Inverse Reinforcement Learning for Autonomous Ground Navigation Using Aerial and Satellite Observation Data

Yeeho Song

CMU-RI-TR-19-24

May 20th 2019

Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Jean Oh, Chair
Luis Ernesto Navarro-Serment
Arne Suppe

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

Keywords: Inverse Reinforcement Learning, Learning from Demonstration, Autonomous Navigation, Conditional Learning, Multimodal Learning

Abstract

Inverse reinforcement learning (IRL) is a supervised learning paradigm where a learner observes expert demonstrations to learn a hidden cost function in order to mimic the expert's behavior. Eliminating the need of elaborate feature engineering, deep IRL approaches have been gaining interests in various problem domains including robot navigation. With the advent of low-cost drones and satellite services increasing the availability of 2D and 3D data over large area, there has been growing interest in end-to-end autonomous navigation systems which uses aerial and satellite information where prior knowledge of the environment is often outdated or no longer valid. In this paper, we propose a Conditional Multimodal Deep Inverse Reinforcement Learning approach that uses a deep neural network to learn sophisticated features for generating cost maps for multiple driving behaviors in the global planning context while utilizing both 2D and 3D data to accomplish such a task.

Acknowledgement

I would like to especially thank my advisor and head of MSR thesis Committee, Dr.Jean Oh. I am truly grateful to have her as my advisor who is not only a talented researcher but also a great person.

I also thank Dr.Luis Ernesto Navarro-Serment whose experience and feedback on field robots, as well as his wonderful personality, helped my research in various aspects, especially on testing the algorithms on Husky the robotics platform.

Thanks also goes to a doctoral student Arne Suppe who provided great technical insights in to the research, especially on resource management issues such as parallelization and batch updating neural networks for optimal performance.

Finally, I would like to thank Dr.Xavier Perez Sala for his expertise and support in technical and theoretical problems, and my fellow students at the BIG (Bot Intelligence Group) for such a interesting 2-years of my life!

This material is based upon work supported in part by the Air Force Office of Scientific Research under award number FA2386-17-1-4660, and through collaborative participation in the Robotics Consortium sponsored by the U.S Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement W911NF-10-2-0016. This work should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

This research was conducted using computing resources at the National Robotics Engineering Center, SPEEDY1, HINTON, and BIG.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In the recent years, fast-growing interests in self-driving cars have contributed to various technological advances in autonomous navigation on the roads. Whereas highly accurate prior maps are generally available for path planning in such urban environments [33, 48, 49], the information on drivable surfaces in off-road environments is generally unavailable a priori or subject to change frequently. In the off-road driving scenarios such as military or disaster relief operations, it is needed to predict traversability using observation data such as aerial view images and data from on-board sensors. With the advent of low-cost sensors and drones [50] as well as several satellite imagery service providers providing near real-time imagery for disaster zones at low or no cost to disaster relief units [1, 12, 23, 35], a huge amount of high-resolution overhead multimodal data consisting of imagery and 3D data can be collected in a short time; however, it is challenging to utilize such a gigantic amount of raw data to produce useful information for operators. Furthermore, supporting multiple navigation modes in such unstructured environment poses technical challenges. For instance, depending on the type of operations, a different path can be chosen according to the criteria beyond basic traversability, *e.g.*, in order to deliver time-critical supplies and medicines, it is desired to find a quickest path by cutting through vegetation, unpaved surfaces, or destructible objects, whereas, in a stealth operation, covert paths such as hugging around buildings would be preferred. In this context, we address the following research question: given overhead mutlimodal observations, how can we generate customizable traversability cost maps for path planners to accommodate various requirements and preferences?

To answer such challenges, the idea of learning from demonstration, in particular in the form of Inverse Reinforcement Learning (IRL) [36], has been successfully used to predict traversability cost maps for path planning in urban settings [56] as well as off-road settings where over-sized vehicles could drive autonomously for tens of kilometers in distance [41, 46]. In the case of Deep IRL [52], deep networks were used to replace human feature engineering associated with classical IRL algorithms by using raw LIDAR input, showing that raw sensor data can indeed be used for autonomous navigation tasks.

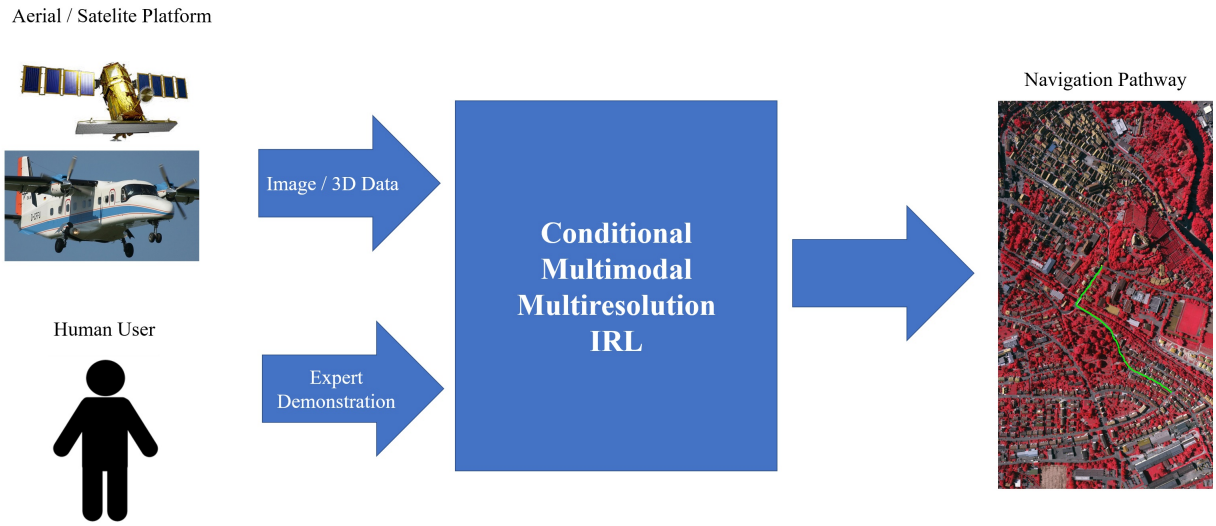


Figure 1.1: Diagram showing the resulting pipeline of the proposed research. Given with Earth Observation data consisting of 3D Data / Imagery from Aerial and Satellite platforms, our algorithm can learn from expert demonstration to recover hidden cost map for ground navigation and use the costmap generate waypoints for autonomous ground navigation as marked in green in the aerial imagery to the right.

1.2 Research Goals

Building upon the success of IRL in navigation, this thesis aims to apply IRL to multimodal earth observation data while supporting versatile navigation behaviors in order to meet the specific needs in highly unstructured scenarios such as postdisaster response as described in Figure 1.1. In this thesis, I will address the following problems of 1) decreasing computation load of Deep IRL to work with large-scale data from earth observation systems, 2) adapting [53] to work with overhead imagery, 3) how to allow single set of networks structure and weights to support more than one navigational behavior, 4) how to utilize more than one modality of data to approximate cost function, and 5), application of all these findings in the real mobile robot vehicle navigation problem.

Chapter 2

Problem Formulation.

We formulate the semantic navigation problem as an IRL problem. Given a start and an end locations in a 2D space, the goal of our navigation problem is to find a path for a ground vehicle that is optimal with respect to a certain objective function, *e.g.*, finding the shortest path or the safest one. In this thesis, we specifically address such a problem in unknown or partially known environments when the input is earth observation data such as satellite imagery or aerial LIDAR scan.

2.1 Markov Decision Process

A Markov Decision Process (MDP) represents a sequential decision-making problem in a stochastic environment. An MDP is defined as a tuple, $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, r, [\gamma]\}$ where $\mathcal{S}$ represents a set of states; $\mathcal{A}$, a set of available actions; $\mathcal{T} : S \times A \times S \to \mathbb{R}$, a stochastic model that represents state transition; r: a reward function (or "cost" in this paper) given from an environment as reinforcement; and $\gamma \in (0, 1]$: an optional discount factor specifying that rewards given in the future may not be as worth as immediate ones. A solution to an MDP, as a result of reinforcement learning (RL), is an optimal policy π^* that maps each state to an optimal action with respect to a discounted long-term expected reward.

2.2 Inverse Reinforcement Learning

Given a forward RL problem in an MDP $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, r\}$, let $\mathcal{M}^{-r}$ denote its inverse RL problem where the cost function r is unknown. The goal here is thus to learn the hidden cost function r from set $\mathcal{D}$ of expert demonstrations.

Here, we analyze the complexity of IRL implementations using two commonly used MDP solvers: value iteration and policy iteration [39]. In Maximum Entropy IRL[56] and MEDIRL[52] framework which this paper is based upon, IRL algorithms iterate the following two steps until convergence: the cost estimation step recovers the underlying cost function, given demonstrations $\mathcal{D}$; the policy estimation step uses the current estimate of the cost function to solve the forward MDP, which provides a policy and expected state visitation frequency (SVF), denoted

by $\mathbb{E}[\mu]$. Finally, the comparison between demonstrations $\mathcal{D}$ and expectations $\mathbb{E}[\mu]$ drives the parameter search in the cost estimation step.

The complexity of *one iteration* of the value and policy iteration algorithms is $\mathcal{O}(|\mathcal{A}||\mathcal{S}|^2)$ and $\mathcal{O}(|\mathcal{A}||\mathcal{S}|^2 + |\mathcal{S}|^3)$, respectively [26]. Additionally,the number of iterations required to converge can grow exponentially if the discount factor approaches 1. Once an optimal policy is obtained, $\mathbb{E}[\mu]$ is estimated using dynamic programming in polynomial-time [27].

2.3 Path Planning as Search

Path planning is closely related to graph search when states are represented as nodes connected by edges. Notable algorithms in this field are the well-known Dijkstra algorithm [13], A* [24], and D* [47] for dynamic environments. Unlike value or policy iteration, path planning algorithms don't need to visit all states in $|\mathcal{S}|$, but explore just enough to find the cheapest path between two given states. Therefore, the complexity of a path planner depend on heuristic it uses, expressed as a function of explored nodes. The complexity of A* is $\mathcal{O}(b^d)$, where b is the branching factor and d is the depth of the goal node. In the worst case scenario where A* is essentially same as Dijkstra, the complexity peaks at $\mathcal{O}(|\mathcal{S}|^2)$.

Interesting connection is that any MDP can be framed as a stochastic the shrotest path problem[6]. As such, there has been many works to exploit the relatively low computational complexity of path planners for solving MDPs[19][32].

Chapter 3

Related Works

Our approach is related to existing works in representation of cost functions, conditional learning and multimodal learning.

3.1 Representation of Cost Functions

The IRL approaches [41, 56] have shown great successes in learning human driving behaviors. The cost functions in early works are linear combinations of input features that are elaborately engineered. To support complex control problems, nonlinear cost functions have been studied, *e.g.*, using Gaussian Processes [28], Dirichlet Processes [10], or function approximation for large state-spaces [29]. The work that is most closely related to our research is Maximum Entropy Deep IRL (MEDIRL) [52] where Maximum Entropy IRL [56] was generalized to a deep learning networks; such an end-to-end learning idea has been applied to generate a costmap using 3D data from a vehicle-mounted LIDAR sensor in a semi-urban environment [53].

3.2 Approximate solutions to an MDP

In general, most IRL algorithms including Maximum Entropy IRL(MaxEnt)[56] and MEDIRL[53], iterate the following two steps until convergence: the cost estimation step recovers the underlying cost function, given demonstrations; the policy estimation step uses the current estimate of the cost function to solve the forward MDP, which provides a policy and expected state visitation frequency (SVF).

For example, Relative Entropy IRL(RE-IRL) [9] uses model-free approach while [20] uses policy optimizer to solve MDP, and original Maximum Margin paper [41] used A* planner[24]. Such options allow more efficient solution to solving MDPs and allows for faster training. For example as path planning algorithm such as Dijkstra algorithm [13], A* [24], and D* [47] Planners will explore less or equal number of states compared to value or policy iteration, they can generate trajectory at much less computation cost.

3.3 Multiresolution Networks

For end-to-end IRL navigation for learning, the algorithm has to extract deep semantic understanding from raw data. In such context, it is often desirable to process image data across various resolutions of data and area covered by convolution kernels to better understand the environment with varying viewpoint for convolution layers. In one such case, U-net [42] has multiple pooling and upsampling operations between convolution operations, allowing the data to gradually decrease and then increase the resolution from input to output layers. Such an operation allows the algorithm to capture context at low-resolution operations and use higher-resolution operations for precise segmentation, *e.g.*, in medical images [42] or noise filtering with radio astronomy [2]. For the case of Multi-scale Network [4], the algorithm analyzes encoded information from aerial imagery with three independent branches of convolution operations with the varying kernel size. Using these three branches to predict semantic classification at different kernel sizes, the network can work as if combining results of three different network, exploiting the right size of kernel for extracting specific semantic information. Finally, unlike the Multi-scale Network that only changes the scale of viewing area of convolution operations by varying size of kernels, MRNet [3] combines both ideas of varying resolution and viewing area through the use of pooling and upsampling operations. In MRNet [3], an input image is first processed via a fully convolutional network (FCN). The results from the FCN are pooled down in parallel into multiple branches at various resolutions that consist of convolution operations. The results are later concatenated to match the same size as the output of the convolution operations, and sent through another convolution operation to generate a cost map for autonomous ground navigation. By passing through parallel convolution operations at different resolutions, hierarchical contextual information can be extracted from images. PSPNet [55] also uses a similar structure for scene parsing tasks with promising performance.

3.4 Conditional Imitation Learning

Conditional Imitation Learning (CIL) [11] is a unique form of multimodal learning that by giving an expert demonstration and a descriptor explaining the behavior of the demonstration, a deep neural network can learn multiple navigation behaviors at the same time on a single set of network weights. In [11], CIL was trained with expert driving demonstration comprised of ground level images, various sensor data to learn three different behaviors of going straight, turning left, and turning right at an road intersection by using an extra input to describe the behavior of the expert demonstration. However, CIL's main difference from most other multimodal learning algorithms is how the descriptor is merged with the main data. In the case of [18], [34], [7], and [54] was concatenated with the main data in the form of fully connected layers. Branched approach of CIL [11] uses first part of the network to first process image and pass the extracted deep network to corresponding deep action network for going straight, turning left, and turning right depending on the descriptor input. Such approach showed better results than Command Input approach for CIL [11] where descriptor was simply concatenated as in other multimodal learning networks [18], [34], [7], and [54].

3.5 Deep Multimodal Learning

With a success in [37] that used two different input modalities of audio and video for better speech classification, Deep Multimodal Learning has gained attention in recent years on how deep neural network can utilize more than one modalities of observation to understand an environment. With a diverse field of applications from using MRI and PET for medical images [45] to autonomous navigation using images and LIDAR data [30], deep learning with Earth Observation (EO) data is one major field that can greatly benefit from deep multimodal learning. For instance, while a grassy hill and a swamp covered with dense vegetation may look similar as featureless grass area, and concrete structures, pavement, and roads may all look like square grey objects, deep networks using 3D information such as LIDAR data can disambiguate them. As such, Multimodal Deep Learning for better semantic segmentation using the EO data showed great promises in recent years such as [4] and [38] that used Infrared-Red-Green Imagery, Digial Surface Model (DSM), Normalized Digial Surface Model (NDSM) and Normalized Difference Vegetation Index (NDVI) to better understand an environment.

Chapter 4

Approach

We propose a Conditional Multimodal Deep Inverse Reinforcement Learning approach that can learn multiple driving behaviors via conditional learning and exploit information form 2D and 3D data via multimodal learning. We summarize existing work on which our approach builds on in Section 4.1, followed by our contributions in Section 4.2 and onward.

4.1 Preliminaries

Our approach generally follows the principle of maximum entropy as in [56] where we maximize the likelihood of demonstrations with the highest entropy. Following the derivations directly from [52], this problem can be defined as computing the joint probability of observing the demonstrations $\mathcal{D}$ under parameters θ given cost r as follows:

$$\mathcal{L}(\theta) = logP(\mathcal{D}, \theta|r) = \mathcal{L}_{\mathcal{D}} + \mathcal{L}_{\theta} = logP(\mathcal{D}|r) + logP(\theta)$$

By defining the cost function r as a parameterized function over features f, such that $r = g(f, \theta)$, we develop a neural network representing the function. To compute the gradient of the loss of demonstration $\mathcal{L}_{\mathcal{D}}$, the backpropagation algorithm can be applied and the gradient can be rewritten as the product of the gradient of the loss with respect to the cost, $\frac{\partial \mathcal{L}_{\mathcal{D}}}{\partial r}$, and the gradient of the cost with respect to the network parameters, $\frac{\partial r}{\partial \theta}$. In the case of a linear function, the gradient of demonstration with respect to the cost is the difference in feature frequencies between empirical counts and the learner's expectation [56] as follows:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial \mathcal{L}_{\mathcal{D}}}{\partial r}\frac{\partial r}{\partial \theta} = (\mu_{\mathcal{D}} - \mathbb{E}[\mu])\frac{\partial}{\partial \theta} r(\theta)$$

where $\mu_{\mathcal{D}}$ is the State Visitation Frequency (SVF) of the expert demonstration and $\mathbb{E}[\mu]$ is the expected SVF of the learner. With such derivations, we backprogate the neural network parameters θ to minimize loss, $\mathcal{L}$

4.2 Approximation of Forward Solver

Solving the forward problem during IRL can be expensive [20]. Following ideas in [19, 32, 43] that MDPs can be solved more efficiently with path planners, we use A* planner [24] to approximate Forward Solver during IRL. This approach was also implemented successfully with the case of Maximum Margin Planning [41].

In 2D navigation, the trajectory generated via A* [24] can be viewed as sampling one of the most probable trajectories in the SVF distribution. While this is an approximation of the true SVF at a given iteration, such an approximation is acceptable as A* is likely to visit all states with a distribution similar to the actual SVF during the long training process of deep networks that involves numerous iterations and updates. Marking only the states with highest SVF that would dominate the training process, A* can greatly enhance the speed of training compared to other methods such as Value Iteration [5] used in MEDIRL [53]. As deep networks generally need a large amount of data for training, this speeding up is crucial in time-constrained applications such as post-disaster assistance.

As a result, the update rules for our approach is an approximated version of MEDIRL [53] that is computationally much lighter. At each training iteration, our approach first uses the current cost function to solve the forward problem, *e.g.*, by using an A* planner, the algorithm computes the learner's trajectory (Line 6). Next, the algorithm compares the learner's trajectory with those of the experts by computing the difference in state visitation. (Line 7). Finally, it backpropagates the loss and updates the neural networks' parameters (Line 8, 9).

Algorithm 1 The FA*ST Deep IRL Approach

1: **Input:** IRL $\mathcal{M}^{-r}$, Demonstrations $\mathcal{D}$
2: **Output:** Recovered cost function weights θ
3: θ^1 = initialize_weights()
4: **for** *n=1:N* **do**
5: r^n = feed_forward_nn(f,θ^n)
6: $\mathbb{E}[\mu^n] = A^*$_planner($\mathcal{M}^n$)
7: $\frac{\partial \mathcal{L}_{\mathcal{D}}^n}{\partial r^n} = \mu_{\mathcal{D}} - \mathbb{E}[\mu^n]$
8: $\frac{\partial \mathcal{L}_{\mathcal{D}}^n}{\partial \theta_{\mathcal{D}}^n}$ = back_propagate_nn(f,θ^n, $\frac{\partial \mathcal{L}_{\mathcal{D}}^n}{\partial r^n}$)
9: θ^{n+1} = update_nn_weights(θ^n, $\frac{\partial \mathcal{L}_{\mathcal{D}}^n}{\partial \theta_{\mathcal{D}}^n}$)

4.3 Network Architectures

To extract high-level feature information directly from aerial and satellite imagery for deep IRL, we design network architectures to utilize multiple pooling operations to allow the neural networks to process an input image at multiple resolutions.

The idea is based on the assumption that certain features may appear more pronounced to neural networks (and can thus be more easily trained) depending on two factors: the resolution

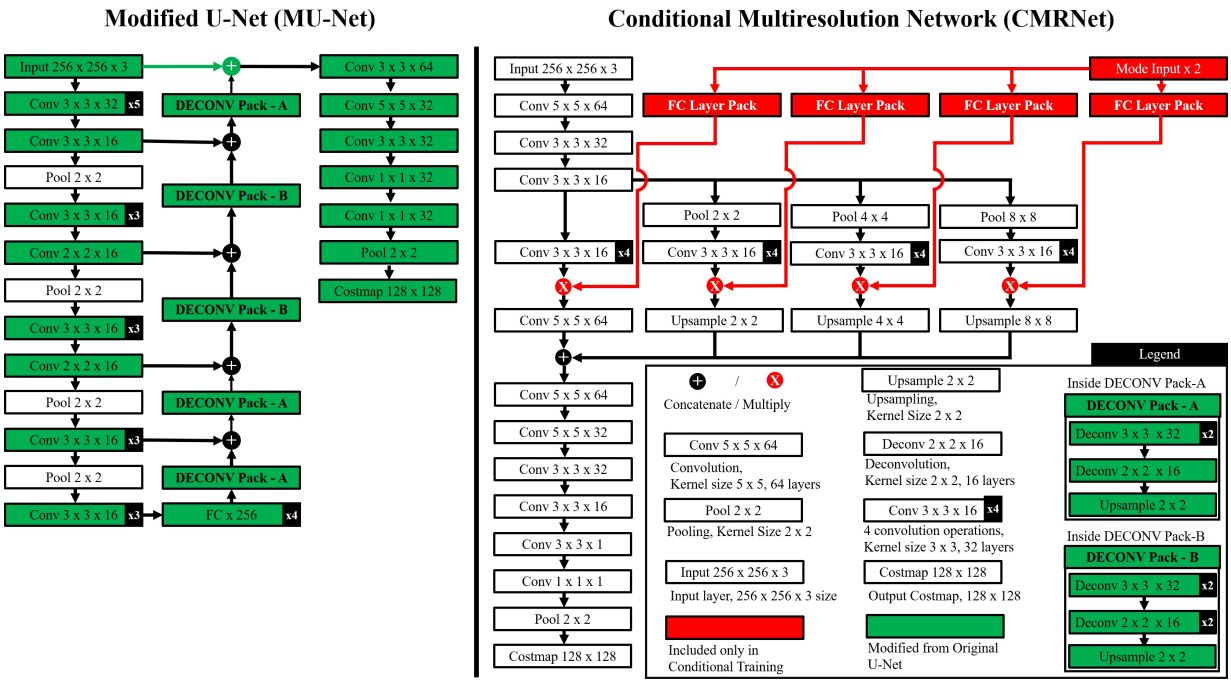


Figure 4.1: Proposed networks: Modified version of U-Net[42] (MU-Net)(left) and Conditional Multiresolution network (CMRNet)(right). Modifications from original U-net[42] is marked in green. Note that Multiresolution Network(MRNet) in this paper refers to CMRNet Network without the conditional parts outlined in red.

of an input image and the size of a kernel. For instance, to differentiate forest from grass fields, we need the texture information that may only be available in a high-resolution view.

Tangentially, objects that look similar in terms of color and texture can be indistinguishable in a small, isolated patch, and require wider context information to discern, *e.g.*, a grey patch of rough texture can be either concrete road surfaces or flat-top grey building roofs depending on size, shape, and what surrounds it. To learn such context features, we need larger kernels in the convolutional layers.

Based on this intuition, we propose two networks that utilize multiple pooling operations and concatenations in order to capture contexts such as texture, geometric, or spatial information, within a reasonable network size. By varying the resolution of an image within the network through the use of multiple pooling operations, the part of the network with higher-resolution images is responsible for picking up texture-related information, while that with lower-resolution images is responsible for geometric, spatial and contextual features.

Under the idea of varying resolutions within networks, we modify the U-Net architecture [42] and introduce Multi-resolution networks (MRNet) as shown in Figure 4.1. The details of the network structures can be found on Section 8.3.

4.4 Modified U-Net (MU-Net)

Originally used for medical imagery, U-Net [42] is a Full Convolution neural network using multiple pooling operations that showed good performance even with small datasets in the field of medical imaging as well as versatility of being applied in various different fields, such as radio astronomy. [2]. The original U-Net [42] starts with 64 channels per convolutional layer; the number of channels doubles every time the network pools down to a smaller height and width. As shown in Figure 4.1, in an attempt to decrease the number of parameters and introduce constraints in the network for simpler training, we reduce the number of channels to 16 or 32, while increasing the number of convolution operations at each resolution from 2 to 3.

In addition, we use a Fully Connected (FC) layer at the lowest resolution since the layer is only 4x4x16 in size; when compared to original U-net's size of 30x30x1024, it is small enough for FC operations to be used. Whereas U-Net uses a larger size input to utilize surrounding pixels in place of zero-padding, we match our MU-Net network output to have the same height and width as a network input such that MU-Net can be used interchangeably with other networks in our deep IRL framework depending on the problem setup and performance requirements.

4.5 Multiresolution Networks (MRNet)

We introduce Multiresolution networks (MRNet) where a set of pooling operations is performed after the first convolutional layer to extract the features at a gradually lower resolution simultaneously as shown in Figure 4.1. The main difference between MRNet and the MU-Net is that, whereas MRNet has a *parallel* structure where input is processed at multiple resolutions along separate pathways and than later concatenated all at once, MU-Net has a *serial* structure where an input is processed at multiple resolutions along a single pathway in a sequential order and a concatenation operation is done for each resolution at a time.

When compared to sequential pooling operations, parallel pooling models tend to be less deep; therefore, parallel models might have issues with supporting complex cost function shapes. At the same time, this feature also helps the models to avoid overfitting. Intuitively, the parallel model has fewer operations for the extracted features to go through before the final output layer compared to the sequential model, potentially reducing the chance of being affected by additional operations.

4.6 Pooling Operations for Resolution Discrepancy

Our proposed methods have pooling operation at end of the network for faster training. For input layers, high resolution imagery is desired to have more information about an environment. However, such high resolution often leads to large search space, making it computational expensive to solve MDP. As a technical solution to solve the discrepancy between resolution requirements, our proposed methods have pooling methods such that input and output resolutions can be tuned separately to maximize speed and performance of training. While such pooling operation results in loss of information for cost maps, such loss is acceptable as global planners has much coarser

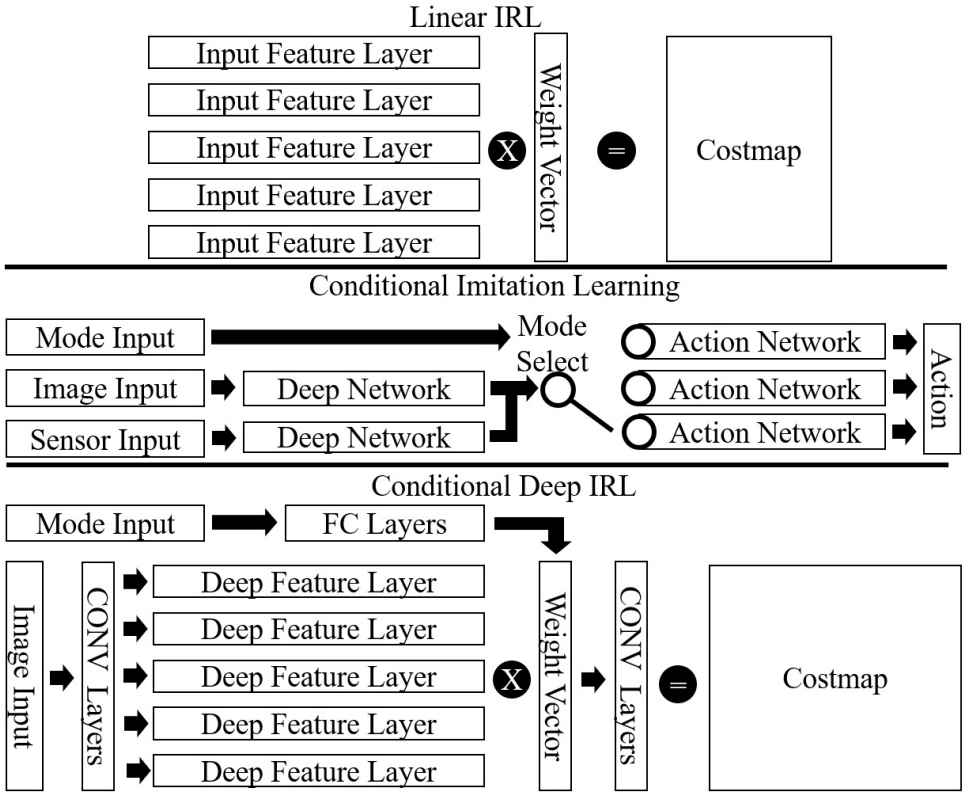


Figure 4.2: Conceptual comparison of Linear IRL, Conditional Imitation Learning, and Conditional Multiresolution Network.

resolution requirement than those required from input images.

4.7 Conditional Multiresolution Network

In traditional linear IRL architectures based on handcrafted feature inputs, each navigation behavior is trained independently [8]. As shown in Figure 4.2, these linear IRL algorithms take an input of handcrafted feature layers, multiply each layer with a corresponding trained weight, and add the results to generate a costmap.

Similarly, Conditional Multiresolution Network (CMRNet) first passes the input image through the multiresolution structure to generate multiple deep feature layers. Each layer would then be multiplied by a set of weights, concatenated, and passed through another set of convolutional layers to generate a costmap. The set of weights for the case of Conditional Multiresolution Network would be generated using a set of Fully Connected Layers that takes an input as a 1-D vector of numbers describing each driving behavior in the dataset. Such a structure is analogous to the structure used in Conditional Imitation Learning (CIL) [11] in a sense that the mode input dictates which part of the network would be used to generate a desired behavior. As illustrated in Figure 4.2, whereas the CIL approach hard-wires each navigation mode to which branch of the network to use, CMRNet represents the mode decision as a weight vector and learns the weights

as opposed to having a separate branch for each navigation mode as in CIL.

Figure 4.1 shows the resulting CMRNet that can be trained to extract texture, spatial, and contextual information from images and learns how to weight each deep feature layer and look at the concatenated result of multiplication operations to recover a costmap according to each different type of expert behaviors.

4.8 Multimodal Multiresolution Network

To utilize both imagery and 3D data, we propose a multimodal multiresolution networks architecture for MaxEnt Deep IRL. Our network structure is inspired from MRNet [3] that offers the state-of-the-art performance on learning complex navigation behaviors using aerial and satellite imagery data in a variety of urban, outdoor, and post-disaster scenarios. In MRNet, whose structure follows the similar principle to those found in [55], the input data is first passed through a set of initial convolutional layers, the results from which are then pooled at different resolutions to be processed in parallel through a set of convolutional layers. The deep features extracted from this operation is concatenated and passed through a series of convolution operations to generate a costmap.

Whereas MRNet was designed for taking large-sized, high-resolution images as inputs, the proposed architecture is specifically focused on how to fuse the 3D data with 2D image by varying the network structure, fusion techniques, and grid resolution of the 3D. To verify the pros and cons of each methods and network structure, the algorithms were kept as modular as possible, switching out parts and data as need to find the correlation between algorithms and their performances. Figure 4.1 show network structures explored with detailed explanations included in the following sections.

Multimodal Image Prioritized Network (MIPNet) and Multimodal 3D Prioritized Network (M3PNet): Inspired by from MRNET [3], we propose a multimodal deep network structure where we first pass one modality through a multiresolution structure and the other modality through convolution operations. The deep features extracted from the two modalities are than concatenated and passed through a series of convolution operation to recover cost function. While the two modalities are supposed to complement each other and therefore difference in performance is not greatly expected, we test the network structure for which modality benefits more from using multiresolution operation for deep feature extraction. For this purpose, we devise Multimodal Image Prioritized Network(MIPNet) and Multimodal 3D Prioritized Network(M3PNet), as marked in blue in Figure 8.4, where image is passed through multiresolution structure in the former and 3D information is passed through multiresolution structure for the later.

Multimodal Multiresolution Network (M2R): As multiresolution network uses parallel convolution operations at multiple resolutions to find deep features, we believe that both 3D and image information can benefit from multiresolution operation to improve network performance. Based on this idea, we test Multimodal Multiresolution Network which uses multiresolution operation for both modalities before concatenating the deep features, as marked in Red in Figure 8.4.

Early vs Intermediate Fusion: The network structures proposed above follows a fusion

technique commonly referred as Intermediate Fusion [40]. Contrast to our approaches where data from different modalities are fused as deep features along the network, Early Fusion is a method which different modalities are merged together at earlier stages of the framework before being passed on to deep networks. The initial estimate is that intermediate fusion would be more applicable to our case as fusing data in the later stages of the framework is quite popular [40] for cases where input modalities greatly differ. This is because intermediate fusion allows for having two separate network structures trained for extracting deep features from input rather than having a single network structure that can cope with multiple different input modalities at the same time. To verify our initial estimate, we compare the results of the intermediate fusion model of Multimodal Image Prioritized network with those from early fusion model of Multimodal Early Fusion Network (MEFNet) as marked in purple in Figure 8.4.

Single-modal Multiresolution Network: To verify the performance differences in having a multimodal approach, we use a Single-modal Multiresolution Network (SMRNet), as marked in Green for Figure 8.4 for single modality input. Inspired from MRNet [3], Single-modal Multiresolution Network shares network structure with rest of the networks proposed in this paper, making it a valid network for comparison.

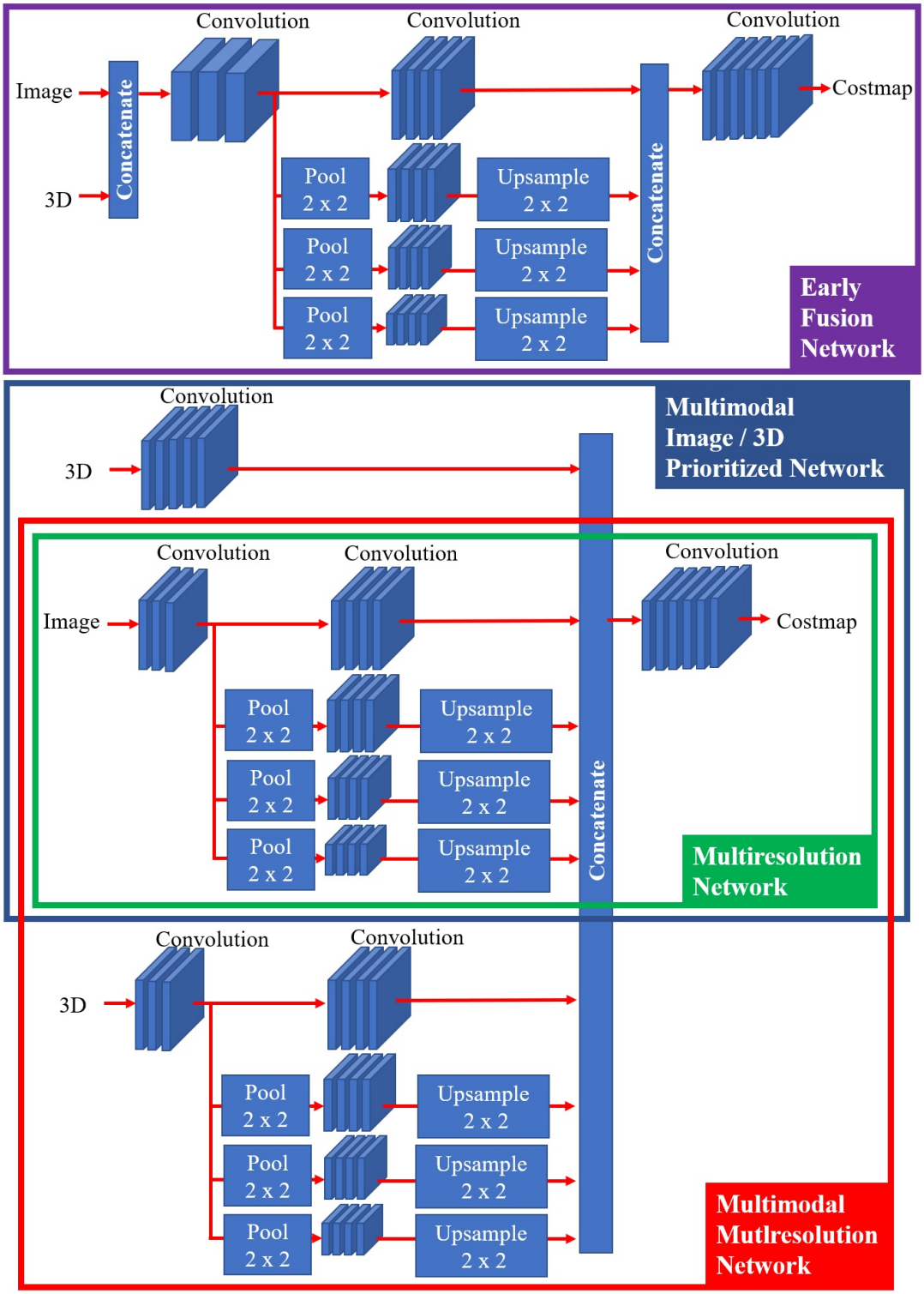


Figure 4.3: Diagram showing network structures used in the paper. Blue box marks the Multimodal Image / 3D Prioritized Network, green marks Singlemodal Multiresolution Network, purple marks the Early Fusion Network, and red box marks the Multimodal Multiresolution Network

Chapter 5

Experiments

5.1 Datasets

Image data: We collected a set of aerial and satellite imagery data in both real and synthesized environments from the following publicly available sources: ISPRS [25], Google Earth Pro and associated image providers [21][22][17][16], and CARLA [14]. The ISPRS set covers various locations in Vaihingen, Germany and provides red, green, and infrared channel inputs at 0.4 m / pixel resolution. All other images were captured with red, green, and blue channels at 0.8 m / pixel. The two Fukushima (FKS) sets were taken in the same locations within 10 km of Fukushima Daiichi Nuclear Power plant in Japan before and after the 2011 Tohoku Earthquake and the subsequent tsunami; FKS-1 contains the images from November 2009 before the earthquake, and FKS-2, March 2011 after the earthquake. All images were captured at bright lighting conditions. Two exceptions to that are: 1) FKS-1 was taken with the dark lighting conditions and 2) CARLA data was taken under 6 different weather conditions supported by the simulator such as rainy day, late after noon, sunny noon, etc. Each image was cropped to 256 x 256.

3D data: To compare with 3D data-based algorithms and proposed multimodal algorithms, 3D data was given for ISPRS and CARLA datasets at resolutions of 0.2, 0.4, 0.8, 1.6, 3.2, 6.4m / pixel resolution. For ISPRS dataset, the 3D data was collected overhead from a Airborne Laser System while for CARLA it was collected via LIDAR simulator collecting data on top of a moving vehicle inside the simulation.

Semantic data: To compare with MaxEnt IRL[56] that uses feature inputs, semantic labeled version of ISPRS datasets was also created with 0.8m / pixel 256 x 256 size. The 6 semantic labels included in the dataset are Impervious Surfaces, Building, Low Vegetation, Tree, Car, and Clutter / Background [25].

Expert demonstrations: We collected expert demonstrations for the following navigation behaviors: normally (default), safely or quickly (fast), driving on the center of the road or on the edge of the road. The default mode, driving normally, represents the behavior of staying on the center of the road (or other types of drivable surfaces) to maximize the distance from potential obstacles. The default mode was collected for all of the sets. The customized behaviors were collected in the ISPRS set to evaluate the learning of multiple navigation behaviors,

such as *safely* staying on paved surfaces while moving through *center* of such areas to keep

Table 5.1: Size of Datasets and Area of Imagery

Name	Size of Dataset				Location
	Source	Augmented	Train	Test	
CARLA*	150	1200	1000	200	CARLA Town 01
ISPRS*	144	1152	960	192	Vaihingen, Germany
PIT	144	1152	960	192	Pittsburgh, US
FKS-1	144	1152	960	192	Fukushima, before tsunami
FKS-2	144	1152	960	192	Fukushima, after tsunami
MOUT	128	1028	800	224	MOUT** Training Facility in the US

* 3D data available at resolution of 0.2, 0.4, 0.8, 1.6, 3.2, 6.4m per pixel

**MOUT : Military Operations in Urban Terrain.

a distance from obstacles (SafeCenter), *safely* staying on paved surfaces and hugging *edges* of obstacles in such areas (SafeEdge), cutting through unpaved surfaces such as vegetation to get to the goal *fast* while moving through *center* of such areas to keep a distance from obstacles (FastCenter), and cutting through unpaved surfaces such as vegetation to get to the goal *fast* and hugging *edges* of obstacles in such areas (FastEdge). As for the CARLA dataset, the driving behavior was to stay on one side of the road without crossing the center line. For each image, given a start and an end positions and a navigation mode, a human expert drew a desired path over the image.

Core dataset: The dataset consists of 4 subsets for the four navigation behaviors on the Vaihingen imagery from ISPRS and 5 subsets for driving normally on the rest of imagery data as listed in Table 6.3.

Data augmentation: We augmented the core dataset using the process of flipping and rotating, resulting in the dataset increased in the size by the factor of 8.

5.2 Evaluation Metric

We use Modified Hausdorff Distance (MHD), an evaluation metric for comparing trajectories generated by planners learn from demonstrations [15, 44, 51], which is defined as follows:

$$\texttt{MHD}(\zeta_D, \zeta_L) = [\sum_{i=1}^{N} d(\zeta_L(i), \zeta_D) + \sum_{i=1}^{N} d(\zeta_D(i), \zeta_L)] * \frac{1}{2N} \tag{5.1}$$

where $(d(\zeta_L(i), \zeta_D))$ is the minimum Euclidean distance from point i on the learner's trajectory ζ_L to the closest point on the expert's trajectory ζ_D, and vice versa.

5.3 Baselines

To evaluate our proposed approach, we compare the experimental results on the dataset described in Section 5.1 for Maximum Entropy IRL [41] and MEDIRL [53]. The former represents a well-known classical IRL algorithm while the later represents a state-of-the-art deep learning based IRL algorithm.

For a fair comparison, our implementation of Maximum Entropy IRL [41] was fed with ground-truth semantic labels as inputs as the algorithm was primarily designed to use labeled inputs. It is fair since algorithms exist to classify terrain types given aerial imagery, *e.g.*, achieving 88.5% accuracy for the images of ISPRS dataset [31]. Also, since MEDIRL [53] was originally designed for LIDAR data only, we use the CARLA simulator data where LIDAR data was also available. Our implementation of the algorithm was also used with 3-channel aerial / satellite imagery to show compare and contrast the pros and cons of LIDAR input against image input.

While feature engineering based on distance to closest label was tried for Maximum Entropy IRL [41], the results was not included in this report. This was because even with extensive efforts, it was difficult to find the right balance of engineered features to outperform our model across all datasets. This highlights how much expert supervision our end-to-end approach removes for autonomous navigation.

5.4 Training Parameters

All the codes used in this research was coded in Python3 / Tensorflow. Training was conducted with learning rate of 1e-5, batch size of 20, and for 500 epochs per session.

5.5 Resource Usage

This research was conducted using computing resources available with BIG, HINTON, and SPEEDY server computers at National Robotics Engineering Center (NREC). To best utilize the multi-core CPUs available with the computers, parallelization and batch update techniques were implemented to get results within reasonable time.

Solving IRL problem with deep neural network involves first passing input data through neural network to generate costmap, solve the forward problem of MDP, and backpropgating and updating the network. As solving the forward problem is a parallel process only dependent on costmap in question, parallelization was used to utilize multiple cores during this process for faster iteration.

Another technique used for faster training was batch updates. This was to solve the discrepancy between GPU and CPU resource availability. During training, Titan V and GTX 1080 GPU's installed on BIG allowed for network batch size of 20 to 32, with this number dropping down to 2 to 10 if the GPU resources had to be shared with other projects within the research group. Therefore, GPU availability was the choke point on how fast the training could run, as CPU resources were rarely congested and allowed 40 to 80 sessions of forward solving to run simultaneously. By separating CPU and GPU processes to use different batch sizes, resource utilization during training was maximized. Overall, this process of refining resource usage resulted in 4 to 20 times faster training speed compared to those of earlier versions of the code which had no advanced resource usage.

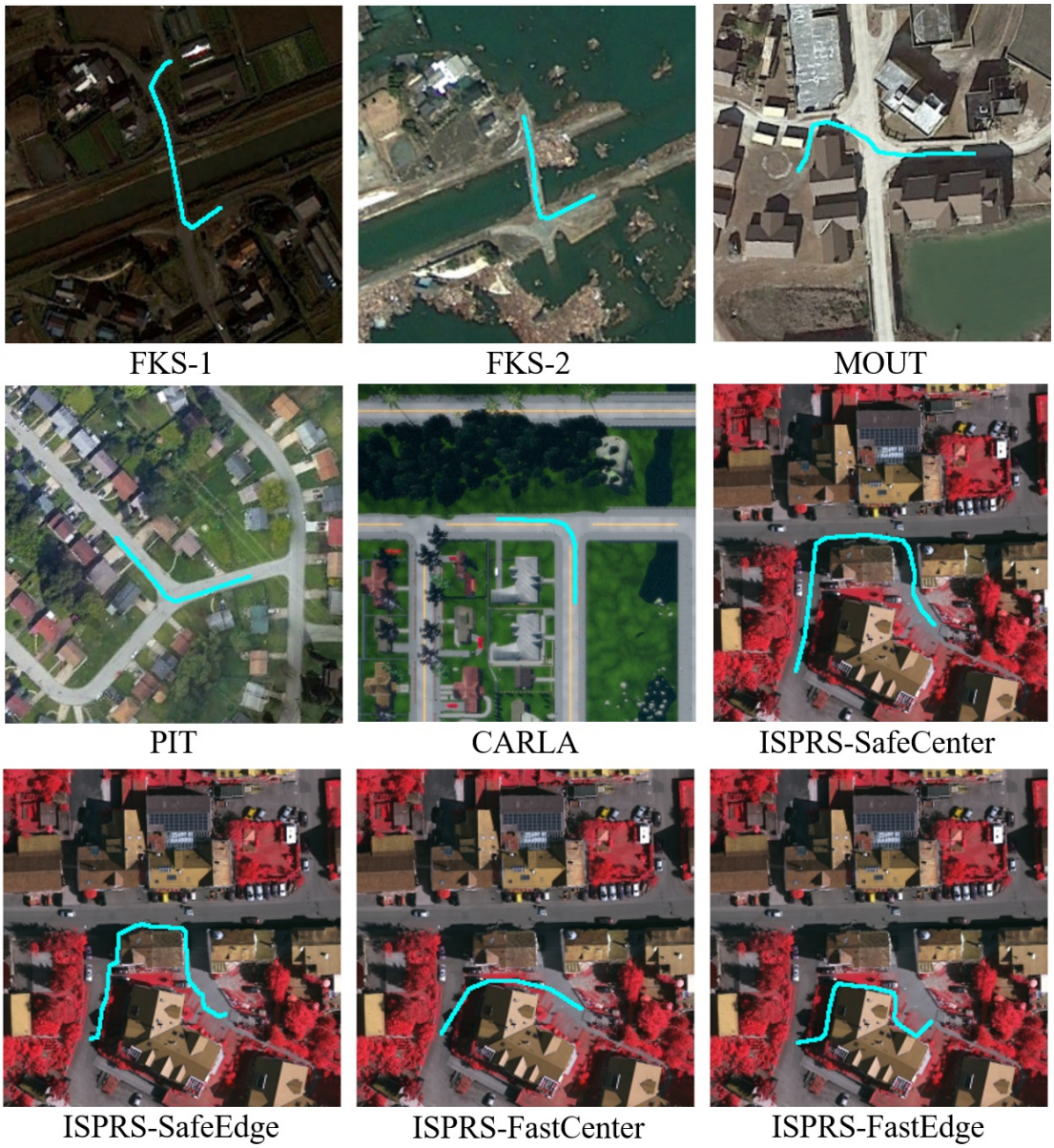


Figure 5.1: Characteristic samples of datasets used in this paper.Note how FKS-1 and FKS-2 covers same location before and after tsunami, while ISPRS datasets show different behaviors on a same location.

Chapter 6

Results

6.1 Overall Results

Table 6.1 shows the evaluation results for the two baseline approaches and the proposed deep IRL approach using two new network structures, namely MU-Net and MRNet. Our proposed methods outperform the state-of the art baseline approaches across all 9 datasets that include European cities, US suburbs, a synthesized environment, a simulated town, undeveloped villages, and an area struck by a natural disaster. We compare our approach against baseline approaches and their respective results to datasets highlights the strengths of our proposed methods as follows:

6.2 Comparison with MaxEnt IRL algorithm

Table 6.1 shows traditional IRL method such as MaxEnt IRL [56] can perform comparablely to MEDIRL [53] and match the performance of MU-Net for FastCenter. We note that such performance is only possible if appropriate feature input can be provided. For instance, MaxEnt IRL [56] does worse in SafeCenter compared to SafeEdge as while given semantic labels are suitable for supporting edge following behaviors, it doesn't mark where the center of the road is, making beahvior in the former dataset difficult to replicate.

Across all dataset, MRNet outperforms MaxEnt IRL [56]. This shows deep IRL's ability to extract features and complex cost functions through learning. While most application of MaxEnt IRL [56] involves feature engineering to overcome this issue, it was prohibitvely expensive to employ such method in this research. As each environment is slightly different, it was difficult, fine tunning features so that MaxEnt IRL [56] will outperform deep learning based IRL algorithms was difficult, highlighting how using deep IRL can greatly reduce human supervision.

6.3 Comparison with LIDAR based IRL algorithm

In the CARLA dataset, designed to contrast the strengths and weakness of 3D LIDAR data and 2D image data, we can see that the 2D images offers additional information not easily conveyed in 3D point cloud data. While the former can mark size and location of objects in the object,

Table 6.1: Average MHD of Various IRL Methods. Lower is Better(Bold: best, Underline: second)

Dataset	MaxEnt [56]	MEDIRL [53]	MU-Net	MRNet
SafeCenter	3.844	3.407	<u>2.512</u>	**1.783**
SafeEdge	2.994	4.064	**2.855**	<u>2.960</u>
FastCenter	<u>3.350</u>	7.055	3.822	**2.694**
FastEdge	<u>3.644</u>	5.812	5.821	**4.616**
CARLA	2.382	RGB : 1.792 / LIDAR : 2.690	**1.029**	<u>1.277</u>
PIT	N/A	2.698	**1.224**	<u>2.005</u>
FKS-1	N/A	1.955	**1.681**	<u>1.721</u>
FKS-2	N/A	6.665	**1.974**	<u>2.703</u>
MOUT	N/A	6.806	<u>6.167</u>	**3.298**

*MEDIRL was run with 3-channel imagery for all dataset, with another run for CARLA dataset using 3D LIDAR Data.

Input Image MaxEnt, SafeCenter MRNet, SafeCenter

Figure 6.1: Results of SafeCenter training for MAXENT IRL [56] and MRNET. With high cost regions marked in black, we can see that unlike MAXENT where transition between obstacles and driveable space is sharp, MRNET has gradual transition between the two regions, favoring center of the road then edge of roads.

the later also shows texture such as yellow center line marked in the center of the road. In the case of CARLA dataset where expert behavior was to stay on one side of a road, such additional information was beneficial in allowing image-based algorithm to easily assign high cost to center of the road as shown in Fig.6.2 much easier than LIDAR based methods which has no such cues.

6.4 Results of Various Multiresolution Networks

The overall results as shown in Table 6.3 implies that a deep, carefully designed network architecture is needed to learn a good costmap from high-resolution imagery. When compared to the network architecture used in MEDIRL [53], the proposed models use multiple pooling and upsampling layers to utilize both low-level local features as well as high-level contextual features. On one hand, the network needs the texture information from low-level layers to distinguish whether an object is vegetation or roads. On the other hand, the network needs to pay attention

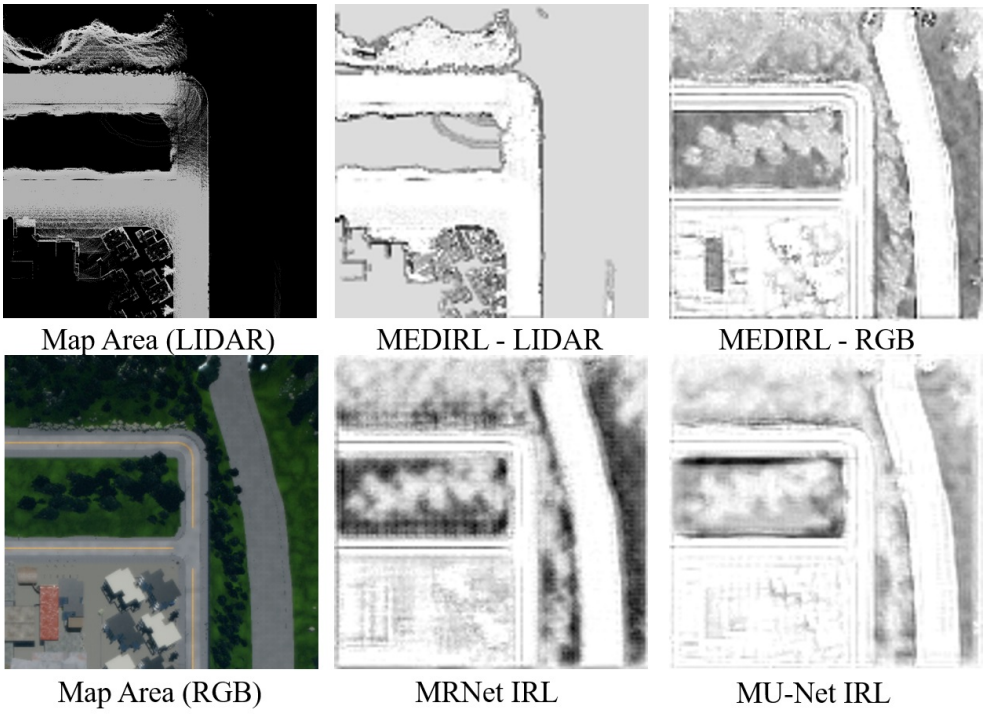


Figure 6.2: Results of CARLA Dataset with RGB and LIDAR input on MEDIRL [53], MRNet, and MU-Net. With high cost regions marked in black, we can see that LIDAR based method cannot detect center line easily as those based on RGB images.

to size and shape information to determine if a green patch is a grass field or a tree as drivable road surface may be hidden beneath a tree but not so likely under a grass field Sometimes, the network even needs to look at the surrounding context to see if a tree is standing in the middle of a forest or next to a road covering a traversable road underneath. By using multiple pooling, upsampling, and concatenation operations, the proposed networks can run convolution operations at a high resolution with a narrow viewing window to extract texture information, while running convolution operations at low resolution with a wide viewing angle to extract the spatial and contextual information.

While both MRNet and MU-Net are similar in their network sizes, their differences in performance show pros and cons of how pooling, upsamplig, and concatenation operations can be structured for an enhanced performance. In the case of MU-Net, the deep features extracted at the high-resolution layers are fed as inputs to the consecutive lower-resolution layers. Such *linear* structure induces high-resolution features to be weighted more than lower-resolution features. At the same time, the path that an input image goes through in the network pipeline can be extended, possibly leading the model to overfit the training set. Conversely, MRNet has a *parallel* structure where both high- and low- resolution convolution operations share the same input, and their outputs are concatenated together. In this way, the spatial and contextual information extracted from the low-resolution layers can be weighted fairly as the texture information extracted from the high-resolution layers. Due to this parallelism, MRNet performs better than MU-Net for those cases where the contextual information is more critical in generating a costmap, *e.g.*,

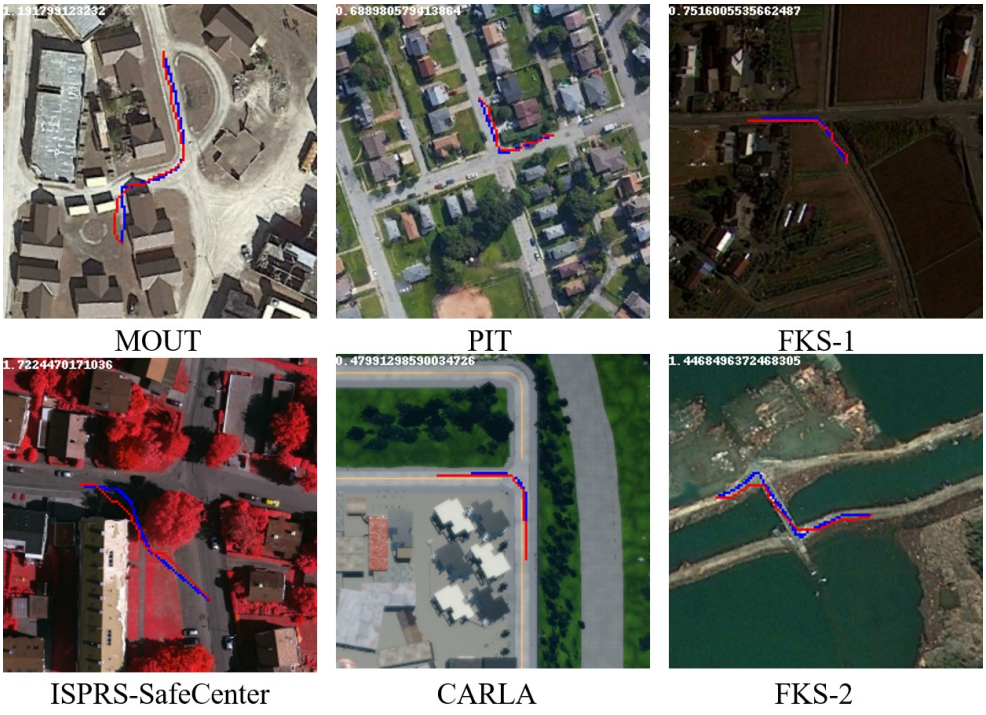


Figure 6.3: Results of MRNet on various datasets. Expert demonstration is marked in Blue and trajectory generated from IRL's recovered costmap is marked in Red, with MHD value marked on top left corner of each images.

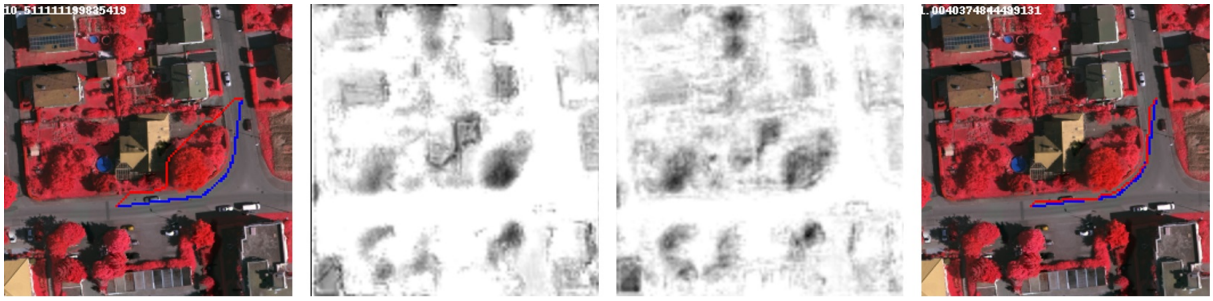
Figure 6.4 shows that MRNet discerns various types of vegetation that share similar color texture but exhibit different spatial and contextual features.

6.5 Results of Conditional Training

To evaluate the conditional learning performance of the proposed approach, the models were trained using the ISPRS series datasets in three experimental settings: 1) Independently: where each navigation behaviors were trained and their weights stored separately as a benchmark, 2) Combined: where all behaviors were trained at the same time on single set of weights. For combined training, each driving behavior was assigned with descriptors to show preference for

Table 6.2: MHD Results for Conditional Training. Lower is Better

Dataset	Independent	Combined	One-Out
SafeCenter	1.783	1.968	2.402
SafeEdge	2.960	2.950	2.980
FastCenter	2.694	2.951	3.934
FastEdge	4.616	4.104	4.878
Average	2.993	3.013	3.5486



MU-Net - Results | MU-Net - Costmap | MRNet - Costmap | MRNet - Results

Figure 6.4: Results of MRNet and MU-Net on ISPRS - FastCenter. Expert demonstration is marked in Blue and trajectory generated from IRL's recovered costmap is marked in Red, with MHD value marked on top left corner of each images. Dark regions in costmaps refer to high cost regions.

safe / fast driving and center / edge of traversable space, 3) One-Out: where CMRNet was trained for a combined dataset excluding the SafeEdge dataset and later tested with the excluded dataset to verify if the network can predict an unseen navigation behavior based on descriptions on behaviors included in the training set.

As shown in Table 6.2 CMRNet's architecture of using Fully Connected Network to learn weights assigned to each deep features extracted during training allows the network to train multiple navigation behaviors with around on average of 0.663 % on MHD compared to training each behaviors separately. Fig. 6.5 shows CMRNet generating appropriate costmaps for each navigation behaviors, such as showing gradual transition of high cost buildings to low-cost center of roads for navigation behaviors which favors center of the roads, and assigning cost slightly higher than those of roads to grass fields for where navigation behaviors allows crossing over such areas to cut travel such areas for shortest trajectory.

The last column of Table 6.2 shows the preliminary results for interpolating behaviors in One-Out scenario where a unseen navigation behavior of FastEdge was recovered with 5.675% loss in MHD compared to case in which the behavior was trained separately. While One-Out scenario shows slight performance drop for seen and unseen navigation behaviors, this is likely from bigger combined training making use of common navigation features across the behaviors. For example, all ISPRS series dataset provides examples of expert avoiding buildings, allowing CMRNet to exploit the larger size of training dataset for learning such common behaviors.

6.6 Results of Multimodal Training vs. Singlemodal Training

The results shown in Table 6.3 demonstrate that the proposed multimodal approach outperforms all of the single modal approaches including the state-of-the-art LIDAR-based model [53]. The results are not surprising because more context can be captured through multimodal sources than any single modality, subsequently reducing the level of ambiguity in understanding environments. For example, short bushes in 2D images may appear very similar to tall tree branches in terms of texture, shape, and color. Concrete buildings may look similar to concrete pavements, both of which share the same texture of grey concrete and similar shapes such as rectangles. Such ambiguity can be resolved by incorporating height information in 3D data. Likewise, while

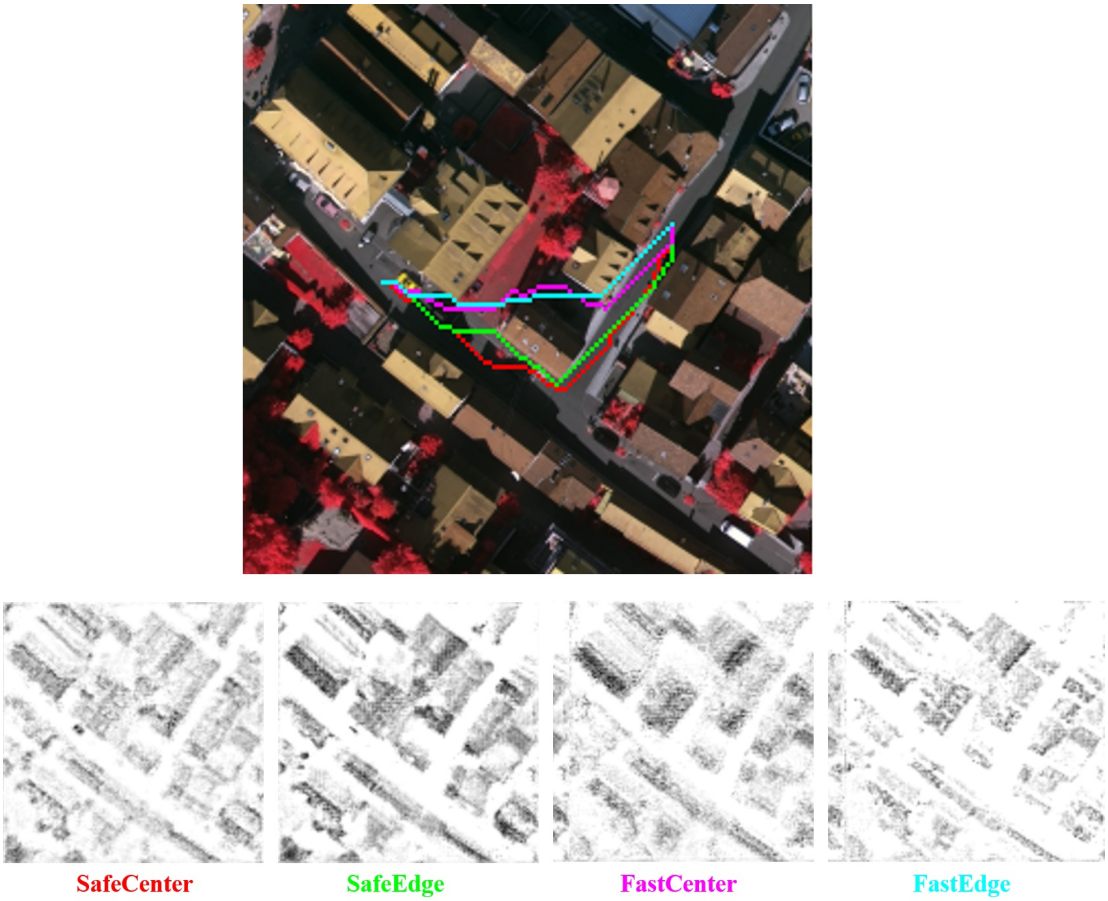


Figure 6.5: Results from Combined Training on ISPRS Dataset. Top row shows recovered trajectory for each navigation behavior colored in red (SafeCenter), green (SafeEdge), pink (FastCenter) and cyan (FastEdge). Images in the lower rows shows recovered costmaps for the navigation behaviors, with higher cost regions marked in darker colors.

the 3D data generally lack semantic information to distinguish certain objects of a similar shape. For instance, the center or side of the road both of which appear flat in 3D; using color or texture information available in images such ambiguity can be resolved.

Among the multimodal models tested, the M2RNet outperformed all other approaches in the experiments. The M2RNet maximally utilizes both input modalities with multiresolution parallel convolution operations. In this way, various types of information can be extracted from both 2D and 3D data, including texture information coming from the difference between a cell and its immediate surroundings, geometric information of how the cells of similar texture can be grouped together, and spatial information on how such groups are located in relationship to each other.

Figure 6.7 is an example of qualitative evaluation showing the performance differences in a large-scale area. For the case of MRNet (marked in cyan), only an image input is used to learn the hidden cost function. As shown in the intersection to the right, such a method generated an unfeasible trajectory crossing over a building as the network had a difficulty discerning and assigning correct costs between roads and building roofs that are both flat, grayish objects that

Table 6.3: Average MHD of Various IRL Methods. Lower is Better (Bold: best, Underline: second)

Model	MHD (CARLA)	MHD (ISPRS)	Resolution (3D)
MRNet (Image)	1.281	3.822	N/A
MRNet (3D)	5.654	N/A	0.8m
MRNet (3D)	N/A	11.005	0.2m
MEDIRL [53]	2.690	11.005	0.2m
MAXENT [56]	2.382	N/A	N/A
MIPNet	0.906	<u>2.523</u>	Same as Image
M3PNet	**0.891**	3.624	Same as Image
Multimodal Early Fusion	5.654	3.994	Same as Image
M2RNet	<u>0.894</u>	**2.229**	Same as Image

may be hard to distinguish from images alone. In the case of the MIP Net (marked in Green), the algorithm performed much better by using the 3D information to avoid buildings. However, it failed to learn to avoid small hedges where the height differences are not as distinct as buildings from surroundings and may be considered as simple height variances in flat drivable surfaces. In the case of M2R Network, multiresolution operations were used for both image and 3D data, allowing the network to identify hedges not just as local height variances, but as a continuous line of peaks within the dataset. This allowed the network to properly identify all of the obstacles including buildings, trees, and hedges.

In addition, we note that the trajectories shown in Figure 6.7 are many times longer than each demonstration segment that the networks have seen during training. This result is promising in two aspects. First, it shows that the model trained using demonstrations given in small-area patches can perform generally for long-range global planning. Second, it supports the idea that the proposed networks can learn efficiently, keeping the size of training data and computational load at a reasonable level.

6.7 Results of Various Multimodal Fusion Techniques

The lower half of Table 6.3 shows the impacts of different fusion methods on the performance. First, we observe that Early Fusion performs poorly when compared to Intermediate Fusion used in other models. In our datasets, each modality is represented in a unique way, *e.g.*, an image is provided as a 3-channel input of color spectrum while 3D data is given as the mean height, standard deviation between points, and the presence of point clouds in discrete sections of an environment. Such discrepancies in data representations explain the drastic performance gap between the early and intermediate fusion methods. The intermediate fusion method allows each modality to be processed by its own deep network structure before being merged with the features extracted from another modality.

Between the two modality-prioritized models, MIPNet and M3PNet, there was no statistically significant differences in performance.

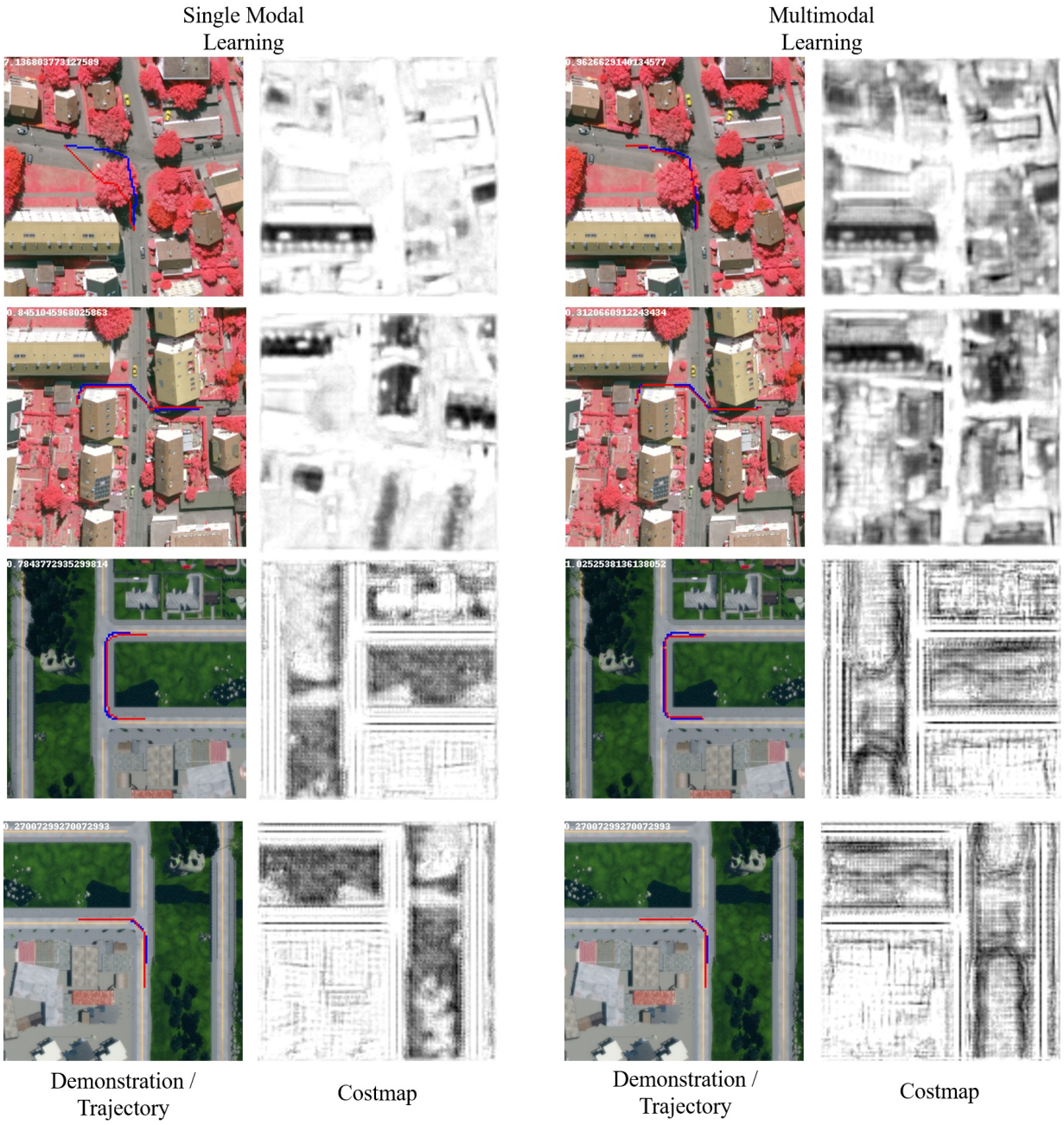


Figure 6.6: Results highlighting the difference between Single Modal(left) and Multimodal(Right) Learning in navigation. For cost maps, high cost regions are marked in black. For Demonstrations / Trajectories, expert demonstration is marked in Blue with algorithm's generated trajectory from recovered cost function is in read.

6.8 An observation on 3D data resolutions

Table 6.4 shows that, when gradually increasing the resolution of 3D data, the network performance improves at first but then deteriorates. The reason for this observation may be due to the

Figure 6.7: Example of Various Multimodal Training Demonstrated over a Longrange Task

Table 6.4: MHD performance of MIP on varying resolutions (Bold: best, Underline: second)

Resolution (m)	MHD (CARLA)	MHD (ISPRS)
0.2	0.958	7.739
0.4	0.946	<u>2.523</u>
0.8	**0.906**	3.601
1.6	<u>0.911</u>	**2.207**
3.2	0.918	3.425
6.4	0.953	4.662

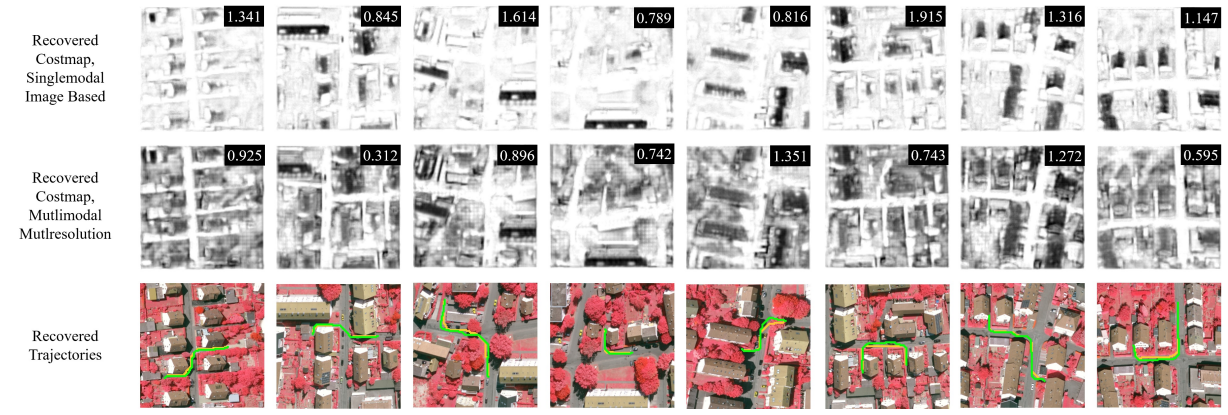


Figure 6.8: Costmaps and trajectories generated from our proposed network. Green marks the trajectories from M2RNet, yellow marks trajectories from MRNet Network, and high cost regions are marked in darker colors and trajectory is marked in cyan. Numbers in black boxes on top right corner of the costmaps is MHD of the costmap, where lower is better. From the figure, we can see that Multimodal approach is better at distinguishing costs between objects that share similar color and texture. One such case is distinguishing vegetation with bushes and trees in the gardens which should have high cost and tree branches with driveable surface hidden underneath which should have low cost. Likewise, while roads, walls of the buildings and roofs look similar as flat gray surfaces, former has much lower cost than the later. By using multimodal data, our proposed network outperforms single modal approaches as shown in the figure.

Table 6.5: MHD Results for Effect of Data Augmentation for SafeCenter Dataset

Dataset	Training Set Size	MU-Net	MRNet
Without Augmentation	118	3.705	3.336
Sample of With Augmentation	118	2.571	2.759
With Augmentation	960	2.512	1.9681

fact that the 3D data is only partially available. In the case of simulation data, CARLA, we have not observed statistically significant performance changes over various resolutions. In the case of real data, ISPRS, this trend is more pronounced, encouraging a further investigation on the effect of data resolutions.

6.9 Effect of Data Augmentation Process

Table 6.5 shows the impact of data augmentation via 1) increased size of training data and 2) reducing the risk of dataset biases such as sun casting a shadow towards a certain direction etc. Augmentation operation of rotating and flipping images help reducing such biases, improving overall performances.

*Table 6.6: Iteration Speed of MDP solvers during training**

Results, Algorithm	MEDIRL[53]	Proposed
Time Per Task(s)	660.0	0.1967

*Average result of generating 50 trajectories in 256x256 gridworld using Intel(R) Xeon(R) CPU X5690 @ 3.47GHz

6.10 Efficiency of the Forward Solver

Table 6.6 shows that our proposed Forward Solver can find solution roughly 3,000 times faster than the value iteration solver used in [53] for every iteration. This is because unlike the former, the later needs to calculate and update values for all cells until convergence (average of 95.5 updates per map of size 256 x 256), uses computational expensive exponential operations to calculate probability distribution of policies for all states, our proposed method only explores, calculates, and updates states just enough to find a trajectory that, on average, consists of 178.4 states in a 256 x 256 grid.

Chapter 7

Conclusion

In this paper, we address the research question of generating customizable traversability costmaps to accommodate various requirements and preferences given overhead multimodal observations.

We propose a Conditional Multimodal Multiresolution Deep IRL algorithm that can utilize information from image and 3D data to recover a hidden cost function for autonomous ground navigation as well as learning a spectrum of navigational behaviors to responds to various mission requirements of unstructured environments such as disaster response and / or military operations.

Main contributions are the following;

1) Efficient deep learning pipeline for using high-resolution aerial and satellite imagery for navigation in a normally computationally expensive IRL framework;

2) Two new networks for visual learning from high-resolution aerial-view imagery and share the lessons learned from our experiments on how to design a network architecture for effective cost map learning for navigation;

3) Conditional inverse reinforcement learning that can be trained for multiple navigation behaviors, with a potential for generalization such that a cost map can be predicted for a new behavior as long as it is related to known behaviors;

4) Research into effects of network structures and fusion techniques on the performance of our multimodal framework.

5) A dataset consisting of 9 different environmental settings including pre- and post-disaster scenarios and multiple navigation behaviors, along with;

6) A paired dataset including multimodal input and expert demonstrations that can be used to train IRL algorithms for autonomous navigation behaviors.

7) (Planned) Open Sourcing of codes and generated datasets used in this research.

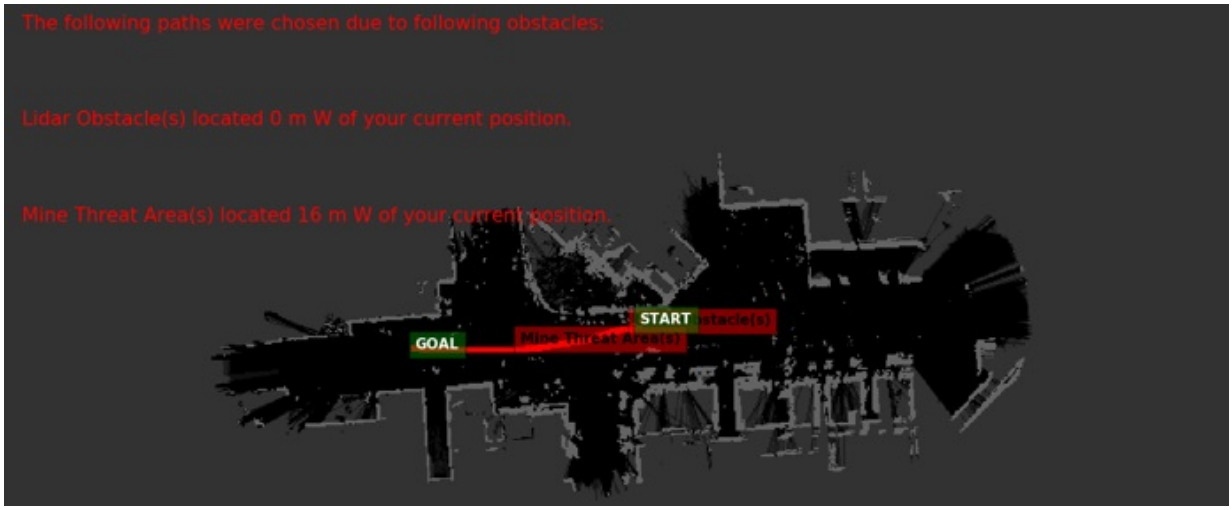


Figure 8.1: Results of Route Explanation module generating explanation for the given trajectory(marked in thick red line) with background showing LIDAR observation data(black for open space, light green for obstacles, and dark green for unobserved area). As shown in the red text over the image, the Route Explanation Module was able to identify the causes for the trajectory taking detour and explained that the the given route was generated due to LIDAR Obstacle detected in close proximity of the starting point, and a mine threat area located 16m west of the starting position.

Chapter 8

Applications and Future Works

The findings of this MSR thesis research are now being applied to various problem domains within the project group as well as being applied as a summer 2019 research project.

8.1 Explainable AI

One application of this thesis's research is proving route explanation algorithm for the human-robot teaming in military and disaster relief operations. Subscribing the ARL's Inverse Optimal Control (IOC) via ROS interface, the proposed algorithm can generate image and text explanation of the route choices.

Figure 8.2: Example of proposed CARLA Global Planner providing waypoints (marked in yellow) between start (marked in green) and goal (marked in red) for BIG imitation learning algorithm. As shown in the figure, instead of making a U-turn at the intersection, which is physically not possible due to vehicle's nonholonomic constraints, or driving backwards, which is not possible due to traffic regulations, the waypoints guide the vehicle towards target in a sequence of left turns, which is the shortest feasible pathway for the vehicle.

The work on Route Explanation will continue during the summer, with the goal of publishing a paper on the subject by next quarter.

8.2 CARLA Global Planner

A heavily modified version of the A* Planner used in this work was applied the self-driving vehicle project using the CARLA simulator as a global planner guiding imitation learning. Given a start point and goal point, the proposed algorithm can generate a sequence of waypoints that does not violate traffic regulations, such as driving in the wrong side of the road.

8.3 Transfer Learning and Husky Controller

To apply the findings in IRL to real life, works are completed / underway on Husky Controller and Transfer Learning.

Husky Controller is a control algorithm in charge of transferring waypoints generated by IRL algorithm into motor control inputs on the Husky Robot platform via ROS interface. The Husky Controller was successfully used in a demonstration at CMU to navigate around obstacles using IRL algorithm. The controller algorithm also provided low-level interface between Husky and the Socially Compliant navigation algorithm for field tests and debugging prior to integration with ARL's low-level controllers.

Transfer Learning in this section refers to modifiying IRL training scheme such that the network can be trained with one dataset and tested with a totally different dataset. By doing so, we hope to eliminate the need to retrain the network every time a new mission arises, allowing our system to respond much faster to natural disaster and military operations.

Bibliography

[1] Airbus. Airbus to provide near real-time access to it's satellite data. https://www.airbus.com, January 2018.

[2] Joel Akeret, Chihway Chang, Aurelien Lucchi, and Alexandre Refregier. Radio frequency interference mitigation using deep convolutional neural networks. *Astronomy and computing*, 18:35–39, 2017.

[3] Anonymous. Conditional deep inverse reinforcement learning for multiple navigation behaviors. In *Under Review*.

[4] Nicolas Audebert, Bertrand Le Saux, and Sébastien Lefèvre. Semantic segmentation of earth observation data using multimodal and multi-scale deep networks. In *Asian Conference on Computer Vision*, pages 180–196. Springer, 2016.

[5] Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1 edition, 1957.

[6] Dimitri P Bertsekas and John N Tsitsiklis. Neuro-dynamic programming: an overview. In *Decision and Control, 1995., Proceedings of the 34th IEEE Conference on*, volume 1, pages 560–564. IEEE, 1995.

[7] Paulo Blikstein and Marcelo Worsley. Multimodal learning analytics and education data mining: using computational technologies to measure complex learning tasks. *Journal of Learning Analytics*, 3(2):220–238, 2016.

[8] A. Boularias, F. Duvallet, J. Oh, and A. Stentz. Learning to ground spatial relations for outdoor robot navigation. In *Proc. of IEEE Conference on Robotics and Automation (ICRA)*, 2015.

[9] Abdeslam Boularias, Jens Kober, and Jan Peters. Relative entropy inverse reinforcement learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 182–189, 2011.

[10] Jaedeug Choi and Kee-Eung Kim. Nonparametric bayesian inverse reinforcement learning for multiple reward functions. In *Advances in Neural Information Processing Systems*, pages 305–313, 2012.

[11] F. Codevilla, M. Miiller, A. López, V. Koltun, and A. Dosovitskiy. End-to-end driving via conditional imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9, May 2018. doi: 10.1109/ICRA.2018.8460487.

[12] Copernicus. Copernicus emergency management service - mapping. https://

emergency.copernicus.eu, January 2018.

[13] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[14] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[15] M-P Dubuisson and Anil K Jain. A modified hausdorff distance for object matching. In *Proceedings of 12th international conference on pattern recognition*, pages 566–568. IEEE, 1994.

[16] Google Earth. Imagery of mout training facility, .

[17] Google Earth. Imagery of pittsburgh, pa, usa, .

[18] Andreas Eitel, Jost Tobias Springenberg, Luciano Spinello, Martin Riedmiller, and Wolfram Burgard. Multimodal deep learning for robust rgb-d object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 681–687. IEEE, 2015.

[19] Dave Ferguson and Anthony Stentz. Focussed processing of mdps for path planning. In *Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on*, pages 310–317. IEEE, 2004.

[20] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International Conference on Machine Learning*, pages 49–58, 2016.

[21] Digital Globe. Imagery of fukushima, japan before 2011 tohoku earthquake, .

[22] Digital Globe. Imagery of fukushima, japan after 2011 tohoku earthquake, .

[23] Digital Globe. Open data program. https://www.digitalglobe.com, January 2018.

[24] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[25] ISPRS. International society of programmetry and remote sensing. http://www.isprs.org.

[26] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[27] Kris M Kitani, Brian D Ziebart, James Andrew Bagnell, and Martial Hebert. Activity forecasting. In *European Conference on Computer Vision*, pages 201–214. Springer, 2012.

[28] Sergey Levine, Zoran Popovic, and Vladlen Koltun. Nonlinear inverse reinforcement learning with gaussian processes. In *Advances in Neural Information Processing Systems*, pages 19–27, 2011.

[29] Kun Li and Joel W Burdick. Inverse reinforcement learning in large state spaces via function approximation. *arXiv preprint arXiv:1707.09394*, 2017.

[30] Guan-Horng Liu, Avinash Siravuru, Sai Prabhakar, Manuela Veloso, and George Kantor. Learning end-to-end multimodal sensor policies for autonomous navigation. *arXiv preprint arXiv:1705.10422*, 2017.

[31] Dimitrios Marmanis, Jan D Wegner, Silvano Galliani, Konrad Schindler, Mihai Datcu, and Uwe Stilla. Semantic segmentation of aerial images with an ensemble of cnns. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 3:473, 2016.

[32] H Brendan McMahan and Geoffrey J Gordon. Fast exact planning in markov decision processes. In *ICAPS*, pages 151–160, 2005.

[33] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, et al. Junior: The stanford entry in the urban challenge. *Journal of field Robotics*, 25(9):569–597, 2008.

[34] Youssef Mroueh, Etienne Marcheret, and Vaibhava Goel. Deep multimodal learning for audio-visual speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 2130–2134. IEEE, 2015.

[35] United Nations. Data application of the month: Free satellite data. `http://www.un-spider.org`, January 2018.

[36] Andrew Y Ng and Stuart J Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 663–670. Morgan Kaufmann Publishers Inc., 2000.

[37] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 689–696, 2011.

[38] Xuran Pan, Lianru Gao, Andrea Marinoni, Bing Zhang, Fan Yang, and Paolo Gamba. Semantic labeling of high resolution aerial imagery and lidar data with fine segmentation network. *Remote Sensing*, 10(5):743, 2018.

[39] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[40] Dhanesh Ramachandram and Graham W Taylor. Deep multimodal learning: A survey on recent advances and trends. *IEEE Signal Processing Magazine*, 34(6):96–108, 2017.

[41] Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. Maximum margin planning. In *Proceedings of the 23rd international conference on Machine learning*, pages 729–736. ACM, 2006.

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

[43] Pierre Sermanet, Raia Hadsell, Marco Scoffier, Matt Grimes, Jan Ben, Ayse Erkan, Chris Crudele, Urs Miller, and Yann LeCun. A multirange architecture for collision-free off-road robot navigation. *Journal of Field Robotics*, 26(1):52–87, 2009.

[44] Fei Shao, Songmei Cai, and Junzhong Gu. A modified hausdorff distance based algo-

rithm for 2-dimensional spatial trajectory matching. In *Computer Science and Education (ICCSE), 2010 5th International Conference on*, pages 166–172. IEEE, 2010.

[45] Jun Shi, Xiao Zheng, Yan Li, Qi Zhang, and Shihui Ying. Multimodal neuroimaging feature learning with multimodal stacked deep polynomial networks for diagnosis of alzheimer's disease. *IEEE journal of biomedical and health informatics*, 22(1):173–183, 2018.

[46] David Silver, J. Andrew Bagnell, and Anthony Stentz. Learning from demonstration for autonomous navigation in complex unstructured terrain. *The International Journal of Robotics Research*, 29(12):1565–1592, 2010.

[47] Anthony Stentz. Optimal and efficient path planning for partially-known environments. In *ICRA*, volume 94, pages 3310–3317, 1994.

[48] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, et al. Stanley: The robot that won the darpa grand challenge. *Journal of field Robotics*, 23(9):661–692, 2006.

[49] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, MN Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.

[50] Olivia Ward. How a canadian team's drones are changing search and rescue. `https://www.thestar.com`, May 2016.

[51] Maggie Wigness, John G Rogers III, and Luis E Navarro-Serment. Robot navigation from human demonstration: Learning control behaviors. *IEEE ICRA 2018*.

[52] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. Deep inverse reinforcement learning. *CoRR, abs/1507.04888*, 2015.

[53] Markus Wulfmeier, Dominic Zeng Wang, and Ingmar Posner. Watch this: Scalable cost-function learning for path planning in urban environments. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 2089–2095. IEEE, 2016.

[54] Wei Zhang, Youmei Zhang, Lin Ma, Jingwei Guan, and Shijie Gong. Multimodal learning for facial expression recognition. *Pattern Recognition*, 48(10):3191–3202, 2015.

[55] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[56] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

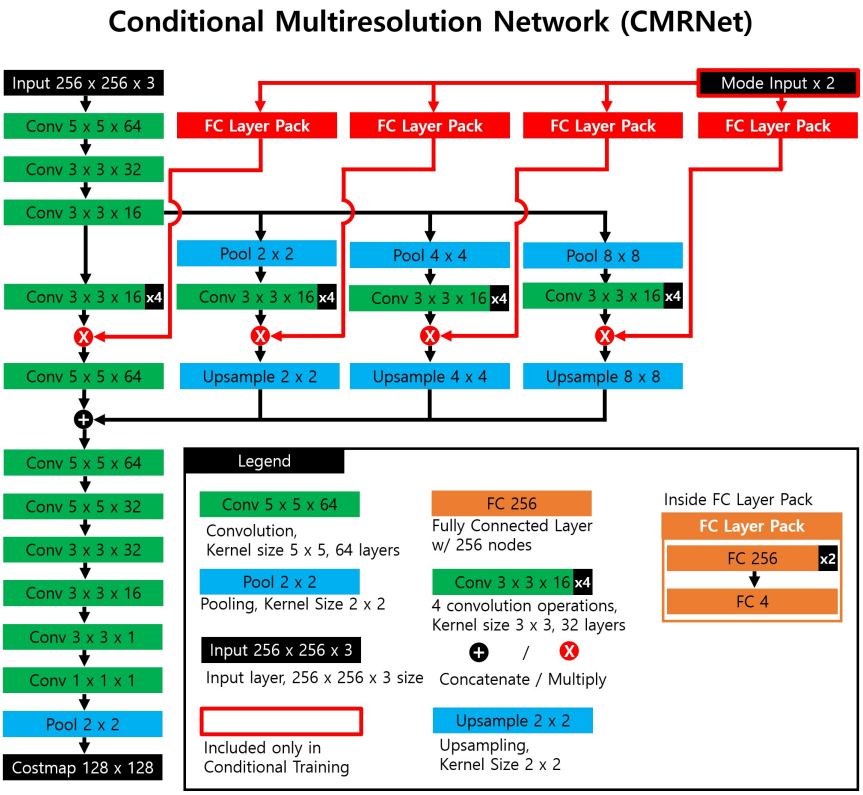


Figure 8.3: Diagram showing detailed structure of the CMRNet used in this paper. Note that all activation functions are Rectified Linear Units(ReLUs)

Appendix

8.4 Detailed Network Structure and Training Parameters

This section of appendix covers details of the network structure used in this report, as well as meta-parameters used for training

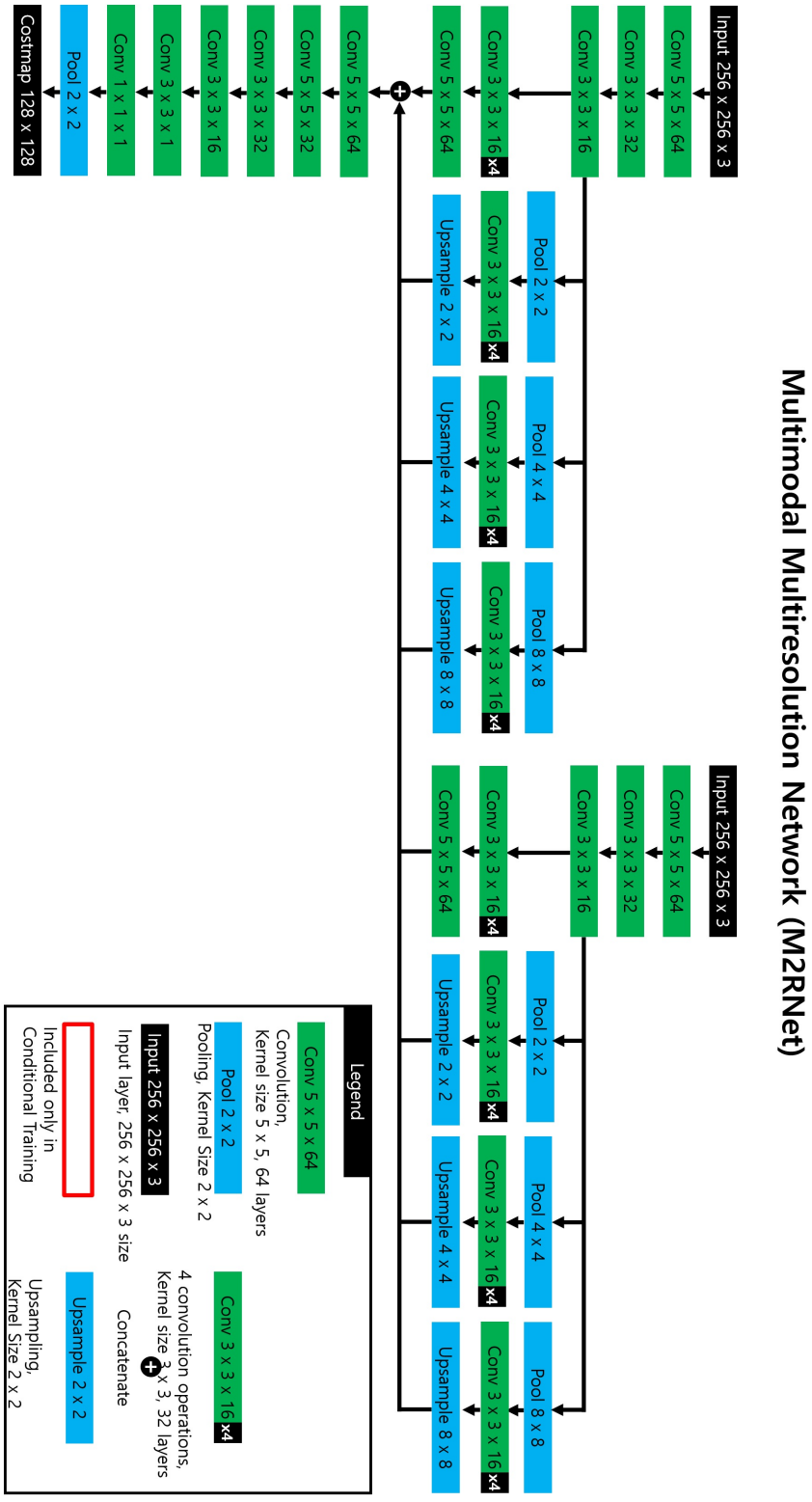


Figure 8.4: Diagram showing detailed structure of the M2RNet used in this paper. Note that all activation functions are Rectified Linear Units(ReLUs)