Using Contrastive Learning for Map Prediction in 3D Environments via Trajectory Map Pretraining

by

Siddarth Narasimhan

Supervisor: Prof. Goldie Nejat

April 2023

B.A.Sc. Thesis

Division of Engineering Science

UNIVERSITY OF TORONTO

Abstract

Autonomous mobile robot teams are often tasked with high-level problems such as exploration, navigation and mapping in real-world environments, while having limited knowledge about the environment prior to deployment. Map inference is an emerging field of research that aims to improve the efficiency of these tasks by providing an accurate model of the unknown environment, thus enabling robots to predict beyond their sensor horizon and make more informed decisions during execution. This thesis paper presents a novel deep learning architecture for map inference in a multi-robot system. We introduce a method called Contrastive Trajectory-Map Pretraining to outperform existing baseline solutions in map inference, while using a data-efficient approach to leverage information from robots in the environment. Our method is evaluated against baseline solutions, with an average of 50% boost in both information gain and accuracy. Our solution shows great potential for use in complex high-level cooperative tasks with large robot teams.

Acknowledgements

This thesis would not have been possible without the guidance and assistance of many people. Firstly, I would like to thank Professor Nejat for giving me this opportunity to join the ASB Lab research group, and for the amazing support throughout my thesis. Your expertise and feedback had brought my work to a higher level.

Secondly, I would like to express my sincere gratitude to my PhD supervisor, Aaron Tan, for his invaluable support and guidance throughout my research journey. Your knowledge, patience, and dedication have been instrumental in shaping the direction of my work, and your insightful feedback has greatly helped improve the quality of my research. I would also like to thank the ASB Lab at the University of Toronto for providing me with the resources needed to pursue this project.

Finally, I would like to thank my friends and family for their encouragement throughout the semester.

Contents

List of Figures

Chapter 1

Introduction

The mobile robot environment inference problem describes the prediction of unexplored regions of an environment using observations from explored regions of the same environment [3]. Mobile robot teams are often required to navigate and explore in real-world environments, from indoor spaces to disaster scenes [4], where information about the environment is typically unknown prior to deployment. Each robot is generally equipped with a range sensor that allow them to map features in the surroundings and generate feasible trajectories for navigation. However, these techniques rely on direct observations of the robot and do not consider environment features beyond the sensor horizon [5], which can be useful for path-planning (e.g., anticipating upcoming walls or corners). A multi-agent system would benefit from being able to infer about the unknown environment to increase spatial awareness, predict future states of teammates and make more informed path-planning decisions [6].

Map inference in 2D can be divided into database [7], representative lines [8], and data-driven approaches [3, 9, 10]. Although database and representative lines approaches are shown to perform accurately, they are only effective in repetitive environments. Data driven approaches, however, extract contours for prediction which make them a more practical solution [11]. These are either based on autoencoders (AE) [3], generative adversarial networks (GANs) [9], or low-rank matrix completion (LRMC) [10]. GANs tend to perform well but are unreasonably difficult to train while LMRC methods rely on matrix rank assumptions and do not scale well for large environments. AE-based methods are the state-of-the-art for map inference, but often result in noisy reconstructions and short-ranged predictions. To date, there has not been any extensive research that focus on map inference in 3D environments with uneven terrain and irregular obstacles, nor map prediction in a multi-robot cooperative system.

The main objective of my thesis is to design a deep learning-based pipeline involving *contrastive learning* (CL) to improve the performance of map prediction in 3D unstructured environments with multiple robots. The goal of this pipeline is to improve map reconstruction accuracy, information gain and increase prediction depth in a multi-robot scenario, three issues which the current AE-based methods suffer from. Specifically, the contrastive learning model will learn to associate robot trajectories with partially complete heightmaps, and will be augmented to the existing baseline solution. By incorporating trajectory data of the robots into the prediction step, the proposed solution will be able to better infer the shape of surrounding obstacles, resulting in increased spatial awareness and more accurate predictions [6].

This document is organized such that Chapter 2 will explore existing literature in map inference and contrastive learning. Chapter 3 will focus on formulating the problem. Chapter 4 will detail the deep learning pipeline. Chapter 5 shows the results of the pipeline. Finally, Chapter 6 discusses limitations and future direction for the project.

Chapter 2

Literature Review

2.1 Map Inference

Map inference can be classified into local [5, 6, 12, 13] and global methods [3, 7, 8, 9, 10, 11]. At the local level, map inference is used to predict incomplete or missing information in the immediate vicinity of the robot [6, 13]. This is often used for semantic mapping, navigation in the local neighbourhood of the robot and predicting the spatial arrangement of obstacles in successive regions of exploration [6, 12]. At the global level, predictions are made over unexplored regions within the entire map [9]. Global inference strategies provide more spatial information about the unknown environment and can be used to influence long-term decisions of the robot in exploration and navigation tasks [3]. Our focus will be on global inference strategies since they perform similar tasks as the local level, but are more informative and often outperform their local method counterpart. Global methods can be categorized into map database [7, 14], representative lines [8, 15] and data-driven approaches [3, 9, 10, 11, 16].

2.1.1 Map Database Approaches

Inference using a database involves specifying an unknown region to predict over, obtaining its visual features using sensor information, and then searching through a database of maps that have the largest probability of fitting the area of interest [7]. This approach is most effective in scenarios where repetitive geometric features are present throughout the environment such as boxed rooms and passages. Additionally, it is assumed that the database contains a sufficient number of maps that are similar in nature to the environment being explored. These methods are shown to be more robust than traditional approaches. In [7], breadth-first-search was applied to the predicted map at unexplored points to identify locations that lead to loop closures, thus helping reduce overall state uncertainty. This was compared to the nearest frontier approach, which was found to be less effective since it consistently resulted in sensor drift and inaccurate maps. Another example is [14], where map inference is achieved in conjunction with GMapping's SLAM and a gaussian filter is used to inform frontier points that have the highest information gain. This method had achieved multi-robot exploration more efficiently than vision-based approaches, which were more computationally expensive.

2.1.2 Representative Lines

The representative lines approach is effective in environments with rectilinear walls, hence, is appropriate for floorplans, office buildings or mazes [15]. The method infers the geometry of an unknown region by comparing against adjacent explored regions and extrapolating them outwards to perform the inference. This approach was used in structured indoor environments in [8] to choose frontier points and evaluate their information gain for exploration, and was found to complete exploration faster than traditional frontier-based approaches.

2.1.3 Data-Driven Approaches

Data-driven approaches make use of a representative dataset to learn the structure of environments offline before execution. Unlike map database and representative lines approaches, they do not require the environments to be structured or repetitive, which make them the most appropriate choice for real-world scenarios. The majority of these techniques are based on deep learning, which use an autoencoder [3, 11, 16] or generative adversarial network (GAN) [9], to provide reconstructions of partially complete maps. In the case of autoencoders, the U-Net architecture is popular [17], which uses encoder network to downsample capture spatial information about the environment and a mirrored decoder network to predict unknown regions in the input. Other popular forms of this architecture include the use of variational autoencoders [3] or convolutional autoencoders with skip connections [16]. Another technique in data-driven approaches is the use of low-rank-matrix completion (LMRC) [10]. The map of the environment is represented as an occupancy grid matrix and incomplete entries can be determined by solving a non-convex optimization problem using singular value decomposition. Although LMRC requires less data to train, they operate based on rank assumptions of the grid and are not generalizable to all types of maps [10]. Deep learning methods have more flexibility in environment choice and have been applied to various unstructured environments, while significantly outperforming several other methods in exploration time and travel distance [18, 19]. However, deep learning requires that you have access to a large dataset that you can use to train the model on your environments.

2.2 Contrastive Learning

Contrastive Learning (CL) is a machine learning technique that has gained recent interest due to its performance in self-supervised learning tasks. CL transforms each data point into an embedding space, where samples are contrasted against each other to equate those that have similar features and reject those that do not [20]. This has been applied to a variety of classification [1] and trajectory prediction problems [21, 22] with success.

2.2.1 Image to Text Association

OpenAI's Contrastive Language-Image Pre-training (CLIP) learns the relationship between images and the text they describe [1]. Given a short sentence, the model is able to retrieve a set of images in an inventory that best fit the description. Each input text and image are converted to embedding space and an inner-product is computed for each entry. During training, the goal is to maximize the entries on the diagonal of the inner product matrix and minimize the remaining entries, thus, matching each text with its respective image. Across a sample 27 datasets, zero-shot CLIP outperforms ResNet-50 features on 16 datasets, including ImageNet. However, some major limitations of CLIP include significant compute capability required for training, struggling with fine-grained classification tasks (such as differentiating car models) and poor generalization to "out-of-distribution" data [1].

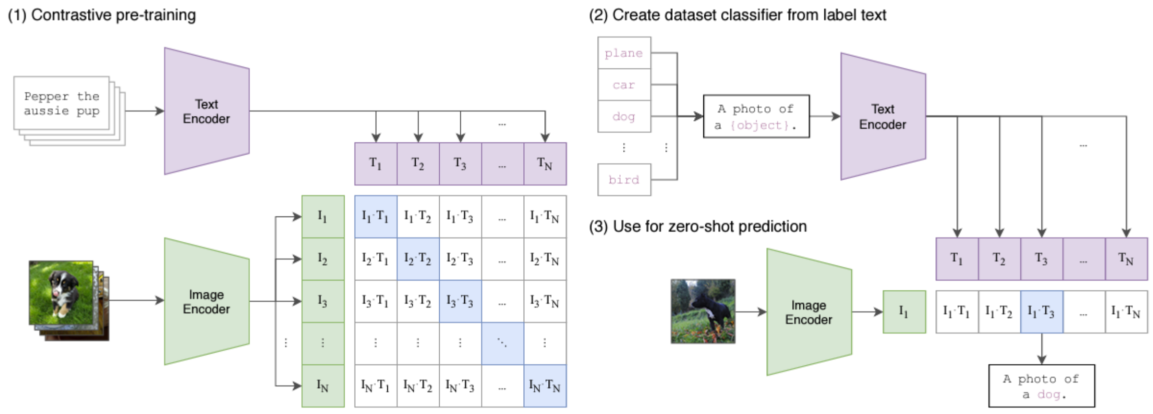

Figure 2.1: OpenAI's CLIP model pipeline. Graphic from [1].

2.2.2 Object Tracking

Contrastive learning has been used for trajectory estimation, which is an important problem in many vision tasks. In [22], image features are extracted from a sequence of video frames and they are contrasted with trajectory centers of nearby objects to improve the tracking accuracy during occlusions. Instead of using the entire trajectories as inputs, the trajectories are represented as a “center vector” to reduce computation effort required during training. However, tracking accuracy may be compromised if irregular trajectories or fast-moving objects are present in the environment, in which case using a simplified representation of the trajectory may be inappropriate. [21] approaches the same problem by contrasting observed trajectories with future trajectories, sampled from a VAE. However, this algorithm for sampling is relatively slow when operating in real-time.

2.3 Conclusion of Literature Review

Overall, there is no existing work that applies map inference in 3D irregular and unstructured environments. Additionally, contrastive learning has been used in object tracking where the trajectory of an object is related to the visual camera information, but has not been used in the case of map inference, which requires relating the trajectory to the surrounding map information. Thus, the literature in map inference presents a gap that this work will aim to close.

Chapter 3

Summary of Approach

3.1 Problem Statement

Consider multiple non-holonomic wheeled robots in a 3D environment, each equipped with a limited range 3D LiDAR. The environment, $\boldsymbol{E}$, is a bounded area of fixed size with rough traversable terrain and irregular shaped obstacles. Each robot, r, builds an incremental heightmap of the environment, $\boldsymbol{H_t^r}$, through the exploration of $\boldsymbol{E}$, following the recursive update $\boldsymbol{H_t^r} = \boldsymbol{H_{t-1}^r} + \boldsymbol{L_t^r}$ where $\boldsymbol{L_t^r}$ is robot r's new local observations of the environment at timestep t. Each robot is also storing its 2D position at each timestep, $(\boldsymbol{x_t^r}, \boldsymbol{y_t^r})$. The trajectory of a robot is given by $\boldsymbol{T_r} = [(\boldsymbol{x_0^r}, \boldsymbol{y_0^r}), (\boldsymbol{x_1^r}, \boldsymbol{y_1^r}), \ldots, (\boldsymbol{x_t^r}, \boldsymbol{y_t^r})]$. The robots in the environment can communicate trajectories with each other, given that they are within c meters of each other.

The goal of general robot map inference problems is for each robot to predict the probability of occupancy of an unknown area in the environment, $\boldsymbol{U_t^r}$, given its current observations at a given timestep. That is, $\boldsymbol{P_t^r} = \boldsymbol{U_t^r} + \boldsymbol{H_t^r} = P(\{\boldsymbol{E} \setminus \boldsymbol{H_t^r}\} | \boldsymbol{H_t^r})$. In our case problem definition, we are additionally given the trajectories of observed robots in the environment. Thus, our goal is to predict $\boldsymbol{P_t^r} = P(\{\boldsymbol{E} \setminus \boldsymbol{H_t^r}\} | \boldsymbol{H_t^r}, \boldsymbol{T_i}, \ldots, \boldsymbol{T_j})$, where $\boldsymbol{T_i} \ldots \boldsymbol{T_j}$ are the observed trajectories of robot r.

3.2 Overview

The proposed workflow is summarized as follows:

1. Robots are initialized at different locations in $\boldsymbol{E}$.
2. Each robot explores the environment while collecting its heightmap, $\boldsymbol{H_t^r}$, and trajectory $\boldsymbol{T_r}$.

3. When two robots are within communication range, c, each robot transfers its trajectory to the other robot.
4. The current heightmap and trajectory are passed into the **map completion network**, and the predicted map, $\boldsymbol{P_t^r}$ is generated.
5. $\boldsymbol{P_t^r}$ is replaced with $\boldsymbol{H_t^r}$. We repeat from Step 2, until the environment has been explored.

The map completion network will be explained in Chapter 4. In order to evaluate the performance of the map completion network, we determine the information gain and the accuracy of the generated map. Information gain refers to how much of the unknown environment becomes known after prediction, and accuracy refers to how far the prediction is from the ground truth. Information gain is chosen as it is a standard metric used in robot exploration, where a robot team aims to maximize the area coverage in an environment in the shortest time. Additionally, we choose accuracy as our second metric in order to evaluate the correctness of our predicted map. The information gain, $\boldsymbol{I_t^r}$, and accuracy, $\boldsymbol{A_t^r}$ are defined as follows:

$$\boldsymbol{I_t^r} = \frac{P_{t,N}^r - H_{t,N}^r}{E_N} \qquad \boldsymbol{A_t^r} = 1 - \frac{\sum_i^{P_{t,N}^r} |\boldsymbol{P_{t,i}^r} - \boldsymbol{E_{t,i}^r}|}{P_{t,N}^r}$$

where $\boldsymbol{E_{t,i}^r}$ represents the ground truth of cell i, $\boldsymbol{P_{t,i}^r}$ is the prediction of cell i, and $P_{t,N}^r$ is the number of cells in the prediction region, $H_{t,N}^r$ is the number of cells in the current observation and E_N is the total number of cells in the environment.

Chapter 4

Methodology

The goal for this chapter is to explain our design for the map inference network. The following sections will detail the data collection process, the network architecture for map inference, the new contribution involving contrastive learning, and finally, the proposed architecture. All neural networks are trained with a NVIDIA GeForce RTX 3070 GPU, with a peak RAM usage of 60GB.

4.1 Data Collection

In order to be able to train both the CL and VAE neural networks, a dataset containing robot trajectory, partial heightmap and ground truth heightmap triplets will need to be generated. A script has been created to achieve the following:

1. Generate uneven terrain with irregular shaped obstacles and spawn environment in Gazebo
2. Select feasible goal locations in traversable regions of the map
3. Spawn robot and environment and perform waypoint navigation to traverse to the goal locations, collecting both height map information and trajectories in the process
4. Rinse and repeat

In total, roughly 20,000 trajectory + heightmap + ground-truth triplets have been collected in over 2,000 unique environments. The following subsections explain the specifics above the above procedure in more detail.

4.1.1 Simulator

A simulator was built in ROS to perform data collection. Some of its important features are listed below:

- **Visualization:** Gazebo and RViz
- **Multiple Robots:** Can simulate up to 6 Clearpath Jackals [23] in an environment. The Jackal was chosen since it is a non-holonomic robot, widely used, and has ROS support.
- **Waypoint Navigation:** Achieves autonomous waypoint navigation using the ROS package move_base [24], given a sequence of 2D waypoints.
- **Elevation Mapping:** Performs heightmap generation using the elevation_mapping package [25, 26]. Builds an elevation map within a given radius around the Jackal using point cloud data from a LiDAR.

4.1.2 Terrain Generation

An existing research paper has focused on building a tool to convert grayscale images to 3D models [2]. The pixel intensity of the grayscale image is proportional to the height of the model in 3D, where darker pixels indicate lower height (see Figure 4.1). Using this tool, building 3D environments simplifies to generating 2D grayscale images.

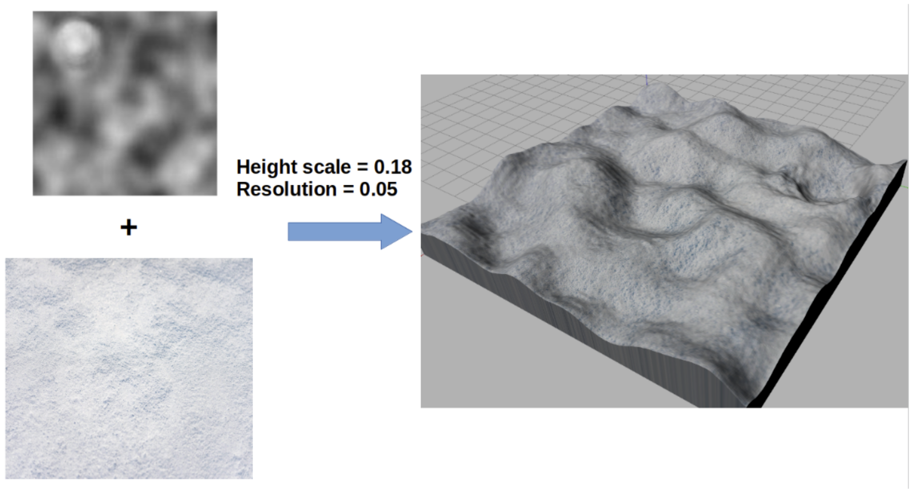

Figure 4.1: Conversion of grayscale image and texture to build 3D generated terrain in Gazebo using LIRS-WCT [2].

These images will need to contain smooth traversable slopes, for a Jackal to be able to traverse, and irregular and large obstacles scattered around the environment. The simplest terrain generation method that can meet these criteria and has existing open-source implementations available is the Diamond-Square Algorithm [27]. Figure 4.2 shows the process that was used to generate randomized terrain.

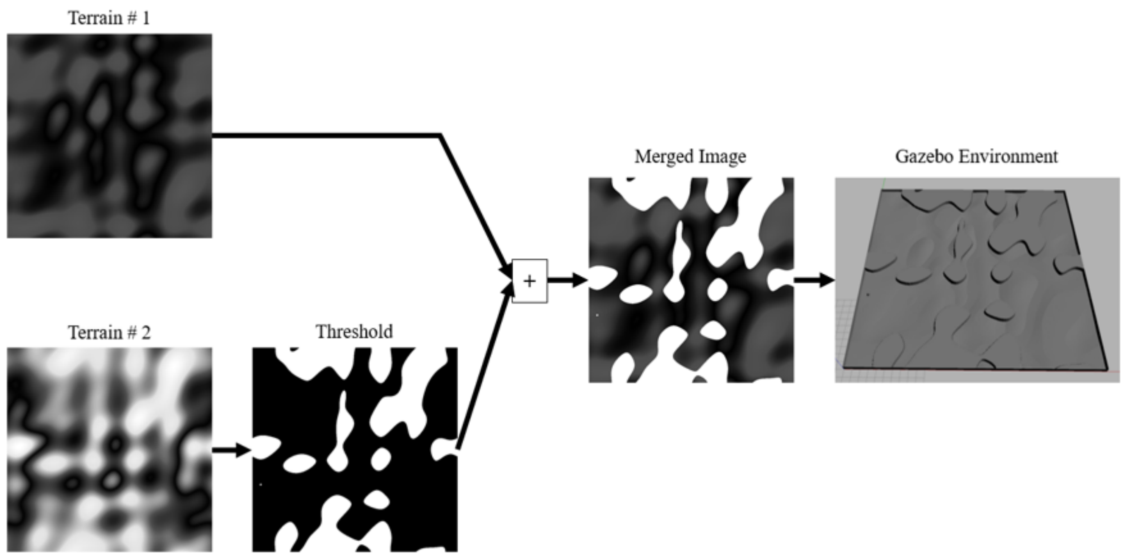

Figure 4.2: Terrain and obstacle generation pipeline. Two heightmaps are generated using the diamond-square algorithm. One of the heightmaps is thresholded to binary (set pixel values above a given intensity to white and the remaining to black), which will form the obstacles, and the other is used as terrain. The two images are combined and the Gazebo environment is generated using LIRS-WCT [2].

The scale of the terrain is preset such that the obstacles are $\tilde{0}$.7m and the terrain ranges between 0 and 0.3m. The obstacle heights are roughly twice the size of the Jackal to ensure that it cannot traverse those regions, and the terrain slopes were selected keeping in mind the 65mm clearance specification of the Jackal [23] Moreover, the environment size was chosen to be 30m$\times$30m. This was chosen as an optimal balance between large exploration size and environment generation time in Gazebo.

4.1.3 Waypoint Navigation

We can use the pre-built simulator achieve autonomous waypoint navigation and heightmap generation. In order to be able to use the 2D navigation stack on 3D environments, we must specify the minimum obstacle height as a parameter to move_base so that it can differentiate between terrain and obstacles. In this case we set it to 0.6m, that is, between the maximum terrain height and the obstacle heights.

To generate the set of waypoints for the robot to follow, we can sample arbitrary points within the traversable region. This is sufficient for our purposes because we don't need to explore the entire environment to be able to collect the dataset. However, the method we used to generate the terrain presents another challenge. Consider the heightmap in Figure 4.3, where there are multiple disconnected traversable regions. In such an environment, we cannot sample points from both regions since there isn't a feasible path to get from one region to the other. Thus, we set an additional constraint that we can only spawn the robot and sample waypoints from the largest connected traversable region. The largest region can be determined using the flood-fill algorithm [28].

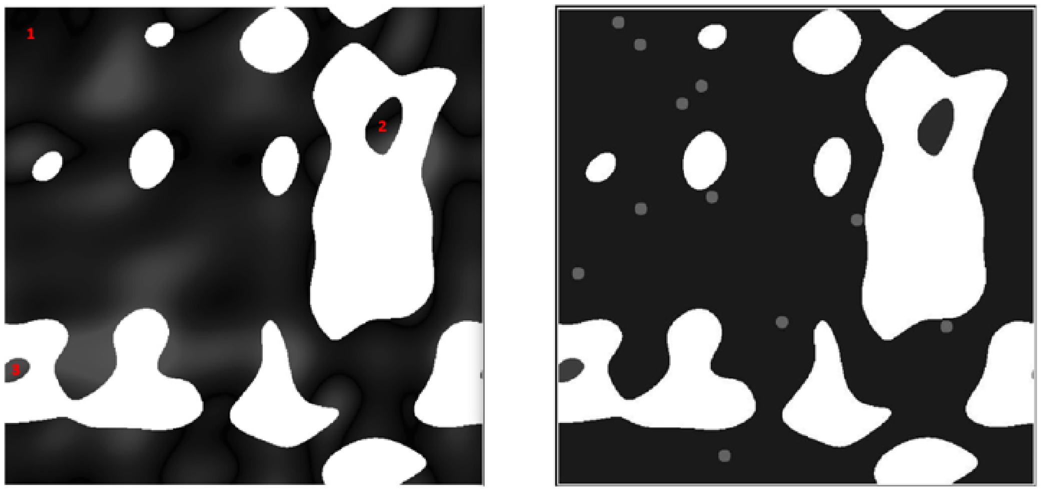

Figure 4.3: Left (a) depicts how this heightmap contains three disconnected regions. Right (b) shows the selected waypoints (grey circles) in the largest region.

Finally, we store the trajectory by saving the ground truth odometry topic of the Jackal published by Gazebo.

4.1.4 Heightmap Generation

During exploration, the Jackal will also need to collect heightmaps based on the observed region of the environment. Using the simulator, we can determine the height of the environment around the robot as it is exploring. By appending previous elevation maps with the current ones, we can incrementally build a heightmap for any given trajectory of the robot, as seen in Figure 4.4.

The ground truth odometry of the Jackal during exploration as well as the heightmaps are all collected in a rosbag file [29]. Using rosbag allows for compact data representation and to store useful metadata that can be accessed later on. Additionally, we save the ground truth heightmaps and the collada files used to generate the environment in Gazebo.

4.1.5 Augmentation

To generate more data, we rotate all of our heightmaps and trajectories by 90^o in all directions. This produced a dataset of roughly 80,000 heightmap image pairs.

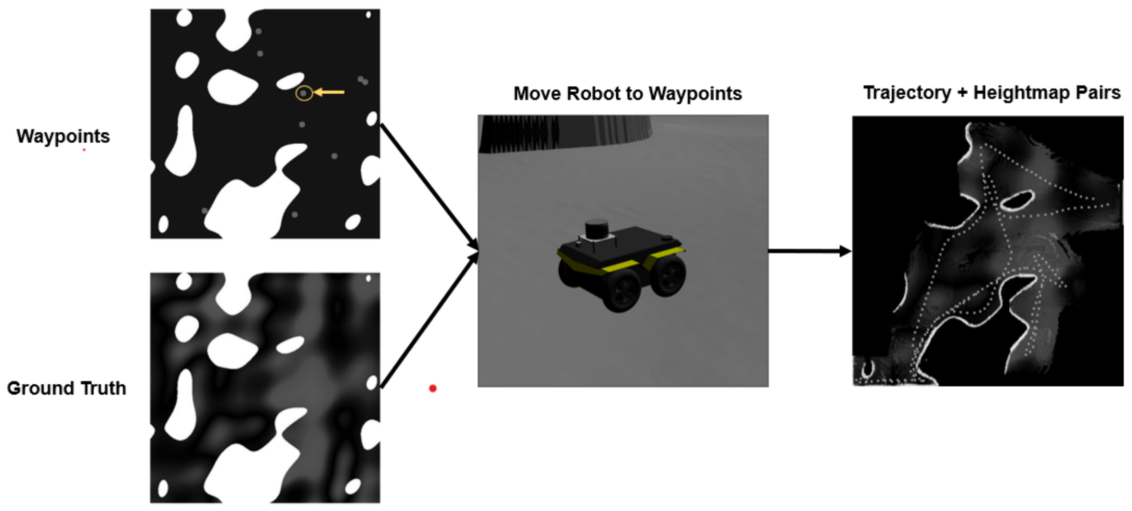

Figure 4.4: By sampling the waypoints and generating the environment from the ground truth, we can use the simulator to move the robot in the environment and collect trajectory and heightmap pairs.

4.2 The Variational Autoencoder for Map Inference

The Variational Autoencoder (VAE) architecture is different from a regular autoencoder, as it aims to reconstruct the output by sampling over a Gaussian distribution in the latent space and learns the latent representation, z (4.5). This latent representation is then sent as input to the decoder to provide the reconstructed completed map. This neural network is appropriate for the map inference problem because the VAE learns a probability distribution rather than a single representation, which makes it better suited to generation problems.

In order to build a baseline model for our 3D prediction pipeline, we leverage existing state-of-the-art models for 2D environments. Our chosen baseline model is [3], which features a VAE for 2D map inference in structured environments. This paper was selected because it outperforms both the GAN based and LMRC methods, so it is the most representative of the literature in data-driven approaches. Additionally, by using this network architecture, extending from binary costmap images (2D environments) to grayscale heightmap images (3D environments) is not challenging and requires little modification. This specific paper employs a VAE with *skip-connections* and the trained model is used to predict the map at a specified frontier location in real-time.

In the following subsections, we first demonstrate our reproduction of the baseline architecture described in [3]. We then demonstrate how to extend this baseline to predict in 3D environments.

4.2.1 VAE Baseline with 2D Structured Environments (SOTA)

The VAE for 2D map inference is trained on the KTH Dataset, which contains over 150 floorplans and 6000 rooms of the KTH Institute of Technology [30]. The prediction task involves pixel wise binary classification, predicting the environment as being free or occupied. Thus, a binary cross entropy loss function is appropriate.

$$L_{BCE} = \sum_n -[y_n \log x_n + (1 - y_n) \log(1 - x_n)]$$

x_n represents the prediction of pixel n, and y_n represents its corresponding ground truth. The paper also introduced an additional loss function for the latent representation, the Kullback-Liebler Divergence (KLD) loss. This was, however, omitted during our reproduction of the paper as the authors place a significantly larger importance weight on the BCE Loss, and hence, the KLD loss did not have a large impact on the reconstruction performance.

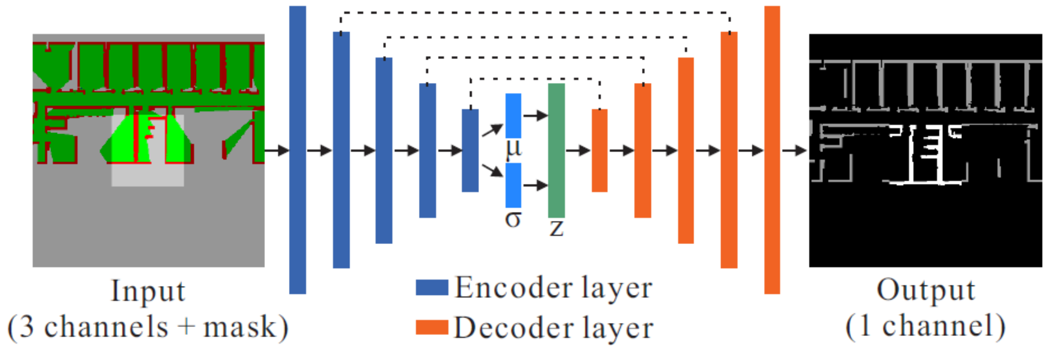

Figure 4.5: The architecture of the VAE paper [3]. The input is a four channel image, representing the concatenation of unknown space, free space, occupied space and the prediction mask. The output of the channel is a reconstructed binary prediction of the heightmap, where white represents occupied space and black represents free/unknown space.

The architecture of the VAE, shown in Figure 4.5, consists of an encoder and decoder layer. The encoder takes in a 256×256 map and consists of five convolutional layers, applying 2D batch normalization and a LeakyRelu activation after each layer. Two additional convolution layers are used to output a mean, μ, and log variance $\log \sigma$, and a latent vector z is sampled. This sampled vector is then passed through five deconvolutional layers, identical to the corresponding convolutional layers, and skip-connections are applied by concatenating the deconvolution layer vectors with the convolutional layer outputs. The output is a binary "completed" map of dimensions 256×256.

Several models were trained by performing a grid search over various hyperparameters including learning rate, weight decay and the size of the latent space, z. These hyperparameters were chosen since they had the largest impact on the training curves. The best model was selected based on the criteria of lowest overall L_{BCE}. The hyperparameters that provided the best results were [learning_rate=1e-5, weight_decay=1e-4, batch_size=32, epochs=250, latent_size=20]. Sample inputs and reconstructions for the best model are shown in Figure 4.6.

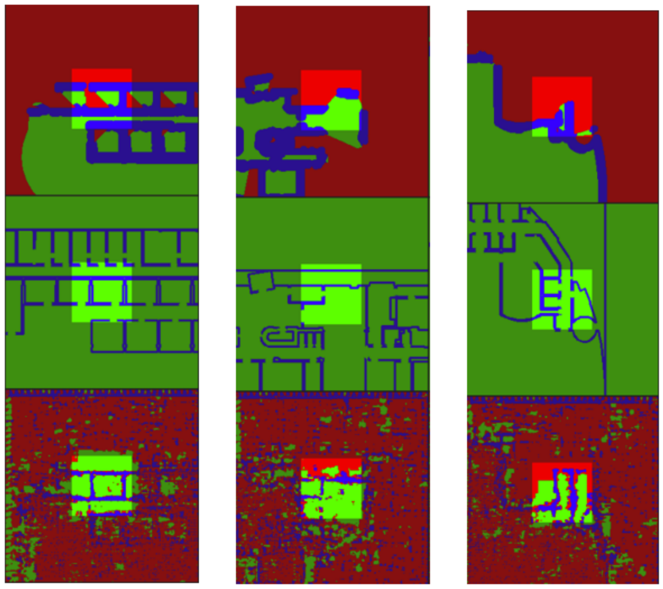

Figure 4.6: Row 1: sample costmap inputs generated by the simulator as the robot is exploring the environment. In the costmap, green=free, blue=obstacle, red=unobserved. Row 2: ground truth images, Row 3: VAE reconstruction. Note that the masked region (highlighted square) is the region that is being predicted over.

To better visualize the results, the predictions were shown real-time in the provided simulator. Figure 4.7 shows a sample snapshot during the simulation. The benefit of reproducing this in the 2D environments is three-fold; 1) to help gain familiarity with the SOTA approach, 2) to evaluate the limitations and the performance of the VAE as a whole and 3) to understand the implications of extending to 3D environments. Some of the key observations relating to the overall performance of the network are listed below:

- The network takes in 256×256 images of costmaps as input. Additionally, a mask can be added to specify the size of the region ($\leq 256 \times 256$ in size) to predict over. To reuse the same structure, the 3D environment will also need to use this as input.
- The reconstructions are noisy and often discontinuous. The VAE struggles with producing realistic reconstructions of the environment.

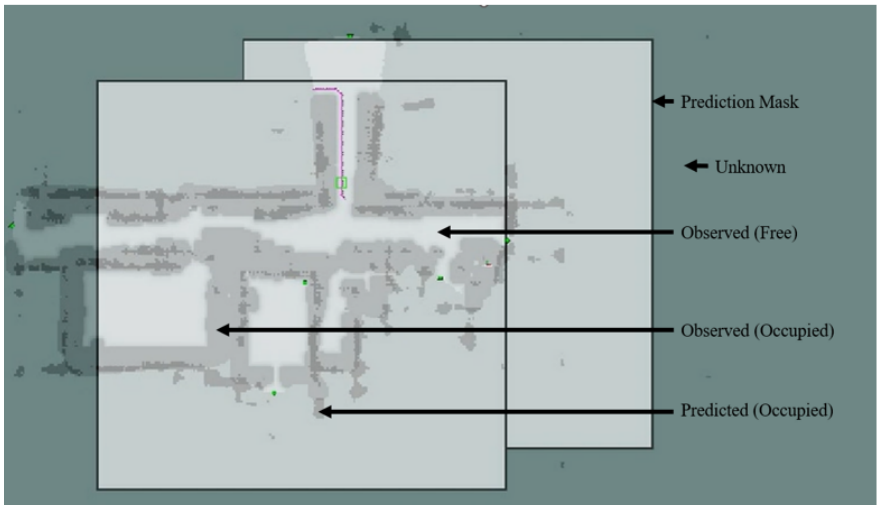

Figure 4.7: Simulator with predictions overlaid in the 2D environments. The square highlight is the prediction mask, light grey is observed free space, light black is the observed occupied space, the dark black is the predicted occupied space by the VAE and the remaining is unknown. Note the inability to predict deep into the environment and the discontinuous reconstructions.

- VAE cannot predict deep into the unknown regions of the map. In other words, the amount information gain for occupied areas is limited.

4.2.2 VAE Baseline with 3D Unstructured Environments

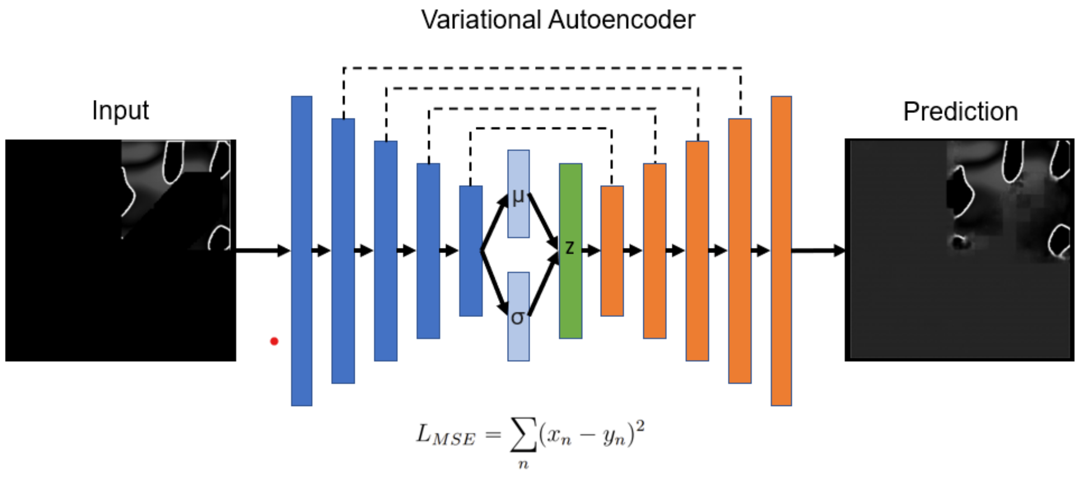

Figure 4.8: VAE Baseline with 3D Terrain prediction.

In order to extend our baseline VAE train in 3D environments, we will need to make a few modifications:

- **Loss Function:** As we are predicting over a range of pixel values, $(0,255)$ for the heightmaps, the loss function will need to be modified. An MSE Loss is appropriate as it is lightweight and apt for multi-class predictions.

$$L_{MSE} = \sum_n (x_n - y_n)^2$$

- **Dataset:** Modifications were made to the dataset handling as we are no longer selecting frontiers to predict over. In the paper [3], the floorplans were cropped around a masked region to predict over. Instead, we propose to predict over the entire map, instead of a local region around the frontier point.
- **Hyperparameters:** Due to the added complexity of the task, some additional modifications were made to the hyperparameters. Specifically, the learning rate was decreased to 1e-6 and the latent size was increased to 64 to improve convergence of the model.

After the above modifications, the VAE baseline is able to generate map predictions for 3D environments as shown in Figure 4.8.

4.3 Contrastive Trajectory-Map Pretraining

Constrastive Learning is an emerging self-supervised learning technique used to associate representations of similar samples and disassociate the dissimilar ones. In Section 2.2, a recent architecture CLIP was introduced to associate text with their corresponding image. In this section, our goal is to modify the CLIP architecture so that it can be used to associate trajectories with their corresponding heightmaps. By integrating this into the architecture of the VAE, we can improve its prediction capacity, as it will be able to better infer the shape of surrounding obstacles.

4.3.1 Data Processing

As outlined in Section 4.1, the simulator was used to generate trajectory-heightmap pairs. By tokenizing the trajectories, the training of the model becomes analogous to that of the CLIP architecture.

One problem we need to resolve is that the trajectory collected contains too many data points, sampled at very high resolution. As a result, the network becomes extremely high dimensional and is difficult to achieve convergence. To resolve this problem, we divide the environment into 1m$\times$1m grids and only include the states of the robots that fall within

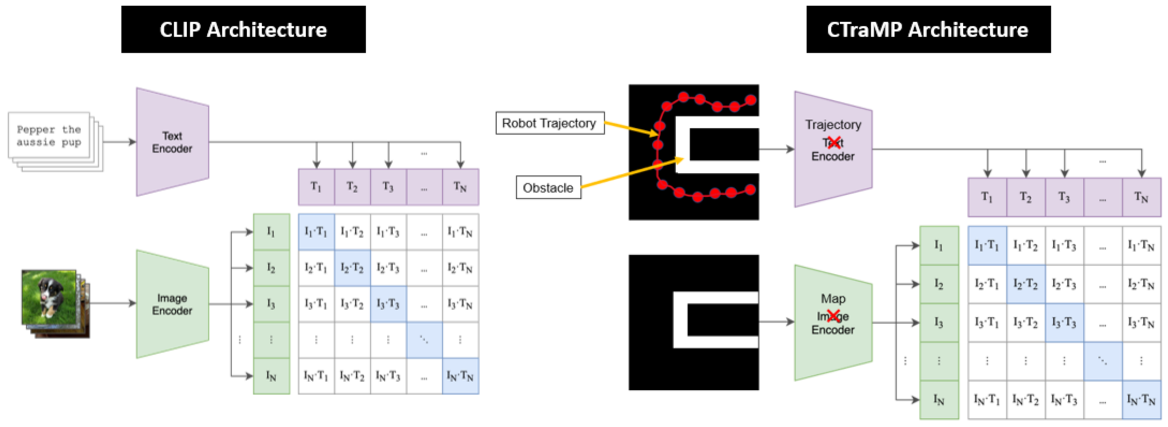

Figure 4.9: Our modification to the CLIP architecture to work for 3D map inference. Notice how the robot trajectory has a similar shape to the obstacle it is moving around.

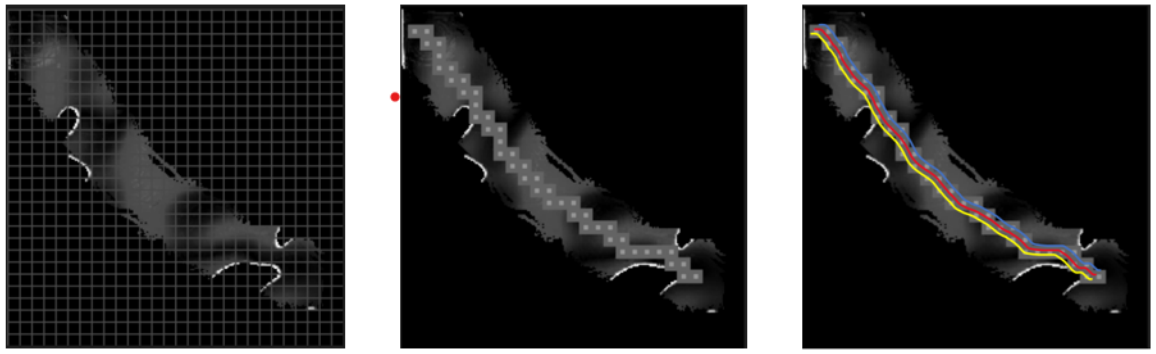

Figure 4.10: Left image (a) shows the resolution of the grid for trajectory encoding. Center image (b) shows a sample trajectory with that resolution. Right image (c) shows how three slightly different trajectories would correspond to the same encoding, thus resulting in significantly better convergence of the model.

these grids into the trajectories. This resulted in a grid size of 28×28. This grid size was chosen as it roughly approximates the dimensions of the robot. This brings the model down from a vocabulary size of 256^2 grids to 28^2 grids. The intuition behind the benefit of using a larger resolution is that it groups trajectories that are very similar to each other. This is demonstrated in Figure 4.10.

The trajectory also needs to be converted into a usable representation for the contrastive learning model. The trajectory will by tokenized using the following method:

1. Each grid cell with the new resolution will be labeled from 0 to $28^2 - 1$. This converts the trajectory into a sequence of numbers.
2. <SOS> and <EOS> tokens, given by $28^2 + 1$ and $28^2 + 2$, will be affixed to the start

and end of the trajectory.

3. The trajectory is padded to a constant length of 250, using the padding id 28^2. This length was chosen as it roughly matched the longest trajectory size in the dataset.
4. Constant attention is used; 0 if the state is a padding token, 1 otherwise.

To augment our dataset further, the longer trajectories were divided into sub-trajectories and associated with the same partial heightmap. This is a reasonable action since the longer trajectories often covered the same amount of area in the map as their sub-trajectories. An example of this is shown in Figure 4.11. By performing this step, the dataset was increased to 200,000 heightmap-trajectory pairs.

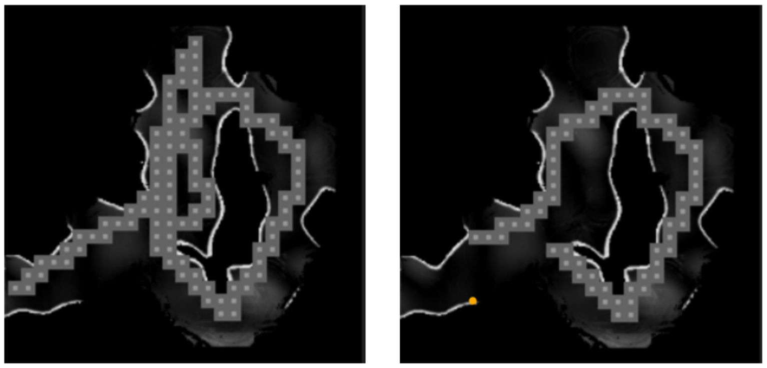
Figure 4.11: These two trajectories are pairs to the same map, where the trajectory on the right is a subtrajectory of the one on the left.

4.3.2 The CTraMP Pipeline

The complete pipeline for training and testing the contrastive learning model is shown in Figure 4.12. The architecture is based on CLIP [1], and was chosen as it largely resembles our current prediction task and achieves state-of-the-art performance. At train time, the trajectories and maps are passed into their respective encoders and their embeddings are contrasted. A DistilBERT model is used as the trajectory encoder and ResNet-50 is used for the map encoder. At test time, a single trajectory is passed in and the map that best fits that trajectory is found from a database.

The model was trained with learning_rate=1e-6, weight_decay=1e-3, batch_size=16 and dropout_prob=0.3 for 50 epochs.

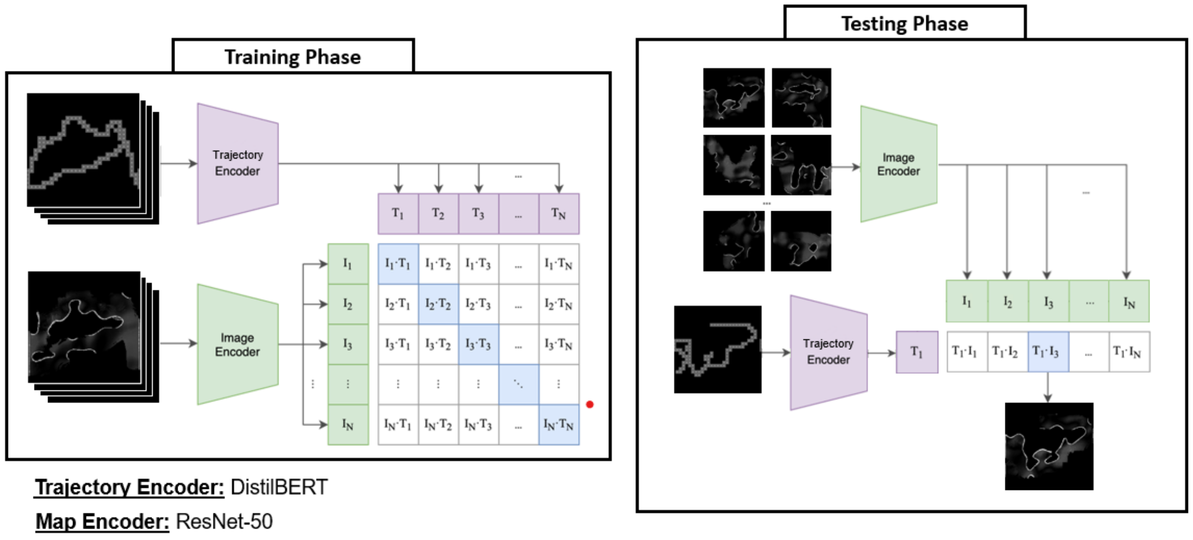
Training Phase
Trajectory Encoder
Image Encoder
T_1 T_2 T_3 ... T_N
I_1 I_2 I_3 I_N
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ ... $I_1 \cdot T_N$
$I_2 \cdot T_1$ $I_2 \cdot T_2$ $I_2 \cdot T_3$ ... $I_2 \cdot T_N$
$I_3 \cdot T_1$ $I_3 \cdot T_2$ $I_3 \cdot T_3$ ... $I_3 \cdot T_N$
$I_N \cdot T_1$ $I_N \cdot T_2$ $I_N \cdot T_3$ ... $I_N \cdot T_N$
Trajectory Encoder: DistilBERT
Map Encoder: ResNet-50
Testing Phase
Image Encoder
Trajectory Encoder
I_1 I_2 I_3 ... I_N
T_1
$T_1 \cdot I_1$ $T_1 \cdot I_2$ $T_1 \cdot I_3$ ... $T_1 \cdot I_N$

Figure 4.12: The complete training and testing pipeline for CTraMP.

4.3.3 Evaluation

Apart from using loss characteristics to identify whether our model works, additional testing was performed on never-before-seen samples. A GUI was built to take in a hand-drawn trajectory as input, and return the best matches from a map database as the output. It was found that our model did well to associate the two across various trajectory shapes and sizes, thus verifying the success of our model. Although this is a qualitative evaluation strategy, it is a sufficient for our purposes.

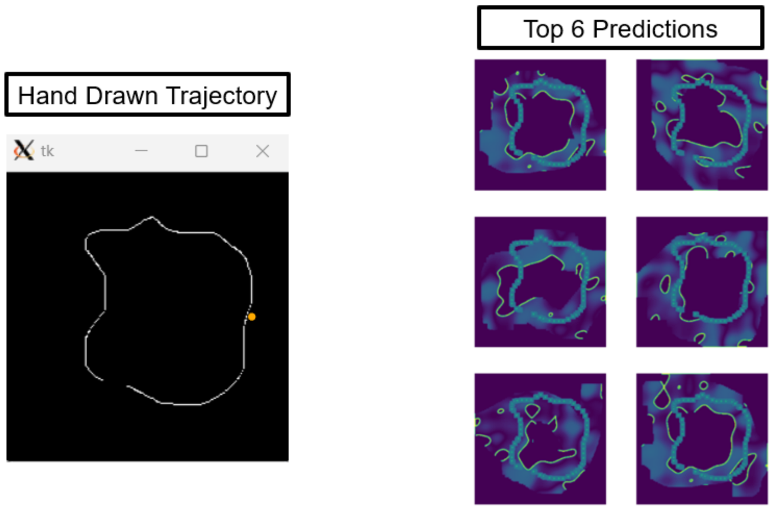

Figure 4.13: Predictions from the contrastive learning model for a hand-drawn trajectory.

4.4 The Proposed Architecture

4.4.1 Dataset

For simplicity, we train our model on a two-robot system. **Robot 1** will be defined as the "observer robot"; that will be performing the map prediction step. It will collect and store its own partial heightmap, H_t^1. **Robot 2** will follow and store its trajectory, T_t^2, which will be communicated to Robot 1 when they are within communication range. The desired prediction output (ground truth) of **Robot 1** will be a heightmap that combines the two heightmaps. Thus, Robot 1's initial heightmap will be the input to the neural network and the desired output will be our ground truth. This is shown in Figure 4.14.

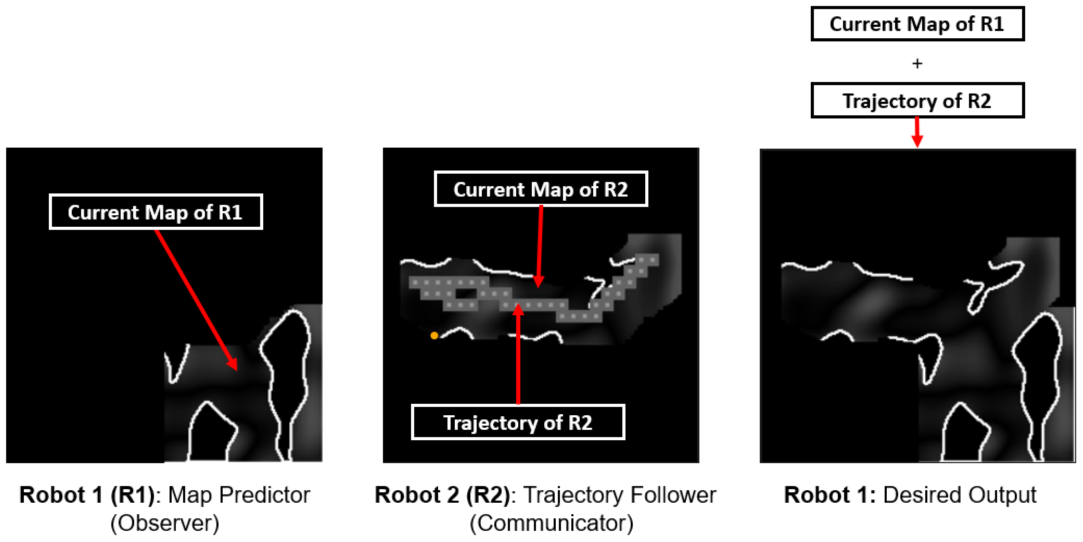
Current Map of R1
Current Map of R2
Trajectory of R2
Current Map of R1
+
Trajectory of R2
Robot 1 (R1): Map Predictor (Observer)
Robot 2 (R2): Trajectory Follower (Communicator)
Robot 1: Desired Output

Figure 4.14: Robot 1 collects its heightmap, Robot 2 follows a trajectory. The desired output is a combination of the two heightmaps.

As outlined in Section 4.1, we only have heightmap-trajectory pairs (ie. the dataset for Robot 2). To build our Robot 1 dataset, we simply mask arbitrarily sized unknown regions around partial heightmap of Robot 2. This will be sufficient to emulate a prediction step in a real-time cooperative multi-robot scenario.

4.4.2 Architecture

Our proposed architecture to achieve multi-robot map prediction is done in a multi-step process, using an *ensemble* learning technique. There are two VAEs used; the first (VAE 1) is used to predict the obstacles using binary classification, the second (VAE 2) is used to predict the terrain. The reason it makes sense to do it this way is because the distribution for the obstacles is very different from that of the terrain (the obstacle height is 3 times that the size of the terrain). Thus, we train VAE 1 using BCE Loss and VAE 2 with MSE Loss, defined in Section 4.2. The process is achieved using the following:

1. Robot 1 map is used as input for VAE 1 and VAE 2.
2. Robot 2 trajectory is used as input to the trajectory encoder, trained in Section 4.3.
3. Robot 1 map passes through encoder stages of VAEs to sample a latent vector z, as described in Section 4.2.
4. Trajectory embedding is appended to the latent vectors. The combined vector is sent as input to the decoders.
5. Each VAE produces its prediction; the obstacles for VAE 1 and the terrain for VAE 2.
6. The obstacle prediction is appended to the terrain prediction, giving the final, overall output.

The complete architecture is shown in Figure 4.15.

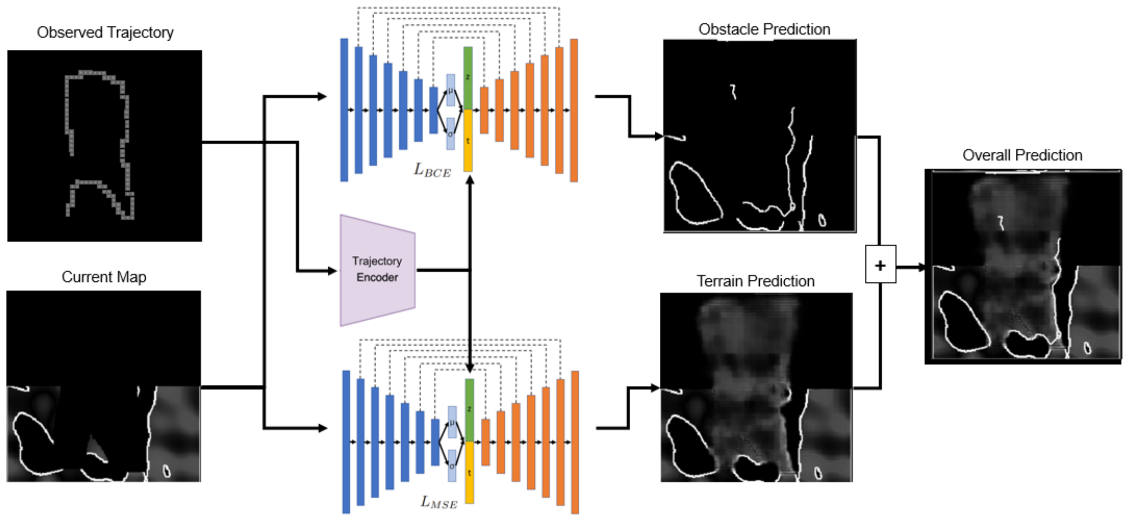

Figure 4.15: The complete training and testing pipeline for CTraMP.

There are also a few additional modifications that were made to enable better prediction outputs for obstacle prediction:

- A weighted BCE Loss was used, to give higher prediction weight to the obstacles. This was needed because VAE 1 often predicted free space significantly more frequently than that of obstacles, likely due to the fact that predicting free space leads to a lower overall loss than predicting obstacles (since the free area is significantly higher than obstacle area).

$$L_{WBCE} = \sum_n -w_n[y_n \log x_n + (1-y_n)\log(1-x_n)]$$

After some experimentation, $w_n = 3$ for obstacles and $w_n = 1$ for free space resulted in the best overall convergence.

- Since VAE 1 produces a distribution, some post-processing was required. OpenCV [31] was used to threshold the image to binary values and perform contour detection around the prediction regions.

The next section will discuss the experiments performed and the results.

Chapter 5

Experiments and Results

The experiments were conducted in two different forms. In the first, offline experiments were done on hold-out test maps and trajectories. Comparisons were made between the baseline VAE and the proposed solution using the metrics outlined in Section 3. In the second, an experiment was performed live in a two robot scenario using our previously built simulator in a new environment.

5.1 Offline Experiments

The contrastive learning embedding had a significant impact on the convergence of the model. Figure 5.1 shows how introducing the contrastive learning embedding acted like an "attention" mechanism to only focus on predicting at specific regions of the image.

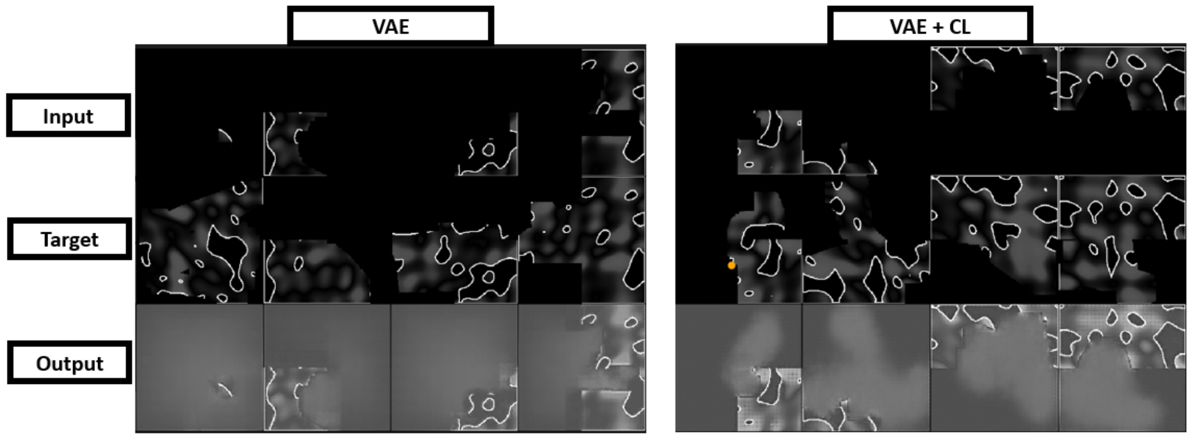

Figure 5.1: Comparison of VAE vs VAE+CL. These are predictions at the 3rd epoch of training. Notice how the non-CL model seems to predict everywhere at random whereas the CL model learns to focus on predicting at regions of the map corresponding to Robot 2's trajectory.

The predictions for both the baseline VAE and the proposed solution are shown in Figure 5.2. The baseline model fails to predict deep into the map and also generates noisy predictions. Note that we noticed very similar problems in Section 4.2, where we use the baseline solution in 2D environments. One explanation for this could be the simplicity of the baseline architecture and its inability to generate multi-modal distributions (ie. both obstacles and terrain). On the other hand, our solution produces predictions that are very similar to the ground truth. Although the obstacle prediction is not perfect, they are feasible generations given the trajectory of Robot 2.

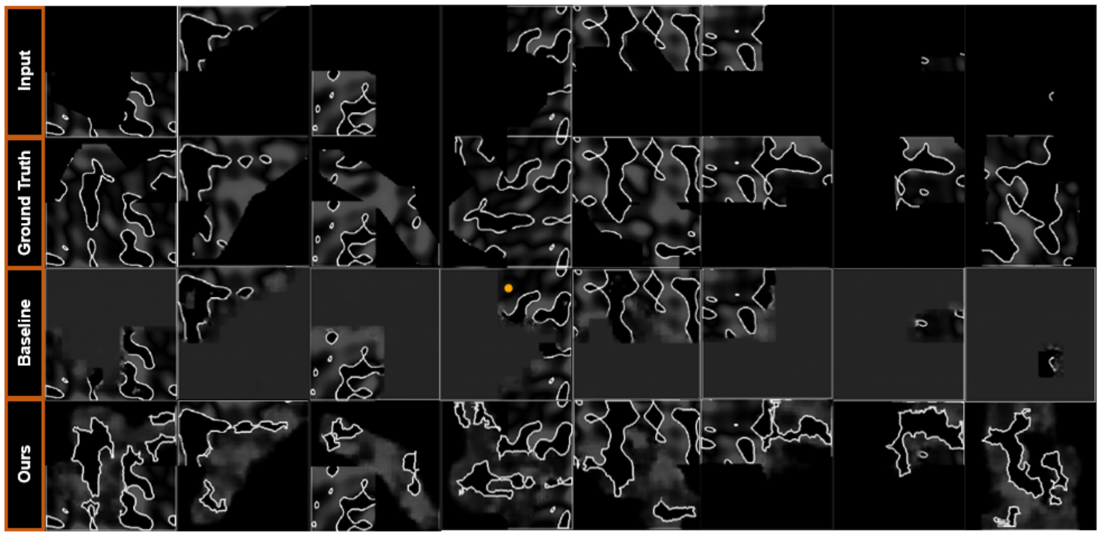

Figure 5.2: Comparison of VAE Baseline vs VAE+CL in predictions.

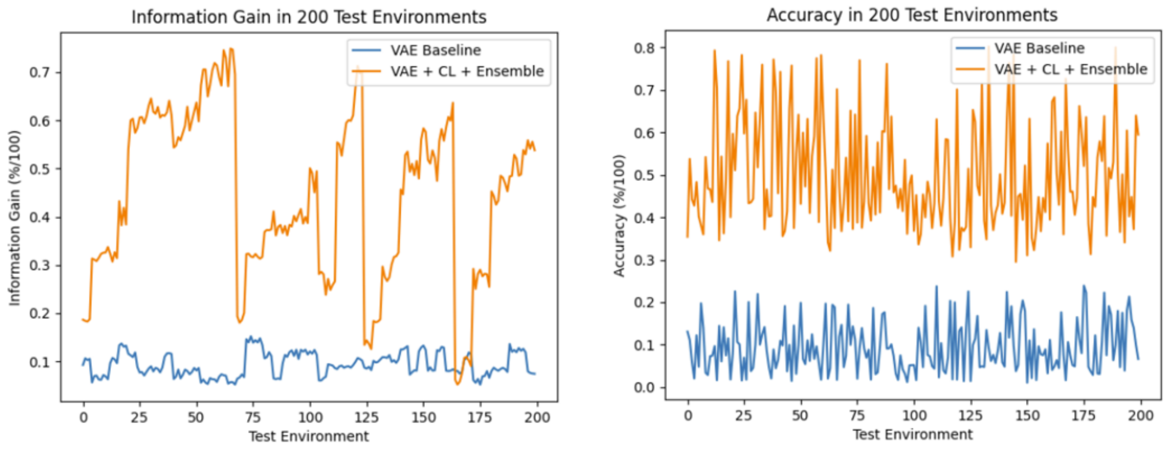

Figure 5.3: Comparison of VAE Baseline vs VAE+CL in information gain and accuracy in 200 unseen test environments.

Figure 5.3 shows the plots of information gain and accuracy, the metrics described in

Chapter 3. Our proposed solution has a significant performance boost in both metrics than the baseline. Note that the spikes seen in the information gain are completely expected. This is because the information gain is dependent on the length of the trajectory of Robot 2. The longer the trajectory of Robot 2, the greater the depth of the predictions. This is further justified by the "attention mechanism" from the trajectory embedding (Figure 5.1) as the attention area depends on the trajectory.

5.2 Simulator Experiments

Due to time constraints of the project, only one experiment was completed in the simulator: Simulator Test. In this test, we have a two robot scenario as described in subsection 4.4.1. The prediction is updated in the map when Robot 1 and 2 are in close range of each other. The obstacle predictions are not entirely accurate, but are feasible given the trajectory of Robot 2. Moreover, the area updated in the prediction spans the entire area covered by the trajectory, with reasonable terrain estimates. This confirms the success of our solution. However, more testing is needed to further validate the performance real-time.

Chapter 6

Conclusions and Future Work

In this work, a novel technique to achieve map prediction with multiple robots was achieved. A multi-layered architecture involving ensemble and contrastive learning was presented. The innovation behind the success of the architecture is using Contrastive Trajectory-Map Pretraining to improve the prediction capacity of the existing solution. The design was evaluated against the baseline solution in both information gain and accuracy, and it provided a 5-fold improvement in both metrics. The proposed solution results in efficient transfer of data in a multi-robot scenario; specifically, only trajectories need to be communicated among robots as opposed to maps, which are significantly more space intensive.

There are a number of limitations in our approach however. Trajectory tokenization and encoding rely on fixed size environments. For environments in the real world that are without bound, a more sophisticated technique to encode the trajectory, without having a large vocabulary size, is required. Additionally, the solution does not address the problem of dynamic obstacles. Moving objects present in the scene may skew the trajectory of the robot and the accuracy of the predicted map. A method to differentiate between static and dynamic obstacles would ensure robustness of the solution. Moreover, the solution relies on perfect trajectory and map estimation, which is not a realistic assumption given that there is always noise and drift in sensors. Accounting for these errors through integration with a SLAM solution may be a potential approach to addressing this.

A few improvements are planned in the future for this project. Testing in the simulation with more than two robots will be integrated in the pipeline. Specifically, an additional step involving map merging will be required, in order to account for map updates from all robots. Our prosposed technique will also be integrated in a multi-robot system with the task of exploration or navigation, which will give us another metric to evaluate the effectiveness of our solution. In addition, robustness to communication dropout will be addressed. This occurs when a robot fails to collect its map or trajectory information for arbitrary segments along the path, or is unable to send its complete trajectory during communication with another robot due to some failure. This can be improved on by retraining our contrastive learning model, using segmented trajectories as input.

References

[1] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning Transferable Visual Models From Natural Language Supervision," *arXiv e-prints*, p. arXiv:2103.00020, Feb. 2021.

[2] B. Abbyasov, R. Lavrenov, A. Zakiev, K. Yakovlev, M. Svinin, and E. Magid, "Automatic tool for gazebo world construction: from a grayscale image to a 3d solid model," International Conference on Robotics and Automation (ICRA), 2020, pp. 7226-7232.

[3] R. Shrestha, F.-P. Tian, W. Feng, P. Tan, and R. Vaughan, "Learned map prediction for enhanced mobile robot exploration," in Proceedings of IEEE International Conference on Robotics and Automation, Montreal, Canada, 1197-1204 (2019).

[4] D. Dworakowski, C. Thompson, M. Pham-Hung, and G. Nejat, "A Robot Architecture Using ContextSLAM to Find Products in Unknown Crowded Retail Environments," Robotics. 10(110), (2021).

[5] A. Elhafsi, B. Ivanovic, L. Janson, and M. Pavone, "Map-Predictive Motion Planning in Unknown Environments," In Proceedings of IEEE International Conference on Robotics and Automation. Paris, France, 8552–8558, (2020).

[6] S. Ramakrishnan, Al-Halah, and K. Grauman, "Occupancy Anticipation for Efficient Exploration and Navigation," In Proceedings of European Conference on Computer Vision. Virtual, 2020, (2020).

[7] D. Strom, F. Nenci, and C. Stachniss, "Predictive Exploration Considering Previously Mapped Environments," In Proceedings of IEEE International Conference on Robotics and Automation. Seattle, Washington, 2761–2766, (2015).

[8] M. Luperto, L. Fochetta, and F. Amogoni, "Exploration of indoor environments through predicting the layout of partially observed rooms," In Proceedings of Inter-

national Joint Conference on Autonomous Agents and Multiagent Systems. Online, 836–843, (2021).

[9] Y. Katsumata, A. Kanechika, A. Taniguchi, L. E. Hafi, Y. Hagiwara, and T. Taniguchi, "Map completion from partial observation using the global structure of multiple environmental maps," Advanced Robotics. 00(00), 1–15 (2022).

[10] Z. Chen, S. Bai, and L. Liu, "Efficient Map Prediction via Low-Rank Matrix Completion," In Proceedings of IEEE International Conference on Robotics and Automation. Xi'an, China, 13953–13959, (2021).

[11] M. Saroya, G. Best, and G. Hollinger, "Online Exploration of Tunnel Networks Leveraging Topological CNN-based World Predictions," In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. Las Vegas, NV, USA, 6038–6045, (2020).

[12] Z. Shen, L. Kastner, and J. Lambrecht, "Spatial Imagination With Semantic Cognition for Mobile Robots," arXiv: 2104.03638v1. April(2021).

[13] L. Wang, H. Ye, Q. Wang, Y. Gao, C. Xu, and F. Gao, "Learning-based 3D Occupancy Prediction for Autonomous Navigation in Occluded Environments," arXiv: 2011.03981v2. March(2021).

[14] A. Smith and G. Hollinger, "Distributed inference-based multi-robot exploration," Autonomous Robots. 42(8), 1651–1668 (2018).

[15] M. Luperto, L. Fochetta, and F. Amogoni, "Predicting the Layout of Partially Observed Rooms from Grid Maps," In Proceedings of IEEE International Conference on Robotics and Automation. Montreal, Canada, 6898–6904, (2019).

[16] S. Hayoun, E. Zwecher, E. Iceland, A. Revivo, S. Levy, and A. Barel, "Integrating Deep-Learning- Based Image Completion and Motion Planning to Expedite Indoor Mapping," Computing Research Repository (CoRR).(2020).

[17] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation." Medical Image Computing and Computer-Assisted Intervention. 234–241 (2015).

[18] S. Kohlbrecher, J. Meyer, T. Graber, K. Petersen, U. Klingauf, and O. Von-Stryk, "Hector open source modules for autonomous mapping and navigation with rescue robots." RoboCup Symposium. 8371 LNAI624–631 (2013).

[19] B. Yamuachi, “A frontier-based approach for autonomous exploration.” In Proceedings of IEEE International Symposium on Computational Intelligence in Robotics and Automation. Monterey, CA, 146–151, (1997).

[20] E. Tiu, “Understanding contrastive learning.” Medium. from https://towardsdatascience.com/understanding-contrastive-learning-d5b19fd96607 (2021).

[21] M. Halawa, O. Hellwich, and P. Bideau, “Action-based Contrastive Learning for Trajectory Prediction,” *arXiv e-prints*, p. arXiv:2207.08664, Jul. 2022.

[22] E. Yu, Z. Li, and S. Han, “Towards Discriminative Representation: Multi-view Trajectory Contrastive Learning for Online Multi-object Tracking,” *arXiv e-prints*, p. arXiv:2203.14208, Mar. 2022.

[23] Clearpath.Inc, “Jackal: Unmanned Ground Vehicle,” Clearpath Robotics. Available at https://clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/.

[24] “move_base,” ROS Wiki. Available at http://wiki.ros.org/move_base.

[25] P. Fankhauser, M. Bloesch, and M. Hutter, “Probabilistic terrain mapping for mobile robots with uncertain localization,” *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 4, pp. 3019–3026, 2018.

[26] P. Fankhauser, M. Bloesch, C. Gehring, M. Hutter, and R. Siegwart, “Robot-centric elevation mapping with uncertainty estimates,” in *International Conference on Climbing and Walking Robots (CLAWAR)*, 2014.

[27] A. Fournier, D. Fussell, and L. Carpenter, “Computer rendering of stochastic models,” Communications of the ACM. 25 (6): 371–384. doi:10.1145/358523.358553. (1982).

[28] A. Smith, “Tint Fill. SIGGRAPH ’79,” Proceedings of the 6th annual conference on Computer graphics and interactive techniques. pp. 276–283. doi:10.1145/800249.807456. (1979).

[29] “rosbag,” ROS Wiki. Available at http://wiki.ros.org/rosbag.

[30] A. Aydemir, P. Jensfelt, and J. Folkesson, “What can we learn from 38,000 rooms?” Reasoning about unexplored space in indoor environments. In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. Vilamoura, Algarve, 4675–4682,(2012).

[31] “OpenCV,” Available at https://opencv.org/.