf{I}_{out}}\right) - \left(\Psi_p^{\mathbf{I}_{gt}}\right)^\top \left(\Psi_p^{\mathbf{I}_{gt}}\right) \right) \right\|_1$$

$$\mathcal{L}_{style_{comp}} = \sum_{p=0}^{P-1} \frac{1}{C_p C_p} \left\| K_p \left(\left(\Psi_p^{\mathbf{I}_{comp}}\right)^\top \left(\Psi_p^{\mathbf{I}_{comp}}\right) - \left(\Psi_p^{\mathbf{I}_{gt}}\right)^\top \left(\Psi_p^{\mathbf{I}_{gt}}\right) \right) \right\|_1$$

$C_p \times C_p$ is the gram matrix, where C_p is the number of channels in the pth selected layer and the normalization factor is

$$K_p = \frac{1}{C_p H_p W_p}$$

where H_p and W_p are the height and width of the pth selected layer, respectively. Finally, total variation loss $\mathcal{L}_{tv}$ encourages smoothness between neighboring pixels by taking the L1 loss between $\mathbf{I}_{comp}$ pixels and their neighbors in both the x and y directions [14].

$$\mathcal{L}_{tv} = \sum_{(i,j)\in R,(i,j+1)\in R} \frac{\left\| \mathbf{I}_{comp}^{i,j+1} - \mathbf{I}_{comp}^{i,j} \right\|_1}{N_{\mathbf{I}_{comp}}} + \sum_{(i,j)\in R,(i+1,j)\in R} \frac{\left\| \mathbf{I}_{comp}^{i+1,j} - \mathbf{I}_{comp}^{i,j} \right\|_1}{N_{\mathbf{I}_{comp}}}$$

$N_{\mathbf{I}_{comp}}$ is the number of elements in $\mathbf{I}_{comp}$. The total loss $\mathcal{L}_{total}$, used to optimize training, is a weighted combination of all these losses[14].

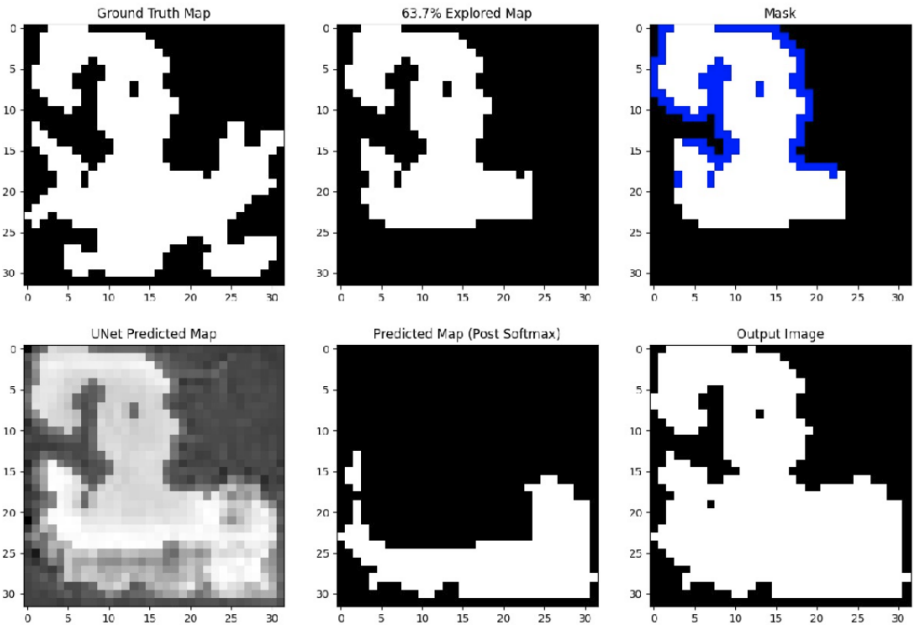


Figure 2.3: The map prediction process from ground truth image $\mathbf{I}_{gt}$ (top left) to input image with hole $\mathbf{I}_{in}$ (top middle) to binary mask $\mathbf{M}$ (top right). The U-Net predicted map (bottom left) leads to the newly predicted map (bottom middle) which, added to $\mathbf{I}_{in}$, results in the output predicted map $\mathbf{I}_{out}$ (bottom right).

2.3 Frontier Exploration

Similar to how Team Explorer's robots explore an environment, this work will use frontier exploration. Frontier exploration, popularized by the Yamauchi method [21], is a complete exploration approach, meaning the entire environment will be explored. The Yamauchi method continuously moves the agent to the nearest frontier. Frontiers are regions marking the transition between known free areas and unexplored areas that could be either free or occupied, meaning the area is traversable or untraversable, respectively. Frontier-based exploration, through the Yamauchi method, chooses newly generated frontiers for the agent to set as its destination, allowing the agent to gain more knowledge of its environment, until it has full knowledge of the environment. When there are no more frontiers, the agent has finished exploring. The Yamauchi method is continuing to be used alongside A* path planning to fully explore unknown environments, with additional pruning of frontier cells to improve computation [15]. Our frontier exploration method is represented by Frontier Exploration in Figure 1.5.

Chapter 3

Approach

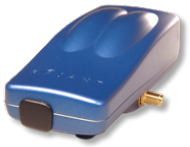
This section will describe the radio transceiver hardware and software used by Team Explorer, giving context to the communications strategy pursued by Team Explorer and the need for improved communications coverage. The radio performance is outlined to create a realistic model of the type of nodes and environments that will be used to test our node dropping approach. The map generation method is explained to demonstrate how highly variable real-world maps can be created to help train networks to learn map structures, leading to the map prediction method used for this research. This chapter also explains how we incorporate and deviate from Yamauchi Frontier Exploration in our approach, leading to all the node placement approaches implemented to be tested on the numerous environments generated.

3.1 Hardware & Software

In order to maintain communications between the robots and basestation at all times, Team Explorer has resolved to dynamically deploy a MANET as the ground robots traverse the environments. To accomplish this, the choice of hardware for radio transceivers is of the utmost importance, as well as the network meshing software that accompanies the radios. For the STIX Event and Tunnel Circuit, Team Explorer used 30 Ubiquiti UniFi Mesh Access Points (UAP-AC-M) [11], pictured in Figure 3.1a, with the Better Approach To Mobile Adhoc Networking (B.A.T.M.A.N.) routing protocol [8]. The UAP-AC-M is a dual-band 2.4 GHz and 5GHz, 2x2 multiple-input multiple-output (MIMO) transceiver. Further specifications on the UAP-AC-M hardware can be found in Table A.1. B.A.T.M.A.N., developed specifically for MANETs, decentralizes what is considered the network's best route so that no single node passes all the data and network changes do not need to be communicated to each node. Each node only needs to be aware of where it receives data from and to where it is sending the data. This process gives the packets individual routes developed dynamically. B.A.T.M.A.N. is developed on a data link layer, or Layer

2, network, meaning data is transferred between neighboring wide area network (WAN) nodes or nodes sharing a local area network (LAN). While routing through B.A.T.M.A.N. met our initial requirements of dynamically building a network, this protocol and the UAP-AC-M hardware caused several issues that required us to move to new hardware following the Tunnel Circuit. Firstly, the B.A.T.M.A.N. meshing protocol often caused remeshing, or network reconfiguration, of nodes at inopportune times that led to the communications being down anywhere from 10 seconds to three minutes. This problem was exacerbated by having each ground robot carry nine nodes containing UAP-AC-M radios to be dropped at user-defined points, increasing the number of possible meshing configurations. We attempted to alleviate this issue by only powering nodes to be dropped one node in the queue prior to when they were to be dropped, allowing time for the radios and their software to boot up, a process that took one to two minutes, while decreasing the number of nodes actively meshing at one time. We knew this solution was not sustainable in the long term due to the long boot up time of the nodes combined with the uncertainty of node placement needs for the testing environments and the speed at which the ground robots would be traveling. This led us to our current radio hardware and software, the Rajant Breadcrumb DX2 alongside the InstaMesh routing protocol [10].

(a) Ubiquiti UAP-AC-M



(b) Rajant DX2

Figure 3.1: Team Explorer communications hardware used in the STIX Event and Tunnel Circuit (a) and the Urban Circuit (b)

Team Explorer is using 30 Rajant Breadcrumb DX2s, specifically the DX2-50 model, a 5.0 GHz, 2x2 MIMO, 300 Mbps transceiver, shown in Figure 3.1b. The DX2 was chosen over the UAP-AC-M because of its higher transmit power, smaller form factor, lighter weight, significant for aerial vehicles, wide power supply range, and increased communications range, all shown in Table A.1. The wide power supply range allowed us to power it directly from a battery as opposed to through a custom-made board with a voltage regulator, like we had done for the UAP-AC-M radios. All Rajant radios use their proprietary InstaMesh meshing protocol. Like B.A.T.M.A.N., InstaMesh directs mesh reconfiguration as nodes move in Layer 2, but InstaMesh does not demonstrate

the constant remeshing behavior seen in B.A.T.M.A.N. Using DX2s with Instamesh allowed us to keep all nodes powered in the network at all times, those on the robots and basestation as well as those being dropped throughout the environment. The meshing configuration according to InstaMesh would rarely change unless there was a significant cost change between nodes, based on Rajant's proprietary cost metric. This cost metric is based on data rate in Mbps, TX power in dB, RSSI in dBm, and signal-to-noise ratio (SNR) in dB. The rest of this work will focus on performance using the Rajant Breadcrumb DX2s.

3.2 Radio Performance

In testing and competition runs for the DARPA Subterranean Challenge, we tended to be conservative when dropping communications nodes in terms of distance between each node to preserve strong connections. In addition, we aimed to keep all nodes dropped within line of sight of another dropped node since, in underground environments with dense walls, communications often will not travel as easily through these walls as in an above-ground office or lab environments. Figure 3.2 shows a point cloud created by one of Team Explorer's ground robots in the Urban Circuit in Winter 2020. Each cell in the grid is 10 m $\times$ 10 m. The cyan dot markers on the map represent where nodes were placed during Team Explorer's hour-long run on this course. There is also a node, not pictured, at the basestation located at the start gate in the bottom right corner of the point cloud map. As can be seen, the farthest a single node is from any other node is approximately five cells, demonstrated by the leftmost node on the map. This means that node is roughly 50 meters from its nearest node. While this node is not line of sight to any other node, there are no thick walls completely blocking its signal from reaching the node five cells to its right. Based on this, and extensive testing done with the Rajant Breadcrumb DX2 Radios in underground tunnels and urban environments, as shown in Section 1, our metric for placing nodes moving forward will be that they have a range of 30 meters in underground environments. Following this metric, two nodes can be 60 meters apart and still communicate if not completely blocked by obstructions since their signal radii will meet. This information will be valuable for the simulation shown in Section 4.2.

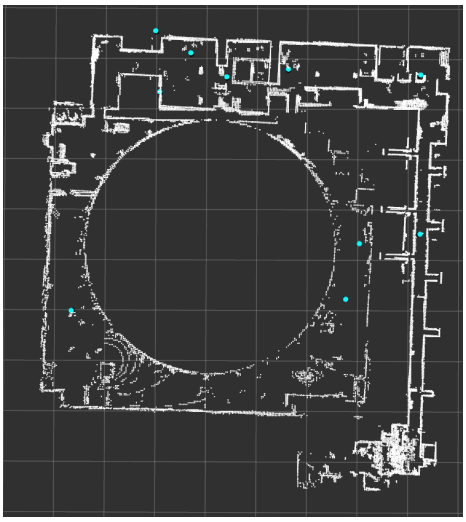


Figure 3.2: Point cloud map of the SATSOP nuclear reactor in Elma, Washington taken at the DARPA Subterranean Challenge Urban Circuit in Winter 2020. Cyan dots indicate node placements.

3.3 Map Generation

Four types of gridworld maps are used in this research: cave, urban, tunnel, and hybrid, a combination of the three. The maps used for map prediction and sensor coverage were generated based on John Conway's Game of Life [9], a famous cellular automaton which applies the following rules to a grid of cells that are either alive or dead:

1. A living cell with less than two living neighbors dies.
2. A living cell with two or three living neighbors lives.
3. A living cell with more than three living neighbors dies.
4. A dead cell with exactly three living neighbors becomes alive.

For map generation, we simplify these criteria to be:

1. A living cell with less than *death limit* living neighbors dies.
2. A living cell with greater than or equal to *death limit* living neighbors lives.
3. A dead cell with greater than *birth limit* living neighbors becomes alive.
4. A dead cell with less than or equal to *birth limit* living neighbors dies.

The *death limit* and *birth limit* parameters are changed based on the type of map generated. The maps used in this research are two-dimensional boolean arrays where false (alive) is represented by a black cell, meaning the cell is occupied and cannot be traversed and true is represented by a white cell, meaning the cell is free (dead) and can be traversed. When initializing the map, we randomly set each cell to be dead or alive with the percent chance of being alive changed based on the different environment maps discussed. We calculate the new cell value for each cell based on the values of its eight neighbors and our simplified rules of The Game of Life and put the resultant value into its corresponding position in a new grid, so that the new values do not affect the old. We repeat this process on each updated grid for the user defined number of steps, in our case three. Figure 3.3 demonstrates the map changing over these steps to become a map with the same properties as one of the cave maps we used for testing. Examples of the different maps created using this method are in Section 4.2.1.

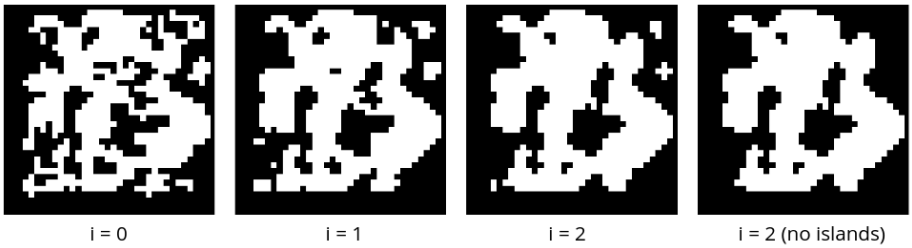


Figure 3.3: From left to right, we see the map improved with iterations based upon The Game of Life to produce a cave-like map. The last map on the right is the final map used for testing with all islands of free cells not neighboring the main free cave cells removed to allow traversal among all free cells.

3.4 Map Prediction

In simulation, as the agent is traversing the created gridworld map, where zeros represent occupied cells (walls) and ones represent free cells (traversable space), the agent is only aware of cells within a range of one cell in all directions, including diagonal, from the cell it is currently at and the cells it has traversed. To maximize communications coverage while placing nodes in the map in real time, the agent must predict the rest of the map based on the free and occupied cells it already knows. We assume the agent is aware of the total size of the gridworld, a safe assumption given Team Explorer is often aware of the total size of their deployment environment when testing in real-world environments. Map prediction is accomplished using a combination of the U-Net CNN architecture [17] and image inpainting [14]. Since the networks will not predict the unknown map with 100% accuracy, as shown by the losses in Section 4.1, we give a higher probability of an unknown cell being free in the softmax function used for individual cell state prediction to encourage nodes to provide communications

to new frontiers.

3.4.1 U-Net Architecture

The U-Net Architecture [17] used for map prediction in this research is shown in Figure 3.4, based on the work in [18]. At each level of the encoder, two 3×3 convolutions are applied, each followed by a rectified linear unit (ReLU). The Rectified Linear Unit (ReLU) activation function, defined as $f(x) = \max(0, x)$, is computationally faster and has fewer vanishing gradients than the sigmoid and hyperbolic tangent activation functions, making it ideal for multilayer CNNs. At each downsampling iteration, we perform 2×2 max pooling with stride two, halving the image size and doubling the number of feature channels, with the exception of the first iteration, where we jump from the 3 channel RGB image we feed the network to 64 feature channels. The encoding step allows the network to learn what is in the image, but does not perform localization. The second half of the network, the decoder, applies 2×2 transposed convolutions to double the image size and halve the number of feature channels. In addition, on each level the cropped feature map from the parallel encoding level is concatenated with the decoding tensor to localize and two 3×3 convolutions, each followed by a ReLU, are performed.

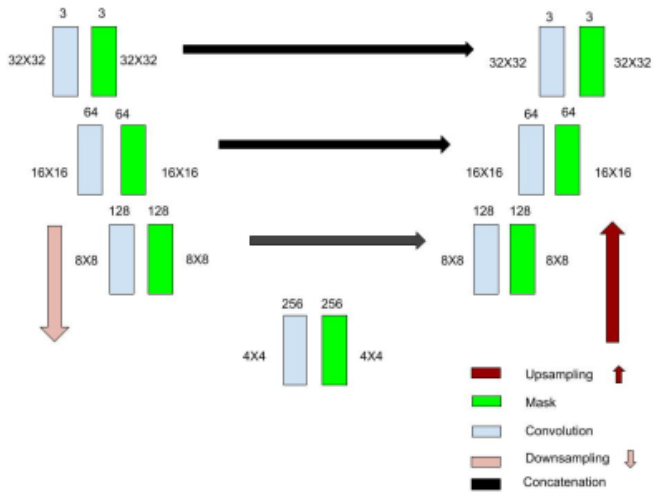


Figure 3.4: The U-Net Architecture for a 32x32 image, where the encoder (left) captures spatial information and the decoder (right) localises features, courtesy of [18].

Based on the work in [18], 80% of the map datasets input to the CNN are used for training, 10% are used for validation, and the other 10% are saved for testing. Each index of the dataset consists of three arrays that represent different input images: Ground Truth, Image, and Mask. Ground Truth is an

$x \times y$ size array where zeros represent occupied cells (walls) and ones represent free cells (traversable space). Image is a percentage cropped version of the free cells from the Ground Truth array, meaning a cropped percentage of the free cells in the Ground Truth array are kept constant and the rest of the cells in the array are set to zero, representing unexplored, or unknown, area. Our maps were randomly cropped percentages between 10% and 90%. Like in Section 2.2.2, Mask is equivalent to Image, with added ones representing the walls that surround the known free cells. Figure 3.5 shows a visual representation of the three arrays.

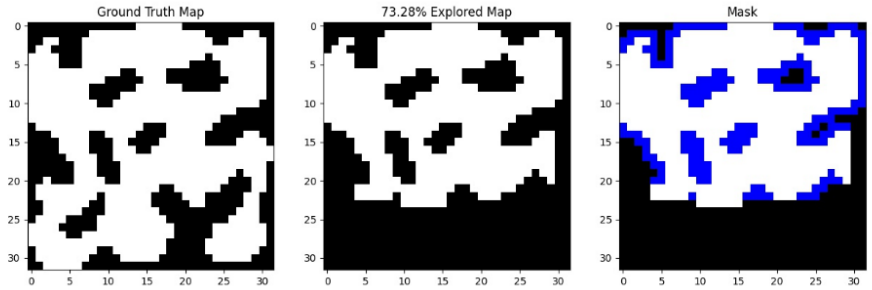


Figure 3.5: (Left to Right) Ground Truth, Image, and Mask visual representations of the input arrays to the network. For each map type, 40,000 of these were generated for training, validation, and testing. Ground Truth is the full gridworld array. Image is a percentage of neighboring free cells from Ground Truth with all other cells set to occupied. Mask is Image with occupied cells surrounding the Image free cells where occupied cells are located. This allows better prediction of frontiers, the border between known free cells and unknown cells.

3.4.2 Inpainting Loss

Working with 32×32 grid images, all losses, explained in Section 2.2.2, are calculated according to that size. Therefore, to calculate $\mathcal{L}_{perceptual}$, $N_{\Psi^{\mathrm{I}}_p g_t}$, the number of elements in $\Psi^{\mathrm{I}_{gt}}_p$, is 1,024. To calculate $\mathcal{L}_{tv}$, $N_{\mathbf{I}_{comp}}$, the number of elements in $\mathbf{I}_{comp}$, is also 1,024. Finally, H_p and W_p, the height and width of the environment used to calculate the normalization factor K_p, are both 32. For our purposes, the total loss $\mathcal{L}_{total}$, is defined as

$$\mathcal{L}_{\text{total}} = 120\mathcal{L}_{\text{valid}} + 20\mathcal{L}_{\text{hole}} + 0.05\mathcal{L}_{\text{perceptual}} + 120\left(\mathcal{L}_{\text{style}_{\text{out}}} + \mathcal{L}_{\text{style}_{\text{comp}}}\right) + 2\mathcal{L}_{tv},$$

where the constant coefficients for all losses were calculated by a hyperparameter search on 100 validation images in [14]. The weights for $\mathcal{L}_{valid}$, $\mathcal{L}_{hole}$, and $\mathcal{L}_{tv}$ were increased from those found in [14] to prevent overfitting the validation data to the training data for our map datasets.

3.5 Frontier Exploration

Team Explorer's ground and aerial vehicles seek to autonomously traverse the entire environment in the DARPA Subterranean Challenge. These agents have incomplete knowledge of their environments. Therefore, they must observe and learn the environment through exploration. For our purposes, the agent explores using the Yamauchi method [21], explained in Section 2.3. In our Yamauchi implementation, outlined in Algorithm 1, the A* path planning algorithm is used to find and guide the robot to its nearest frontier. All gridworld cells are checked for being frontiers in line 6 with isFrontier(). This method confirms that a cell is a potential frontier if it has not been visited by the agent, is a free cell, and neighbors a cell that has been visited. Dist() in line 10 calculates the length of an A* path between the agent location a and each potential frontier in F. Once a destination is reached, it is added to a list of visited frontiers. Our approach, specified in Section 3.6.2, deviates from the standard Yamauchi method when the agent needs to drop a node to maintain communications. When this occurs, the chosen node placement position is selected as the next Yamauchi frontier. Once this destination is reached and the node is dropped, Yamauchi Frontier Exploration proceeds as normal. Yamauchi Frontier Exploration is complete when the agent has visited all free cells of the ground truth map. For our purposes, if this occurs and not all the allotted nodes are dropped, the agent will calculate a greedy connected node placement and add it as a new frontier, even if the location has already been visited. This placement will increase communications coverage and allow for comparison of our approaches, as seen in Sections 4.3 and 4.4, since the number of nodes dropped in all tests will be the same.

Algorithm 1: Yamauchi Frontier Exploration

```
   Input: potential node location N, need to drop d, agent location a
(1) if d then
(2) |  new frontier f = N;
(3) else
(4) |  potential frontiers F = [];
(5) |  for each cell C in gridworld do
(6) |  |  if isFrontier(C) then
(7) |  |  |  add C to F;
(8) |  |  end
(9) |  end
(10) |  f = min(Dist(a, F));
(11) end
(12) return f
```

3.6 Node Placement Approaches

The node placement approaches are the four methods for placing nodes in all types of environments that we use for evaluation, namely the Prediction Ap-

proach, the pessimistic and optimistic Maximum Coverage (Without Prediction) Approaches, and the Naive Approach. The Greedy Connected Solution is the ideal layout of nodes if the agent were not constantly changing frontiers and moving due to Yamauchi Frontier Exploration. The approaches all use Yamauchi Frontier Exploration to best demonstrate Team Explorer's robot moving in a real environment. These approaches will be simulated and compared to each other and the Greedy Connected Solution in Section 4.

3.6.1 Greedy Connected Solution

The Greedy Connected Solution implements the greedy algorithm for maximum coverage, explained in Section 2.1.3, with adjustments for our research applications. This new version of greedy maximum coverage is called greedy connected coverage. Since we are placing communications nodes to make a connected MANET, each dropped node must be within communications range of another dropped node, creating a fully connected network, verified by Connected() in line 6 of Algorithm 2. All dropped node locations are stored in D, including the location of the basestation node, given by the initial agent location a. In the Greedy Connected Solution, the agent has full knowledge of the entire universe. The greedy connected coverage problem runs N iterations, where N is the number of nodes to be dropped. Each iteration, the node placement with the maximum coverage of an area without communications coverage is chosen to drop a node, if the condition is obeyed that this new node's communications range connects with that of an already placed node. The Greedy Connected Solution is reached when the final node is dropped.

Algorithm 2: Greedy Connected Algorithm

```
Input: ground truth map M, agent location a, number of nodes to
       drop N, node range R
(1)  number of nodes dropped n = 0;
(2) dropped node locations D = a;
(3) while n < N do
(4)     maximum area covered A = 0;
(5)     for each free location x in M do
(6)         if Connected(x, R, D) then
(7)             coverage C =
                 size S of newly covered area by node placed at x;
(8)             if C > A then
(9)                 A = C;
(10)                L = x;
(11)            end
(12)        end
(13)    end
(14)    Add L to D;
(15)    n += 1;
(16) end
(17) return D
```

3.6.2 Prediction Approach

The Prediction Approach is the foundation on which this research is based. It incorporates all the methods discussed in Sections 3.4 and 3.5, namely map prediction, sensor coverage, and frontier exploration. Specifically, as the agent explores the unknown environment using Yamauchi Frontier Exploration, it uses what it knows of the environment from exploration to predict the rest, demonstrated by line 5 of Algorithm 3. The observed map M is input to Yamauchi(), producing an updated M and agent location a from exploration. The more the agent explores, the more accurate the predicted maps will be to the ground truth maps. Implementing MapPrediction() on M, the agent creates a predicted map I on which to calculate sensor coverage. Every time the agent reaches a location that is on the edge of being out of communications of the established communications network, as shown by line 6, the agent uses GreedyConnectedCoverage() to solve for sensor coverage for each of its remaining available nodes. In addition, Dist() calculates the A* path length from a to each individual dropped node location in D. When the closest potential node location is found, the agent drops a node in this location with Drop(). The agent solves for potential node placements for all of its remaining nodes to give it options in the case that its nearest potential node placement is an obstacle due to an incorrect map prediction. In this work, the agent will not predict a potential node placement in an occupied cell, but moving forward this should be a con-

sideration. If this occurred, the agent would drop the node at the next nearest potential node placement. If there were no valid potential node placements, the agent would recalculate greedy connected coverage with its new knowledge of free and occupied cells. This ensures the agent can get a node placement solution as close to the Greedy Connected Solution as possible with minimal deviations from Yamauchi Frontier Exploration.

Algorithm 3: Prediction Algorithm

Input: observed map M, agent location a, number of nodes to drop N, node range R

```
(1)  number of nodes dropped n = 0;
(2)  dropped node locations D = a;
(3)  while n < N do
(4)      M, a = Yamauchi(M);
(5)      predicted map I = MapPrediction(M);
(6)      if min(Dist(a, D)) ≥ R then
(7)          potential locations to drop nodes L =
               GreedyConnectedCoverage(I, a, N − n, R);
(8)          closest location to drop a node L = min(Dist(a, L));
(9)          Drop(L);
(10)         Add L to D;
(11)         n += 1;
(12)     end
(13)   end
(14)   return D
```

3.6.3 Maximum Coverage (Without Prediction) Approach

To prove or disprove the value of using mpa prediction to determine potential node placements using greedy connected coverage, we are also testing approaches using the same Yamauchi Frontier Exploration and greedy connected coverage as the Prediction Approach, but without the prediction step. In the Prediction Approach, the universe in which to solve greedy connected coverage is all free cells the agent is aware of or has predicted. Without prediction there are two approaches, a pessimistic and an optimistic approach, known as No Prediction Pessimistic and No Prediction Optimistic, respectively. In No Prediction Pessimistic, the agent assumes the universe in which to solve greedy connected coverage is the free cells it has observed. We remove Line 6 from Algorithm 3 and set I to the known map M. Therefore, the universe the agent uses to calculate greedy connected coverage is only the currently observed areas, limiting its ability to provide communication to new frontiers. No Prediction Optimistic assumes all areas the agent has not visited are free, allowing it to calculate node placements anywhere in a universe of all free cells, except for the cells the agent knows are walls through observation. We remove Line 6 from Algorithm 3 and set I to a map of size M map's size, with only known obstacle points set to occupied. No Prediction Optimistic behaves as if the probabil-

ity associated with an area being free in the prediction step for the Prediction Approach is one.

3.6.4 Naive Approach

The Naive Approach directly implements Team Explorer's sensor placement model from the Tunnel and Urban Circuits, explained in Algorithm 4. Specifically, nodes are placed either once the robot reaches the edge of communications range R, calculated using Euclidean distance (Dist()), with any dropped nodes D or if the agent is no longer in line of sight with any dropped nodes and a threshold half of R from the nearest node is exceeded. This latter condition is abbreviated as LineOfSightInRange() in Algorithm 4 and has been used in testing due to Team Explorer's noisy communications model. These conditions are maintained in this work because it is representative of Team Explorer's real-world testing. We predict this approach is sub-optimal because it lacks foresight into node locations that will maximize coverage in the given environment. Using this approach, a node is as easily put in a secluded corner as it is put in a free hallway leading to many new communications frontiers.

Algorithm 4: Naive Algorithm

Input: observed map M, agent location a, number of nodes to drop N, node range R

number of nodes dropped $n = 0$;
dropped node locations $D = a$;
while $n < N$ **do**
    M, a = Yamauchi(M);
    if min(Dist(a,D)) $\geq R$ **or not** LineOfSightInRange(a, D) **then**
        Drop(a);
        Add a to D;
        n += 1;
    end
end
return D

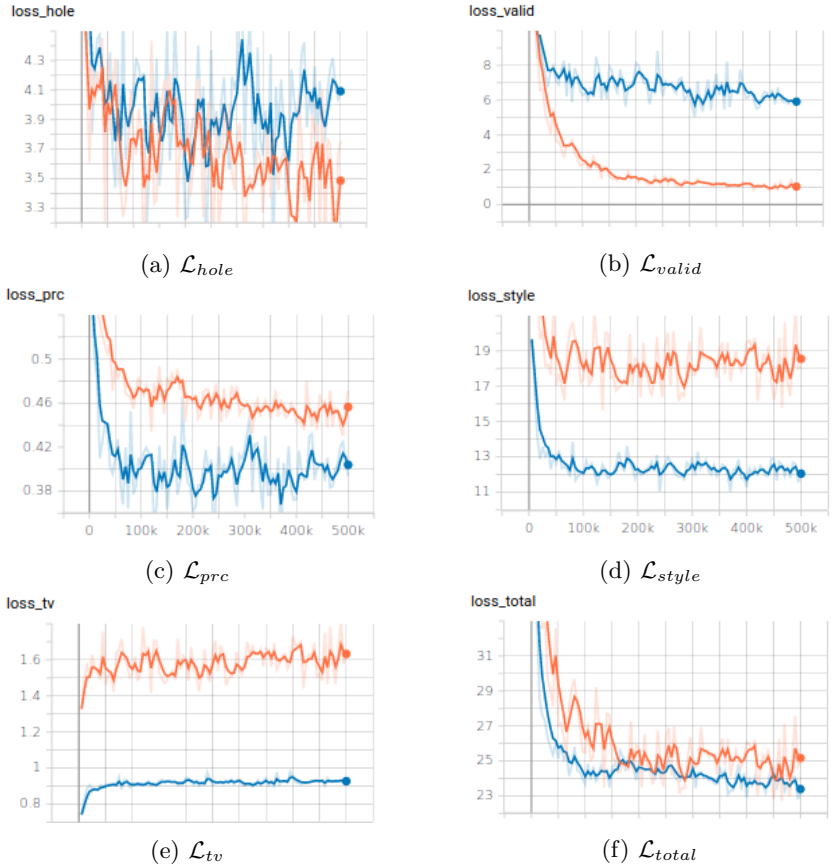
Chapter 4

Results

To fully understand the benefits and trade-offs of our approach, we developed simulation environments in which to test and evaluate all approaches. This section will present the simulation used to compare all approaches taken for node placement, namely Greedy Connected, Prediction, No Prediction Pessimistic, No Prediction Optimistic, and Naive. First, we will give an overview of the results of the map prediction training of the CNN. Second, we will cover the simulation and the different map types used for all approaches. Then, the Prediction Approach will be compared to both the No Prediction Pessimistic and the No Prediction Optimistic Approaches. This comparison is made to justify using map prediction moving forward in this research. Finally, we will evaluate the results from the Prediction Approach to those from the Naive Approach and Greedy Connected Solution.

4.1 Training & Validation Data Losses

Figure 4.1 shows the losses over 500,000 training iterations for the dataset of hybrid maps, that will be discussed in Section 4.2.1. These losses are described in detail in Section 2.2.2. All the losses, with the exception of $\mathcal{L}_{hole}$, plateau around iteration 100,000, showing successful training. $\mathcal{L}_{hole}$ appears to slightly overfit due to the validation data loss rising above the training data loss. After numerous trials with increased $\mathcal{L}_{hole}$ weights, the results shown with a weight of 20 were found to be the best possible. Since $\mathcal{L}_{hole}$ is the loss concerning per-pixel accuracy of predictions for unexplored regions, and it is being trained on highly variable environments, it is expected to be imperfect.



(a) $\mathcal{L}_{hole}$ (b) $\mathcal{L}_{valid}$

(c) $\mathcal{L}_{prc}$ (d) $\mathcal{L}_{style}$

(e) $\mathcal{L}_{tv}$ (f) $\mathcal{L}_{total}$

Figure 4.1: Losses for training data (orange) and validation data (blue) over 500,000 iterations.

4.2 Simulation

The simulation, inspired by [5], is run in Python using the Tkinter Python binding to the Tk GUI toolkit on an Intel Core i7-8750H CPU running the Ubuntu 18.04 operating system. A map is a 32×32 gridworld, where four different approaches to sensor placement are simulated to demonstrate the benefit of using map prediction and maximum coverage for sensor placement. The Prediction, Naive, No Prediction Pessimistic, and No Prediction Optimistic Approaches are compared to each other and the Greedy Connected Solution for dropping a set number of nodes in an unknown environment on three different types of maps representing cave, urban, and tunnel environments, as well as a combination of the three. For all simulation tests, the agent's initial location is randomly chosen from the list of free cells. These initial locations are kept the same when com-

paring corresponding maps with different approaches. A node is dropped at the initial location, represented by an orange cell, as shown in Figure 4.4. The node represents the basestation node and its communications coverage contributes to the total number of free cells covered, but does not count as one of the available nodes dropped. For all simulations, regardless of node placement method, the agent has five available nodes to drop. In a completely free map without any occupied cells, the maximum number of free cells the node placements could cover is 289 of the 1,024 free cells, including those covered by the initial node dropped at the basesation. Since our map types vary from mostly free to mostly occupied cells, we chose to test with dropping five nodes to display the agent's ability to choose placement positions wisely in both confined and free spaces with a number of nodes that will rarely fill every free cell in the gridworld. In all simulations implementing the greedy connected coverage problem, namely simulations implementing the Greedy Connected Solution and the Prediction, No Prediction Pessimistic, and No Prediction Optimistic Approaches, a set is a box grid subset of the gridworld of node range radius around a potential cell location of a node. Our node range is three cells, representing a 30 meter coverage radius in the real world as discussed in Section 3.2. Unless the range around the node location exceeds the width or height of the gridworld or it intersects with an occupied cell, each node radius set is a 7×7 box grid. The agent has an observation range of one cell block in all directions, representing a 10 meter observation radius. The agent has the same communications range as the nodes, in our case three cells or 30 meters. The maximum distance the robot can be from its nearest dropped node while remaining in communications range is six cells, since a node dropped at this location will allow both communications ranges to meet. In the results shown for the Prediction Approach for Sections 4.2.3, 4.3, and 4.4, all map predictions are made from networks trained on the same type of map the specific simulation is using. The rest of this section will explain the simulation environment of each approach for better understanding of the approaches and context leading into the results.

4.2.1 Maps

As shown by the examples in Figure 4.3, four types of 32×32 grid maps were used for simulation testing to mimic the environments seen in the DARPA Subterranean Challenge. The white cells represent free, traversable areas, while the blue and black cells are occupied and untraversable areas. The blue cells represent the beginning of walls, marking the transition from free cells to occupied cells. Like the point cloud shown in Figure 3.2, each cell in these gridworlds represents 10 m $\times$ 10 m in the real world. Therefore, the entire gridworld is 320m $\times$ 320m, which is within range of the sizes of all the map types we have seen thus far in the DARPA Subterranean Challenge, as shown by the approximately 90m $\times$ 90m Urban Circuit map in Figure 3.2 and the approximately 350m $\times$ 400m Tunnel Circuit map in Figure 4.2. Since we will be testing all environment types in the gridworld, we want to use a grid size and scale representative of all environments. To represent the environment in the Cave Circuit, maps like

Figure 4.3a were created. The multiple rooms of various sizes, tight freeings, and obstacles within large rooms represent natural cave structures and what is expected at the Cave Circuit. Figure 4.3b represents urban environments seen in the Urban Circuit. These environments are much more free with obstacles dispersed throughout representing debris, pillars, or large obstructions. Finally, Figure 4.3c represents tunnel environments seen in the Tunnel Circuit. The narrow passages and many branches are representative of the mine environments we competed in for the Tunnel Circuit. Finally, the DARPA Subterranean Grand Challenge will incorporate all three of these environments into one hybrid environment. Figure 4.3d is a combination of the three environments, where the first third of the total columns represents a cave environment, the next third an urban environment, and the final third a tunnel environment. For each type of map shown, 32,000 different variations were produced for training and 4,000 were produced for validation.

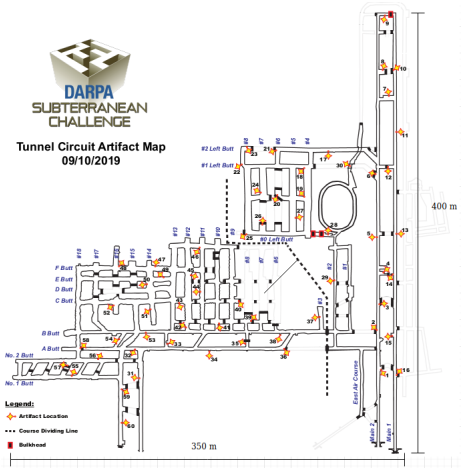
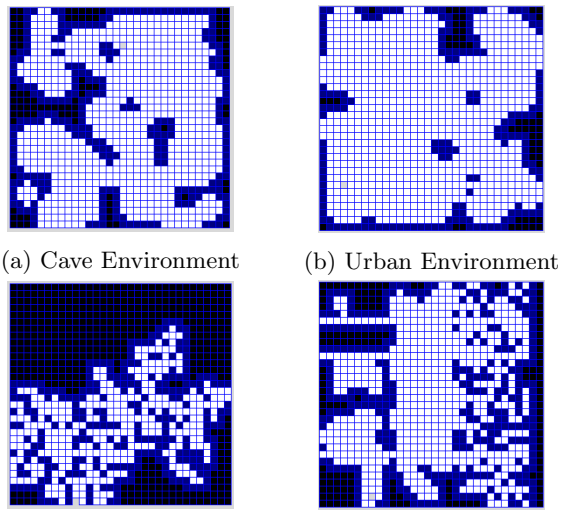


Figure 4.2: Map of Bruceton Mine in Bruceton, Pennsylvania from the DARPA Subterranean Challenge Tunnel Circuit in Summer 2019.



(a) Cave Environment (b) Urban Environment

(c) Tunnel Environment (d) Hybrid Environment

Figure 4.3: The four types of maps used for simulation testing.

4.2.2 Greedy Connected Solution (Gridworld)

Implementing the coverage method explained in Section 3.6.1, the Greedy Connected Solution is used as a metric of comparison to the simulated node dropping approaches. Figure 4.4 demonstrates the output of dropping five nodes in a simulated cave environment using the Greedy Connected Solution. A node placement is identified by a maroon cell and the yellow cells represent areas under communications coverage.

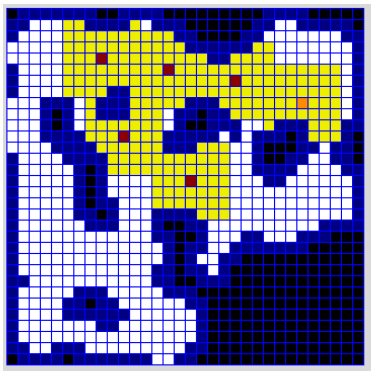
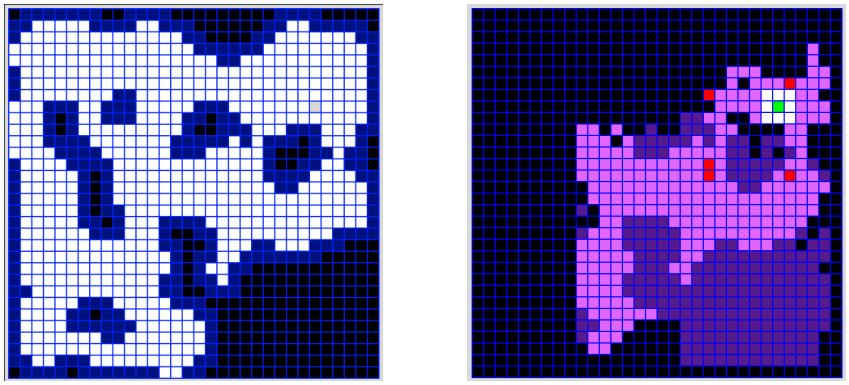


Figure 4.4: Node placement using the Greedy Connected Solution on a cave map.

4.2.3 Prediction Approach (Gridworld)

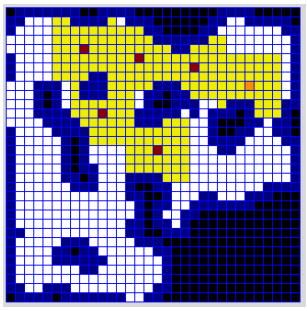
In simulation, the Prediction Approach executes the approach explained in Section 3.6.2. This approach begins with the agent, represented by the green cell in Figure 4.5b, having limited knowledge of the rest of the gridworld. A ground truth gridworld of a cave map is shown in Figure 4.5a. In Figure 4.5b, the white cells surrounding the agent represent the agent's known free area of the map. Both the light and dark purple cells represent the agent's prediction of the rest of the free map based on its known map. The dark purple cells are equivalent to the light purple cells according to the agent, but are dark purple to allow the viewer to be aware of the agent's incorrect prediction because these cells are truly occupied. The prediction in Figure 4.5b is initially mostly incorrect because the known map is only the 3×3 grid at the initial agent location. As the agent learns more of the gridworld, its predictions improve. Once initialized, a node is dropped at the initial location of the agent, representing the communications node that would be located at the basestation. Then, the agent solves the greedy connected coverage problem, outlined in Section 3.6.1, to produce the set number of possible node placements, in our case five, as shown by the five red cells in Figure 4.5b. Only node placements within communications range of already placed nodes are considered. This ensures the agent will always be able to communicate back to the basestation. Since the first node placement predictions are within range of the basestation node, they will appear sub-optimal. In fact, these predictions are a result of greedy connected coverage for the predicted map, but there are more nodes predicted than necessary at this point in the exploration since all nodes must be within range of the basestation node. This phenomenon leads to potential node placements next to each other, as seen with the centermost potential node placements in Figure 4.5b. As more nodes are placed, these placement predictions will spread out to encourage nodes as far apart as possible while maintaining a chain of communication.



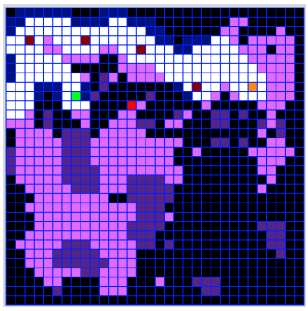
(a) Ground Truth (b) Prediction (Initial)

Figure 4.5: Initial stage of the simulation of Prediction Approach on a cave map.

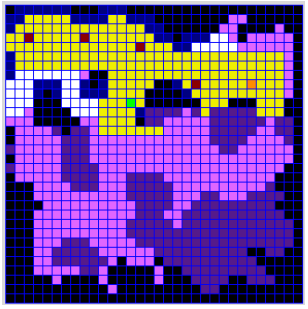
After the first set coverage problem is calculated, the agent begins traversing the gridworld according to the Yamauchi method. The agent predicts the unknown environment every step. Once the agent reaches a cell at the maximum distance from its nearest dropped node, it will recalculate greedy connected coverage to determine its maximum coverage placements with the updated predicted map. The agent will then set a new destination at the greedy connected coverage node location calculated nearest to it. Once this location is reached, the agent will resume standard Yamauchi Frontier Exploration. Figure 4.6b demonstrates the agent approaching its final node to drop in the simulation, represented by the red cell, where maroon cells represent nodes that have already been dropped. When the agent drops its final node, the simulation will end. The cell containing the final node will not appear maroon because the agent, indicated by the green cell, is located on top of it, but the node is there all the same. The final state is shown in Figure 4.6c, where the yellow cells indicate communications coverage. As can be seen, the number of cells within communications range in the Greedy Connected Solution and the Prediction Approach are close at 223 cells covered and 192 cells covered, respectively. Section 4.3 will further elaborate on this performance for all tested maps.



(a) Greedy Connected Solution



(b) Prediction (Mid-Run)

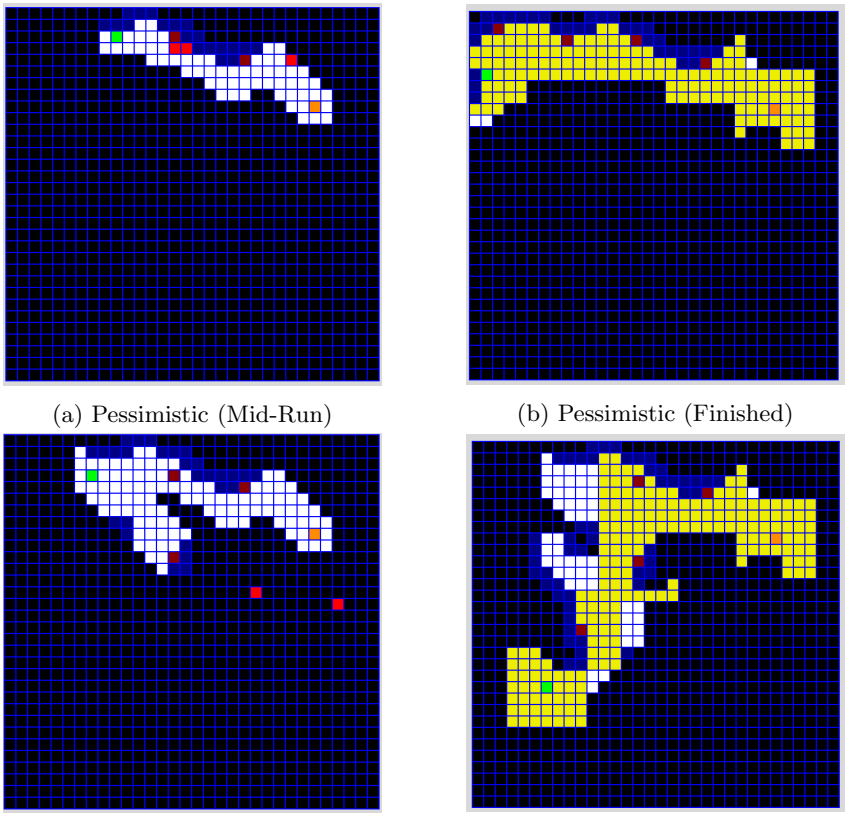


(c) Prediction (Finished)

Figure 4.6: Comparison of the Greedy Connected Solution to Prediction Approach on a cave map.

4.2.4 Maximum Coverage (Without Prediction) Approach (Gridworld)

These simulations follow the approaches explained in Section 3.6.3. As seen in 4.8a, the No Prediction Pessimistic Approach leads to less exploration and nodes placed based solely on the frontiers generated by Yamauchi Frontier Exploration. At the end of this simulation, the No Prediction Pessimistic Approach dropped nodes that covered only 145 cells. In addition, Figure 4.8b demonstrates these nodes are sub-optimally placed against walls rather than in open space for maximum coverage, a product of Yamauchi Frontier Exploration initially exploring along walls. Since the No Prediction Optimistic Approach assumes all cells in the gridworld are free, except those it has observed to be occupied, it has a much larger universe of free cells in which to calculate greedy connected coverage. This approach performs significantly better than the pessimistic approach, covering 195 cells in communications range.



(a) Pessimistic (Mid-Run)

(b) Pessimistic (Finished)

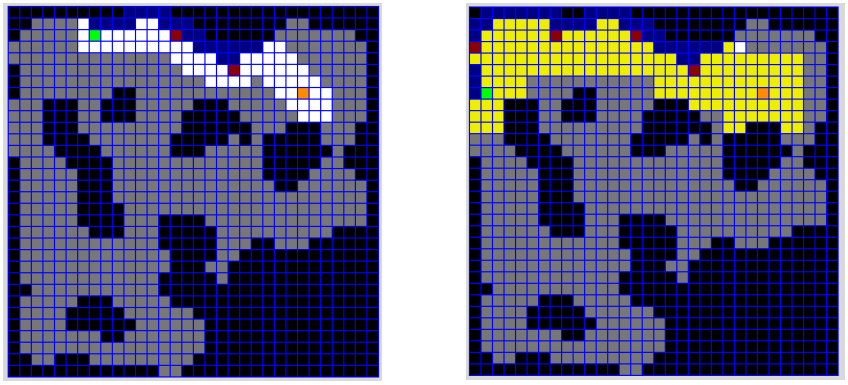
(c) Optimistic (Mid-Run)

(d) Optimistic (Finished)

Figure 4.7: Simulation of both Maximum Coverage (Without Prediction) Approaches in a cave environment.

4.2.5 Naive Approach (Gridworld)

As elaborated upon in Section 3.6.4, the Naive Approach executes Team Explorer's node placement model used in the Tunnel and Urban Circuits. Specifically, nodes are placed either once the agent reaches the edge of communications range, six cells between itself and the nearest dropped node, or if the agent is no longer in line of sight with any dropped nodes and a minimized threshold of three cells between itself and the nearest dropped node is reached. This minimized threshold mimics Team Explorer's current approach of allowing nodes to be dropped out of line of sight as long as a minimum distance, and thus node range, is not exceeded. As shown in Figures 4.8a and 4.8b, the agent lacks foresight into the layout of the entire map, where gray and black cells represent unknown areas. This leads to nodes being dropped in sub-optimal places for coverage, near walls and dead-ends, because the agent does not deviate from Yamauchi Frontier Exploration to drop in more free areas. In this example, the Naive Approach placed nodes to cover 157 free cells, 18% less coverage than the Prediction Approach.



(a) Naive (Mid-Run)

(b) Naive (Finished)

Figure 4.8: Simulation of the Naive Approach in a cave environment.

4.3 Prediction vs. No Prediction Simulations

The metrics we will be using to determine which approach is most useful for node placement are the percentage of cells in the gridworld covered by communications after the final (fifth) node is dropped, the average distance between the corresponding nodes that are dropped by the given approach and the Greedy Connected Solution, the percentage of the free cells in the map explored, the average time of the simulation, and the average node distance between the simulated approach and the Greedy Connected Solution, defined by

$$\frac{\sum_{n=1}^{N} \sqrt{(x_{S_n} - x_{G_n})^2 + (y_{S_n} - y_{G_n})^2}}{N}.$$

In the above equation, N is the number of nodes to be dropped, n is the specific node being dropped, x_{S_n} and y_{S_n} are the x and y locations of the nth node dropped using the simulated approach, and x_{G_n} and y_{G_n} are the x and y locations of the nth node dropped in the Greedy Connected Solution. The results in Table 4.1 come from running simulations on 100 maps from each type of environment, cave, urban, tunnel, and hybrid, on the three node placement approaches: No Prediction Pessimistic (No Pred Pess), No Prediction Optimistic (No Pred Opt), and Prediction (Pred). When compared to No Prediction Pessimistic, both No Prediction Optimistic and Prediction perform significantly better in terms of communications coverage, node dropping accuracy to the Greedy Connected Solution, and amount of the free area in the map explored. This is at the expense of having a longer path for the agent to drop all of its nodes, since it will travel to new frontiers to place nodes in the No Prediction Optimistic and Prediction Approaches. In addition, simulation times, the time from the agent's initialization in the gridworld to dropping its final node, are significantly longer for the No Prediction Optimistic and Prediction Approaches due to the increased exploration, which allows for more coverage. The map prediction step for the Prediction Approach gives it the highest run-times of the three approaches. Even still, the Prediction Approach's ability to have higher coverage, lower path length, and more accurate node placement on three out of the four map types for each metric demonstrates the value of map prediction when determining where to place nodes in unknown areas. As stated in Section 3.6.3, the No Prediction Optimistic Approach behaves like a highly optimistic version of the Prediction Approach, where all cells the agent has not visited are considered free. Therefore, the approach has the freedom to use all cells in the gridworld to calculate greedy connected coverage, with the exception of known occupied cells. Even still, taking the time to do more accurate map prediction is worthwhile in preserving agent movement, approaching Greedy Connected Solution placement accuracy, and maximizing communications coverage.

	Cave			Urban		
	No Pred Pess	No Pred Opt	Pred	No Pred Pess	No Pred Opt	Pred
Average % Covered	31	38	**39**	20	**30**	**30**
Average Path Length	**124.3**	151.10	157.19	**101.13**	144.34	139.02
Average Node Distance	9.71	8.69	**8.27**	11.85	9.25	**9.07**
Average % Explored	30	**43**	42	17	**32**	**32**
Average Time (s)	**39.59**	196.48	275.74	**31.73**	200.43	392.0

(a) Cave and Urban Data

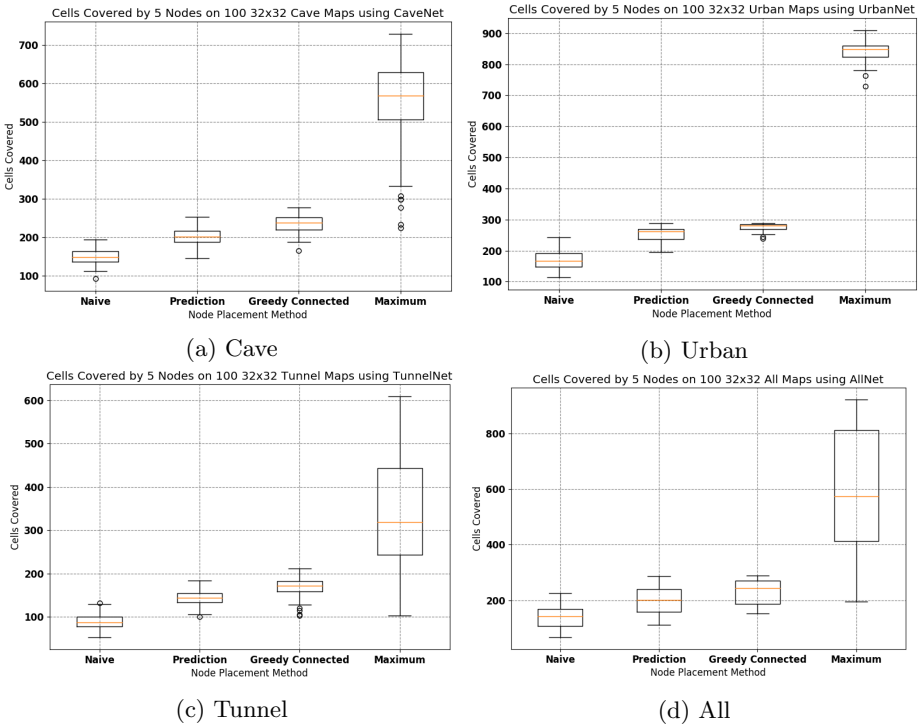
	Tunnel			Hybrid		
	No Pred Pess	No Pred Opt	Pred	No Pred Pess	No Pred Opt	Pred
Average % Covered	43	48	**49**	26	32	**34**
Average Path Length	**129.57**	164.35	142.09	**117.48**	151.31	138.60
Average Node Distance	8.54	**7.74**	7.86	9.17	7.60	**7.37**
Average % Explored	46	**61**	51	24	**38**	36
Average Time (s)	**48.10**	269.11	301.65	**35.96**	185.99	272.21

(b) Tunnel and Hybrid Data

Table 4.1: Prediction and No Prediction (Pessimistic and Optimistic) comparison results, demonstrating the value of predictions that favor keeping unknown areas free for potential node placements.

4.4 Prediction vs. Naive Simulations

The following results come from running the Naive and Prediction Approaches and the Greedy Connected Solution on 100 maps from each of the three map types, cave, urban, and tunnel, as well as a combination of the three, referred to as "All". Figures 4.9 and 4.10 demonstrate the significant increase in cell coverage when using the Prediction Approach over the Naive Approach. This is expected, given the Prediction Approach's goal of placing nodes in free areas, whereas the Naive Approach drops a node as soon as the edge of communications range with a dropped node is reached.



(a) Cave

(b) Urban

(c) Tunnel

(d) All

Figure 4.9: Results in the node dropping simulation showing the number of free cells covered in communications range after all available nodes are dropped on 100 maps of each map type. The Maximum data is the total number of free cells available to put in communications range in the map.

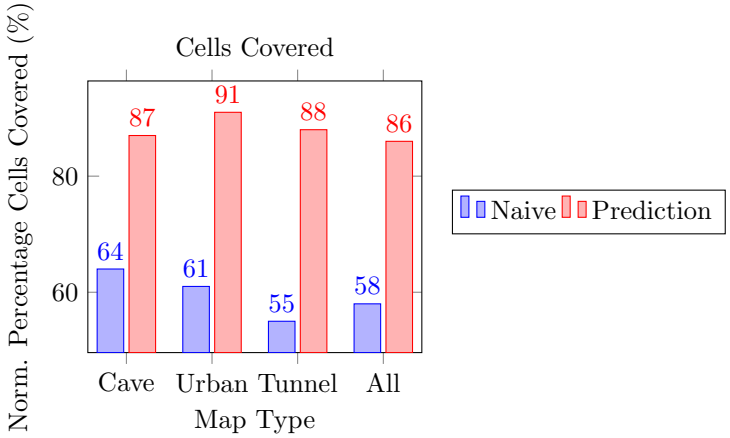


Figure 4.10: Results in the node dropping simulation showing the percentage of free cells covered in communications range normalized over the percentage of free cells covered by the Greedy Connected Solution over 100 maps of each map type.

Added benefits of the Prediction Approach are its ability to explore more of the total environment and place nodes more accurately to the positions calculated using the Greedy Connected Solution. Figure 4.11 shows that in every environment, the Prediction Approach explores nearly, if not more than, double the percent of free cells explored by the Naive Approach. This behavior will benefit Team Explorer as they seek to explore as much of the test environment as possible with as few nodes dropped as possible.

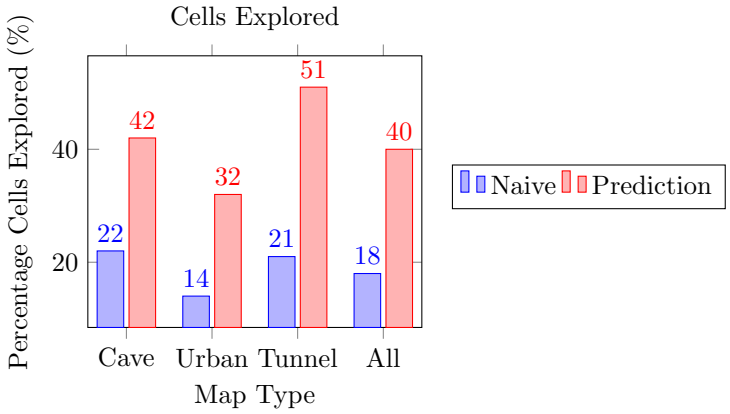


Figure 4.11: Results in the node dropping simulation showing the percentage of free cells explored using the Naive Approach and the Prediction Approach on 100 maps of each map type.

Figure 4.12 shows the average node distance between nodes at the same order in the dropping queue from the Prediction and Naive Approaches compared to the Greedy Connected Solution. Therefore, the distance is an average over the 100 map simulation tests run of the average nodes location distances. When compared to the Naive Approach, the Prediction Approach is at most a 40% decrease in node distance from the Greedy Connected Solution, shown in the urban maps, and at least a 31% decrease, shown in the tunnel maps.

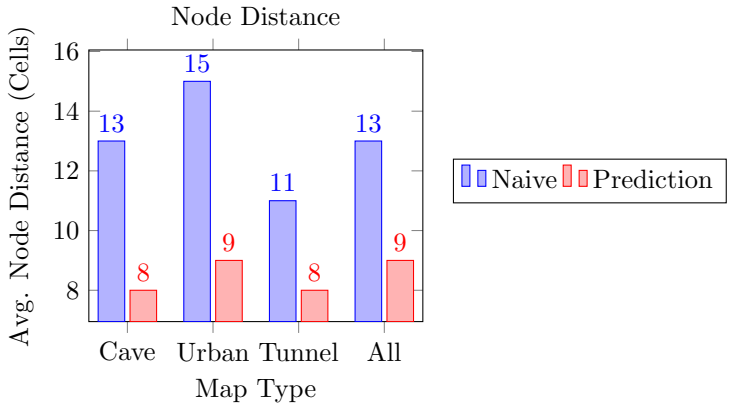


Figure 4.12: Results in the node dropping simulation showing the average node distance to the Greedy Connected Solution from nodes dropped using the Naive Approach and the Prediction Approach on 100 maps of each map type.

Figure 4.13 demonstrates the large discrepancy in simulation run-times between the Naive and Prediction Approaches. The Prediction Approach's longer run-time can be attributed to its increased exploration as well as the time to run map prediction and the greedy connected coverage problem. This metric is one of the main trade-offs that must be considered when using this method. Measures were taken to decrease this run-time, such as decreasing the number of times the greedy connected coverage problem was solved and only performing map prediction every three steps the agent took, but more measures need to be taken if faster times are desired.

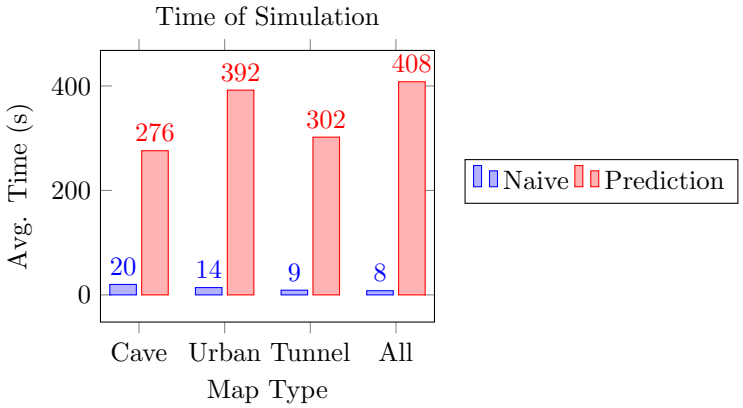


Figure 4.13: Results in the node dropping simulation showing the average simulation time from the Naive and Prediction Approaches on 100 maps of each map type.

4.5 Cell Coverage Comparison

CNNs were trained on all map types separately as well as a combination of the three, resulting in four networks used for simulation: CaveNet, UrbanNet, TunnelNet, and AllNet. All network names correspond to the map types they were trained on, with AllNet being trained on a combination of cave, urban, and tunnel maps. The average percentages of free cells covered in communications range normalized over the Greedy Connected Solution for all combinations of map type and network, simulated on 100 maps, are shown in Table 4.2. It was expected that map types run in simulation using their corresponding network would result in better coverage due to better map predictions. While this behavior is true for most map types and their respective networks, the increase is not significant. For example, the Prediction Approach with the urban maps tend to always have high normalized coverage, regardless of the network used. This is most likely due to its free nature, in which nodes placed are more likely to cover more cells. As stated in Section 3.4, we give a higher probability of an unknown cell being free in the softmax function used for individual cell state prediction to encourage nodes to provide communications to new frontiers. This map prediction behavior explains why the No Prediction Optimistic and Prediction Approaches behaved similarly on urban maps, shown in Table 4.1a. No Prediction Optimistic gives all unknown cells 100% probability of being free.

	CaveNet	**UrbanNet**	**TunnelNet**	**AllNet**
Cave	87%	86%	83%	83%
Urban	91%	91%	85%	91%
Tunnel	85%	87%	88%	85%
All	86%	87%	85%	86%

Table 4.2: Results in the node dropping simulation showing the average percentage of free cells covered in communications range normalized over the percentage of free cells covered by the Greedy Connected Solution for 100 maps. This data is from simulations on every combination of map input type and network trained on all map types.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

The results of our simulations demonstrate the significant improvement in performance of using map prediction and greedy connected coverage with the Prediction Approach to determine node placement, as opposed to Team Explorer's current approach, the Naive Approach. Map prediction allows the agent to have a larger universe of free cells from which to determine node placements and greedy connected coverage uses this larger universe to maximize communications coverage while maintaining connection in polynomial time. In the Prediction Approach, we found a significant benefit from greedy connected coverage and Yamauchi Frontier Exploration working together to increase agent exploration while remaining in communications range with a limited number of nodes. The agent's deviation from the generated Yamauchi frontiers to pursue a node placement when necessary encourages new frontiers to be covered with communications so that the agent can keep exploring. In addition, the Prediction Approach helped us gain a better understanding of how map prediction can be used to improve communications coverage. While predictions favored predicting unknown cells as free similar to the fully optimistic approach seen in the No Prediction Optimistic Approach, predictions positively impacted maximization of cell coverage and minimization of the agent path length and node distance from the Greedy Connected Solution.

5.2 Future Work

Based on the close results from the No Prediction Optimistic and Prediction Approaches in Section 4.3 it would be interesting to see how the Prediction Approach improves with a tighter softmax function when predicting which cells are

free and occupied. This may result in more accurate predictions to the ground truth or clustered node placements due to the agent predicting more occupied cells than necessary. In addition, improving computation time of map prediction and the greedy connected coverage problem is essential to implementing the Prediction Approach in the real-world. Currently the greedy connected coverage problem is implementing A* for all potential node placements to ensure there is a communications path between the potential placement and a cell that is known to be in communications coverage. This process adds time, but is necessary to ensure there is not a wall completely blocking the coverage between a node and its connection to the entire network.

Future work that would also benefit this research is answering how to approach this problem with unknown total map size or a map that grows in total size with exploration. For our simulation, we maintain a 32×32 gridworld representing a 320m$\times$320m world because it is representative of the environments Team Explorer has worked in thus far. Even still, caves, urban environments, and tunnels in the real world can be much larger and grow to become larger than expected when the full size is not known during exploration. A possible solution worth exploring to solve node placement with a map of unknown size is to use a graph to represent the world, rather than the array data structure that is currently used. In this way, every cell in the gridworld would be represented by a node in the graph with edges connected between nodes. With the exception of the edges, each node would be independent of the others. Training would behave in the same way by calculating the probability one node is free given its neighbor is free, adjusting losses, such as total variation loss $\mathcal{L}_{tv}$ to encourage smoothness between neighboring nodes. Using a graph instead of an array allows scaling as the agent discovers more of the map or determines the map is larger than originally expected. In addition, using a graph allows for a naturally decentralized method of performing computation, posed as a message passing problem, to allow for adding and subtracting states. We started looking into factor graphs in large scale simultaneous localization and mapping (SLAM) [4], which would allow for state space changes. While this could be of use, factor graphs increase complexity by splitting each cell of the gridworld into a factor node and variable node. While the run-time of solving greedy connected coverage increases with the gridworld size, the only part of the Prediction Approach that would currently be impeded by variable gridworld size is map prediction, since the CNN is trained on 32×32 gridworlds. We need to evaluate the trade-offs between using factor graphs and training on maps of variable size.

Appendices

Appendix A

Hardware

	UAP-AC-M	DX2 (5 GHz)
Dimensions (mm)	$353 \times 46 \times 34.4$	$108 \times 43 \times 40$
Weight (g)	152	123
Power Supply (VDC)	24	8-60
Max Power Consumption (W)	8.5	7.5
Max TX Power (dBm)	20	27 ± 2
Max Range* (meters)	183	5,250

Table A.1: Ubiquiti and Rajant Radio specifications. *Max Range is for optimal environments.

Bibliography

[1] J. A. Caley, N. R. J. Lawrance, and G. A. Hollinger. Deep learning of structured environments for robot search. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3987–3992, 2016.

[2] Gerard Cornuejols, Marshall L. Fisher, and George L. Nemhauser. Exceptional paper—location of bank accounts to optimize float: An analytic study of exact and approximate algorithms. *Management Science*, 23(8):789–810, 1977.

[3] DARPA. Darpa subterranean (subt) challenge, 2019.

[4] Frank Dellaert and Michael Kaess. Factor graphs for robot perception. *Foundations and Trends® in Robotics*, 6(1-2):1–139, August 2017.

[5] Sahib Dhanjal. Path-planning-simulator, Feb 2018.

[6] Maximilian Ernestus, Stephan Friedrichs, Michael Hemmer, Jan Kokemüller, Alexander Kröller, Mahdi Moeini, and Christiane Schmidt. Algorithms for art gallery illumination. *CoRR*, abs/1410.5952, 2014.

[7] Uriel Feige. A threshold of ln n for approximating set cover. *JOURNAL OF THE ACM*, 45:314–318, 1998.

[8] Freifunk. Open-mesh:b.a.t.m.a.n, 2020.

[9] Martin Gardner. Mathematical games-the fantastic combinations of john conway's new solitaire game, life, 1970. *Scientific American, October*, pages 120–123.

[10] Rajant Inc. Rajant breadcrumb dx2, 2020.

[11] Ubiquiti Inc. Unifi mesh access point, 2020.

[12] Richard Karp. Reducibility among combinatorial problems. volume 40, pages 85–103, 01 1972.

[13] G. D. Kazazakis and A. A. Argyros. Fast positioning of limited-visibility guards for the inspection of 2d workspaces. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2843–2848 vol.3, Sep. 2002.

[14] Guilin Liu, Fitsum A. Reda, Kevin J. Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions, 2018.

[15] Anshika Pal, Ritu Tiwari, and A. Shukla. Multi robot exploration using a modified a* algorithm. pages 506–516, 04 2011.

[16] Charles Richter and William Vega-Brown. *Bayesian Learning for Safe High-Speed Navigation in Unknown Environments*, pages 325–341. 01 2018.

[17] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

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

[19] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

[20] Vijay Vazirani. *Approximation Algorithms*. 01 2001.

[21] B. 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, 1997.

[22] M.H. [Fazel Zarandi], S. Davari, and S.A. [Haddad Sisakht]. The large scale maximal covering location problem. *Scientia Iranica*, 18(6):1564 – 1570, 2011.