Master's Thesis

Autonomous Navigation for Mobile Robots: Machine Learning-based Techniques for Obstacle Avoidance

Bumsoo Park

Department of System Design and Control Engineering

Graduate School of UNIST

2019

Autonomous Navigation for Mobile Robots: Machine Learning-based Techniques for Obstacle Avoidance

Bumsoo Park

Department of System Design and Control Engineering

Graduate School of UNIST

ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

Autonomous Navigation for Mobile Robots: Machine Learning-based Techniques for Obstacle Avoidance

A thesis/dissertation
submitted to the Graduate School of UNIST
in partial fulfillment of the
requirements for the degree of
Master of Science

Bumsoo Park

12/13/2018

Approved by

Advisor
Hyondong Oh

Autonomous Navigation for Mobile Robots: Machine Learning-based Techniques for Obstacle Avoidance

Bumsoo Park

This certifies that the thesis/dissertation of Bumsoo Park is approved.

12/13/2018

signature

Advisor: Hyondong Oh

signature

Hungsun Son

signature

Sanghoon Kang

Abstract

Autonomous navigation of unmanned aerial vehicles (UAVs) has posed several challenges due to the limitations regarding the number and size of sensors that can be attached to the mobile robots. Although sensors such as LIDARs that directly obtain distance information of the surrounding environment have proven to be effective for obstacle avoidance, the weight and cost of the sensor contribute to the restrictions on usage for UAVs as recent trends require smaller sizes of UAVs. One practical option is the utilization of monocular vision sensors which tend to be lightweight and have a relatively low cost, yet still the main drawback is that it is difficult to draw a certain rule from the sensor data. Conventional methods regarding visual navigation makes use of features within the image data or estimate the depth of the image using various techniques such as optical flow. These features and methodologies however still rely on human-based rules and features, meaning that robustness can become an issue.

A more recent approach to vision-based obstacle avoidance exploits heuristic methods based on artificial intelligence such as deep learning technologies, which have shown state-of-the-art performance in fields such as image processing or voice recognition. These technologies are capable of automatically selecting important features for classification or prediction tasks, hence allowing superior performance. Such heuristic methods have proven to be more efficient as the rules and features that are drawn from the image are automatically determined, unlike conventional methods where the rules and features are explicitly determined by humans.

In this thesis, we propose an imitation learning framework based on deep learning technologies that can be applied to the obstacle avoidance of UAVs, where the neural networks in this framework are trained upon the flight data obtained from human experts, extracting the necessary features and rules to carry out designated tasks. The system introduced in this thesis mainly consists of three parts: the data acquisition and preprocessing phase, the model training phase, and the model application phase. A CNN (Convolutional Neural Network), 3D-CNN, and a DNN (Deep Neural Network) will each be applied to the framework and tested with respect to the collision ratios to validate the obstacle avoidance performance.

Contents

List of Figures

List of Tables

INTRODUCTION

Motivation

With the recent demand for mobile robots with higher levels of autonomy, several challenges have come to presence, especially when it comes autonomous navigation of an agent within a certain environment. Although there are numerous requirements for an agent to be able to autonomously navigate, obstacle avoidance is considered one of the most fundamental factors, as the agent should be capable of avoiding obstacles during its navigation. During the navigation phase, autonomous navigation is usually obtained by applying path planning methods, where a certain path is assigned to an agent given that it has information on the surrounding environment. For a typical case, geometrical information such as the obstacles between the start and goal point are considered in the path planning stage, enabling obstacle avoidance capabilities for the autonomous agent. However, under certain circumstances where the environment is uncertain, or where the path cannot be generated due to issues such as the inability to obtain a map of the surrounding environment, real-time path planning may not be available. In such cases, local path planning methods take place instead of global path planning methods [1] where obstacle avoidance is considered as the main criterion.

Meanwhile, due to the sensor limitations for small sized UAVs (Unmanned Aerial Vehicles), sensors that directly measure the distance may not always be a viable choice. As a result, monocular camera sensors have been a widely accepted choice of sensor due to its efficiency with respect to cost and computational power, weight, and size. These monocular camera sensors however, still pose some major disadvantages as the image data obtained from monocular cameras are not very intuitive. Considering the fact that there are so many possible number of outcomes and that the image data does not directly provide useful information such as the distance, it is extremely difficult to determine the features necessary for obstacle avoidance or derive certain rules based on the sensor data. Therefore, we focus on developing an effective measure to heuristically extract the necessary features and determine a rule for obstacle avoidance in this thesis.

Research Objectives

The objective of this thesis is to propose a human data-based imitation learning framework that can be applied to the obstacle avoidance of UAVs, where the appropriate features and rules are heuristically determined upon the given data. We propose a 3D-CNN based feature extractor and classifier for the imitation learning framework. The main goals of this study is as shown below.

The first goal of this work is to develop a human data acquisition system for the end-to-end imitation learning framework. A communication node that obtains data from the simulation environment will be used to both obtain data from the environment, and send control inputs later on as the neural network

is successfully trained. Through implementation of this communication node, not only can we test and evaluate the proposed system, but also expect future usages of the system in actual systems.

The second goal is to develop a deep learning model that is capable of extracting the necessary spatiotemporal features and derive a certain rule upon the training dataset. Unlike the majority of conventional researches regarding obstacle avoidance, the proposed neural network in this study will heuristically extract the features and obtain a certain rule for obstacle avoidance.

Under the main assumption that there underlies a certain relationship or function between visual inputs and actions of a human when it comes to obstacle avoidance, we focused on teaching an agent so that it can map the image inputs to the heading directions for obstacle avoidance. Through sufficient acquisition of camera data and the corresponding control inputs generated from human experts, our neural network is likely to learn the underlying rule for obstacle avoidance. In this paper, we present a framework for obstacle avoidance with deep learning-based imitation learning methods. 3D-CNN models [Learning spatiotemporal features with 3d convolutional networks] will be utilized for the imitation learning framework, as obstacle avoidance with a monocular camera is considered a sequential task.

Outline of the Thesis

There are a total of five chapters in this thesis. The motivation and the objectives of this research are introduced in the first chapter, and a literature survey on researches regarding conventional and recent obstacle avoidance methods is conducted. In chapter 3, background information on deep learning technologies is given, where the CNN (Convolutional Neural Network) and 3D-CNN (3-Dimensional Convolutional Neural Network) utilized in this research are described. The overall imitation learning framework used in this study is described in detail in chapter 4. Experimental results as well as the training results of the neural network are given, and analyzed through a comparison of different neural networks. Chapter 5 provides the conclusion of this research, followed by the potential future work.

LITERATURE REVIEW

Conventional Obstacle Avoidance Methods

Obstacle Avoidance using Distance Measuring Sensors

The most intuitive method for obstacle avoidance is to use sensors that directly measure the distance of the obstacles within the environment. Although LIDARs, depth sensors, and ultrasonic sensors all provide direct information on the distance information of the surrounding environment, we will be focusing on researches that utilize ultrasonic sensors due to the sensor restrictions of small-sized UAVs. Borenstein and Koren [2] introduce a nursing robot that uses two ultrasonic range finders to detect obstacles and map the surrounding environment. This research mainly focuses on developing methods to identify obstacles from the sensor data and enhancing the results in uncertain situations. Borenstein and Koren proposes another method [3] using vector field histograms that enables real-time obstacle avoidance. This study mainly uses a two-dimensional Cartesian histogram as a world model, where the grid is updated using on-board range sensors. Both researches do provide effective measures for obstacle avoidance of mobile robots, yet the main limitation of ultrasonic sensors is that they have a time delay due to their operation mechanism, and that the information provided by the sensor contains less contextual information compared to sensor data from LIDARs or vision sensors.

Rule-based Obstacle Avoidance with Monocular Vision Sensors

Obstacle avoidance methods using monocular vision sensors usually rely on feature extraction methods, where certain features are extracted from the image to identify and locate potential obstacles, or depth estimation methods, where techniques such as optical flow are used to estimate the depth based on the given monocular image.

Souhila and Karim [4] introduce an obstacle avoidance system using optical flow to estimate the depth of the image. By using optical flow and FOE (Focus of Expansion) calculation, the agent utilizes a balance strategy to avoid the obstacles based on the monocular image. Song and Huang [5] introduces methods for fast optical flow estimation and applying it to the obstacle avoidance of a mobile robot. Although obstacle avoidance based on depth estimation techniques provide effective methodologies, the algorithm may lose robustness in some situations as the decision making logic or rule for obstacle avoidance relies on human made rules.

Heuristic Obstacle Avoidance Methods

Reinforcement Learning

Among the various heuristic approaches, reinforcement learning-based methods have recently been receiving attention in the field of robotics and control, due to its robustness and heuristic characteristics when it comes to the derivation of rules. Reinforcement learning is a type of machine learning which is mainly concerned about how an agent should behave within a given environment so as to maximize a cumulative reward, developing a policy for a designated task.

Michels, Saxena, and Ng [6] introduced reinforcement learning methods for obstacle avoidance using a monocular vision sensor. In this study, a feature vector is determined from the image, where information on the distance to the nearest obstacle is included in the vector for learning purposes. Synthetic images are used and trained using reinforcement learning exhibiting reasonable depth estimates on real image data, and the control policy trained in a simulator worked well on real autonomous driving situations. Huang, Cao, and Guo also propose a reinforcement learning-based obstacle avoidance method [7] using Q-learning and neural networks. Infrared distance sensors with respect to multiple directions are used as the sensory input, determining the state of robot in the reinforcement learning framework.

Imitation Learning

Recent researches more actively utilize heuristic methods where data-driven or machine learning-based approaches that use apprenticeship methods [8], yet the algorithms that are used for the supervision of the machine learning model are still developed upon human-made rules. As a result, direct learning from human expert demonstration data may provide better results depending on the task, especially for cases where constraints on sensor usage are present. In fact, there have been multiple attempts regarding imitation learning from human experts. One of the most primitive researches was proposed by Pomerleau [9], where a single layer of a neural network was used to map the images to the steering angles, enabling the agent to stay on the road. Other work regarding how to teach an aircraft to fly using human data [10] has been introduced by Sammut, Hurst, Kedzier, and Michie. In this work, decision trees were used to train and design the autopilot for the aircraft. One representative example of imitation learning from human data was presented by Ross, Gordon, and Bagnell [11], using the DAGGER (Dataset Aggregation) algorithm. Here, a policy is determined online in an iterative manner, using the aggregated data from the algorithm. This algorithm was also used for vision-based autonomous navigation in forest trails [12]. Kim and Chen [13] presented an imitation learning framework for drones to track and follow a certain target based on neural networks.

DEEP NEURAL NETWORKS

Artificial Neural Network

Deep learning, a type of machine learning, refers to artificial neural networks (ANNs) with deep layers. The concept of ANNs was first introduced in 1943 by McCulloch, a psychiatrist and neuroanatomist, and Pitts, a mathematician. Through the utilization of ANNs, they were able to show that ANNs are capable of performing logic operations such as AND, OR, and NOT. This was obtained by the modeling of biological neurons, constructing networks with interconnecting units that have very simple functions [14]. This concept was later reviewed in 1958 by Rosenblatt [15] in 1958 as a perceptron problem. By the 1960s it was considered that all problems could be solved by ANNs, receiving a great amount of attention in various fields of research. However, Minsky and Papert discovered that ANNs were not capable of classifying XOR problems in 1969, which was considered a serious drawback of the technology. Rumelhart approached this issue by proposing a multilayer perceptron model with numerous layers in 1986 [16], and by doing so, he was able to approximate complex nonlinear functions including the previously unresolved XOR problem [17] with the existence of abundant data using the error backpropagation learning algorithm. The recent introduction of deep learning technologies into various fields, has brought about significant advances as numerous studies on basic multilayer perceptrons known as ANNs, or modified versions of ANN such as the CNN [18], which is well known for its performance in image processing, and the RNN (Recurrent Neural Network), which is mainly used in analyzing sequential data due to its recurrent structure.

Among the various types of neural networks in deep learning technologies, ANNs are the most basic type of neural network. A typical ANN is mainly consisted of an input layer, multiple hidden layers where a certain number of nodes or neurons make up a single layer, and an output layer. Each node is a perceptron that carries out a simple linear operation, and for a general ANN, each node from different layers are interconnected to each other. This is also known as a fully connected layer and is used in a combination of different types of layers for other neural networks.

Within the hidden layers, the connected nodes as a whole represent a highly abstracted function despite the fact that each node simply consists of a linear operation and an activation function, meaning that the deeper the layer of the ANN, the more complicated information the ANN can represent. An ANN layout is shown in Figure 1. x is the input vector of the ANN, where an n-dimensional vector is used. h_i refers to the nodes within the hidden layer, where f is the activation function, ω_{ij} and b_{ij} represent the weight bias of the of the jth node in the ith layer. The output layer is expressed without an activation function.

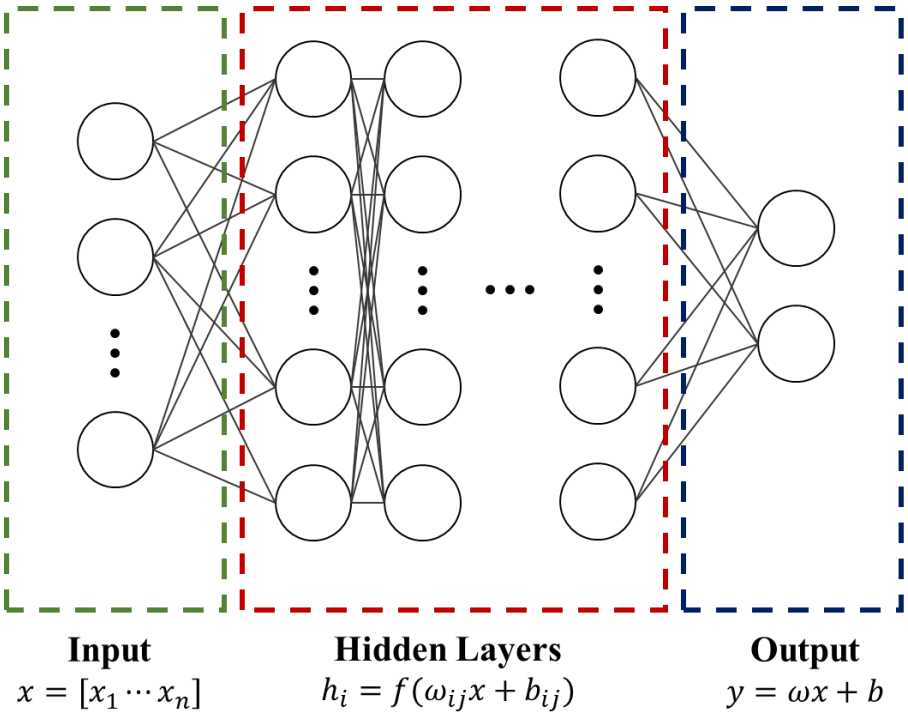


Figure 1 Artificial Neural Network Layout

Activation Function

For neural networks, a nonlinear function also known as an activation function is used to signal the neurons of the network. Sigmoid and rectified linear unit functions are the most widely used activation functions for various neural networks, and functions such as hyperbolic tangent, leaky ReLU, Maxout, or ELU functions have also been used for certain purposes.

Sigmoid Function

A sigmoid function is a function with continuous values between 0 and 1, which is similar to a unit step function in a continuous sense. The most known advantage of this type of activation function is that it is capable of expressing derivative values easily, making it easy to apply to neural networks. Nevertheless because the majority of values are expressed as values between 0 and 1, the weights of the nodes may not be effectively updated as the layers get deeper, or the learning speed may be decreased given that the gradients are such small values [19]. The sigmoid function and its plot are given below in Equation (1) and Figure 2, respectively.

$$g(z) = \frac{1}{1+e^{-z}} \tag{1}$$

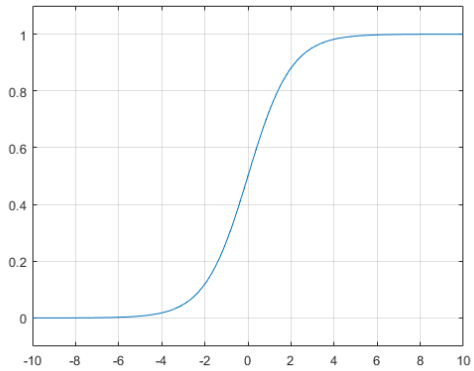


Figure 2 Sigmoid Function

Hyperbolic Tangent Function

The hyperbolic tangent function has a sigmoidal shape similar to the sigmoid function, but the output values are between the range -1 and 1. Such properties reduce the probability of the network getting stuck during the training phase as the range is increased from (0, 1) to (-1, 1). The hyperbolic tangent function and its plot are shown below in Equation (2) and Figure 3.

$$\tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{2}$$

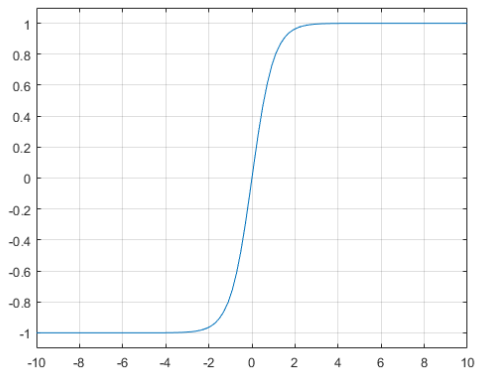


Figure 3 Hyperbolic Tangent Function

Rectified Linear Unit Function

The Rectified Linear Unit (ReLU) Function is another type of activation function used for neural networks. In response to the learning speed or weight update problem for the sigmoid function, the ReLU function has been widely adopted as it only uses positive output values. The main advantage of this function is that the gradient is either 0 or 1, overcoming issues such as the vanishing gradient problem. [20][21]. The ReLU function and its plot are shown below in Equation (3) and Figure 4.

$$f(z) = \max(0, z) \tag{3}$$

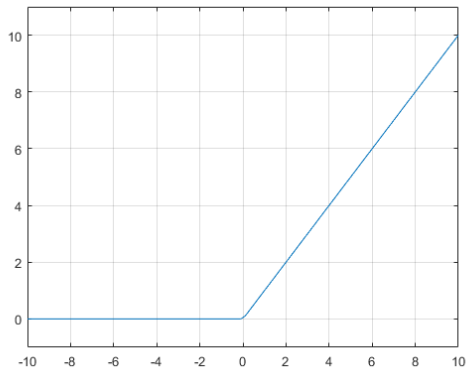


Figure 4 Rectified Linear Unit (ReLU) Function

Backpropagation

In general, the weights and biases of the neural network are optimized using algorithms such as gradient descent for the backpropagated errors during the training phase of the neural network. The training process of a neural network mainly consists of three steps: the training data is first passed through the network using forward propagation, next the error is evaluated based on the feedforwarded value, and finally the network parameters are updated based on the calculated error. These steps are repeated until the error or loss of the object function reaches a certain threshold value.

The process of the backpropagation of errors is shown below in Figure 5, and the errors in each the output and hidden layers with respect the ith node are expressed in Equations (4) and (5), respectively. $\delta_i^{(n_l)}$ is the error in the layer n_l, in the ith node and f' is the derivative of the activation function used in each layer. W_{ji} is the weight of the ith to j^{th} connection in the network.

$$\delta_i^{(n_l)} = -(y_i - a_i^{(n_l)}) \cdot f'(z_i^{(n_l)}) \tag{4}$$

$$\delta_i^{(n_l)} = (\sum_{j=1}^{s_l+1} W_{ji}^{(l)} \delta_j^{(l+1)}) \cdot f'(z_i^{(n_l)}) \tag{5}$$

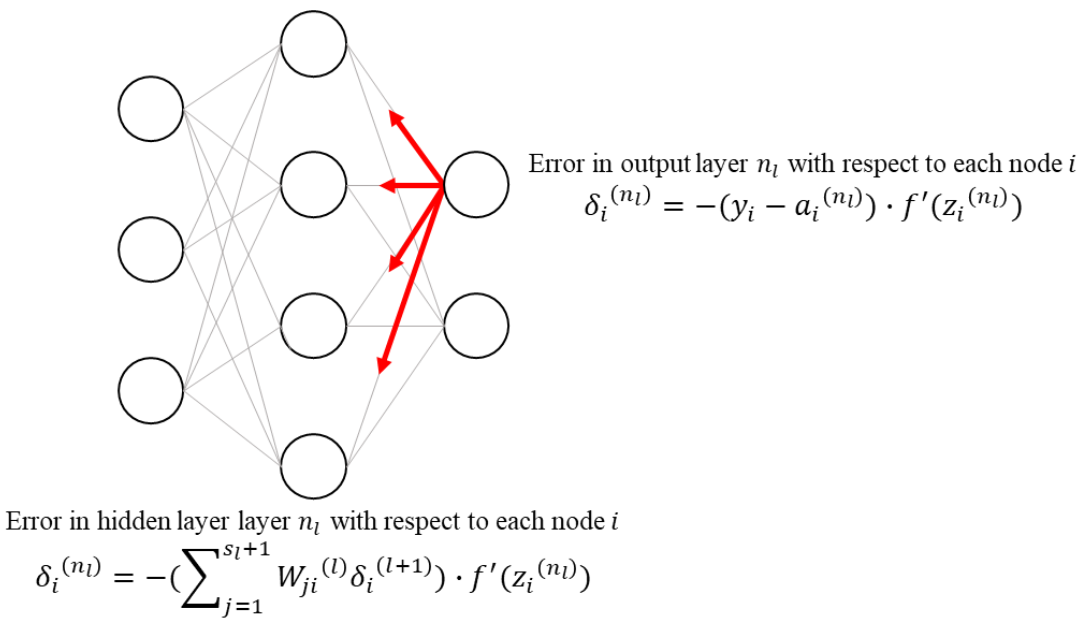


Figure 5 Backpropagation Algorithm

Convolutional Neural Network

A convolutional neural network is a type of deep neural network that uses convolution operations within the layers [18]. Due to the convolution operation, the CNN in particular has proved to be effective when it comes to analyzing images. This type of neural network was originally inspired by the way animals perceives visual inputs, since animals view images in a way similar to that of the convolution operation. Humans as well look at local parts of an image when identifying images instead of looking at the image as a whole, where local features are extracted for the analysis process. CNNs extract features within the input data based on the different types of convolution kernels or filters that are used, providing a significantly more effective measure to analyze imagery compared to basic ANNs.

Convolution Layer

The convolution layer is the most distinctive feature of a CNN. In these convolution layers, a convolution operation is conducted. By definition, the convolution operation in one-dimensional continuous domain is defined as the following Equations. The ω refers to the filter or kernel of the convolution operation, whereas x refers to the input of the function. s is the output or result of the convolution operation.

$$s(t) = \int x(a)w(a-t)da \tag{6}$$

$$s(\mathrm{t}) = (\mathrm{x} * \omega)(\mathrm{t}) \tag{7}$$

In a two-dimensional discrete domain, in which a normal image is considered, the convolution operation is illustrated in Figure 6. The convolution operation is done throughout the whole input, usually with a stride of 1.

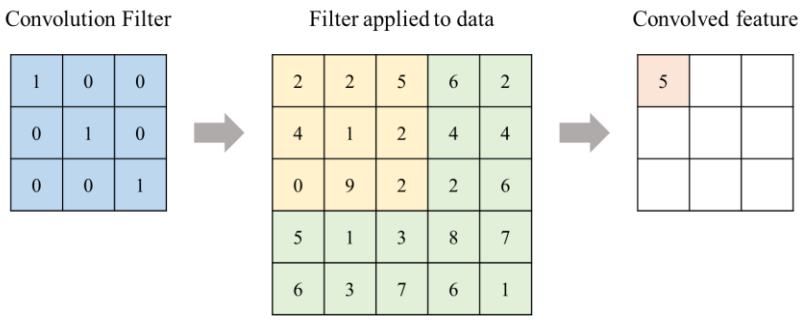


Figure 6 Convolution Operation in 2D CNN

Pooling Layer

One additional characteristic of CNNs is that the pooling operation is done after the convolution operation is conducted. Pooling refers to the process of down-sampling of the input data by reducing the dimension with respect to a certain rule. As the raw input data or image contains a considerable amount of unnecessary data, down-sampling of the data also helps prevent overfitting issues [22], and thus pooling is an essential part of CNNs.

The two most commonly used types of pooling are max pooling and average pooling. Unlike the convolution operation, the pooling kernel usually has a stride value that is same as the size of the kernel itself, so that the pooling operation does not overlap. For max pooling, the maximum value within the pooling kernel is selected, and the average value is saved for the average pooling operation.

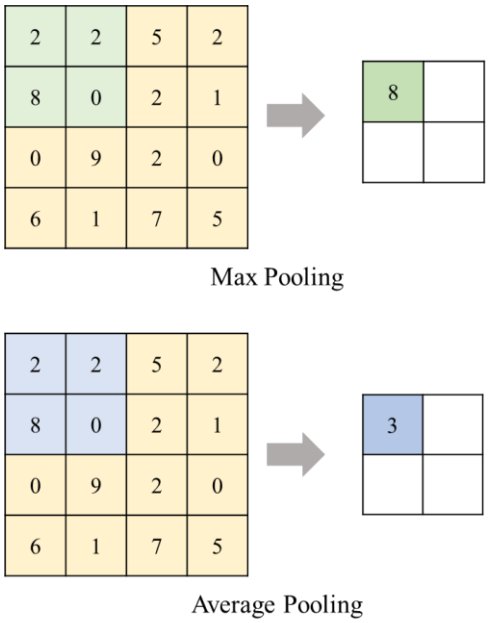


Figure 7 Different Types of Pooling Operations

Fully Connected Layer

The feature extraction and down-sampling process is finished within the convolution and pooling layers. The features that are picked up in the previous layers are input to a fully connected layer in order to carry out tasks such as classification.

3D-Convolutional Neural Network

In addition to 1D and 2D convolutional operations, 3D convolutional operations can be done with the usage of a 3D convolutional filter. This typical type of neural network is known as a 3D CNN, and is widely used for volumetric analysis due to its capabilities to extract three dimensional features. Depending on the type of analysis, the feature vector of the input data is passed on to the fully connected layer, enabling classification tasks in the fully connected layer. An example of a 3D convolution filter is shown below in Figure 8.

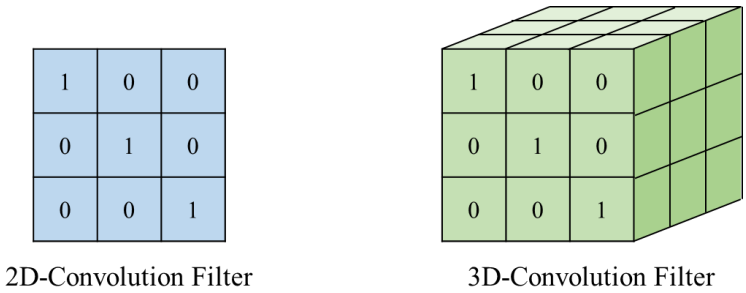


Figure 8 Comparison of 2D and 3D Convolution Filter

Convolution through Time

3D CNNs have originally been applied to volumetric analyses such as MRI imagery [23] or point cloud classification [24] [25]. However, by viewing the time axis as a third physical dimension, we are able to extract spatiotemporal features within the given data through 3D convolution. This type of approach has been introduced in several studies regarding action recognition [26] [27], and can be applied to other sequential problems such as obstacle avoidance. In further sections, we will be discussing the actual usage of the 3D CNN in the imitation learning framework of this study. Figure 9 illustrates the process of convolution through time.

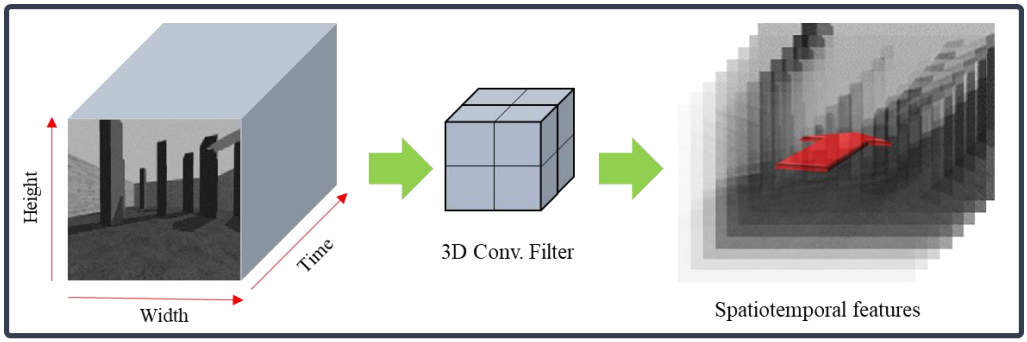


Figure 9 Convolution through Time

IMITATION LEARNING-BASED OBSTACLE AVOIDANCE FRAMEWORK

Problem Statement

For a typical obstacle avoidance problem using a monocular vision sensor, features that are able to abstract the information of the obstacles within the images must be first extracted, followed by the decision making based on such feature vectors. This process may be challenging as human-defined features and rules may not always be guaranteed its robustness. One approach is to heuristically determine the features and rules based on a given dataset of expert demonstration using supervised learning. As humans are able to autonomously navigate within a given environment even with only the usage of visual data, the main assumption in this work is that humans use a certain policy to navigate with visual inputs, i.e., there exists a certain relationship between the images and control inputs in order to achieve obstacle avoidance. In this research, we will focus on how the necessary data is provided and used to train the neural network, and validate the performance of the 3D CNN that is used in this study through comparison with two other types of neural networks.

Imitation Learning

Imitation learning, also known as apprenticeship learning, is a methodology where a certain agent learns rules from expert demonstration. The key concept is similar to the process of a child learning from a teacher or their parents. Initially, data is collected from the expert demonstration, which is used to train a certain classifier or regressor via supervised learning techniques. This classifier or regressor is used as the policy for the imitator or agent to choose certain actions within the given environment. An example of the imitation learning concept is shown below in Figure 10.

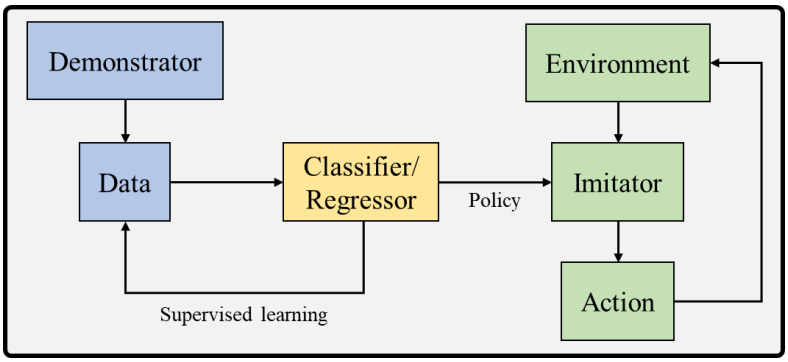


Figure 10 Imitation Learning Concept

Data Acquisition and Preprocessing

To obtain the human expert data for training, we established a connection node based on ROS (Robot Operating System) to save the images from the simulation environment to the training database in the machine learning server, as shown in Figure 11. ROS is a meta-operating system that provides various libraries and functionalities for various types of robot control mechanisms and simulations [28]. A Gazebo simulator was used to provide the simulation environment and a RotorS model was used for the training and testing of this study. The training environment used to obtain the training data is shown in Figure 12 (a) and an example of the drone image is shown in Figure 12 (b).

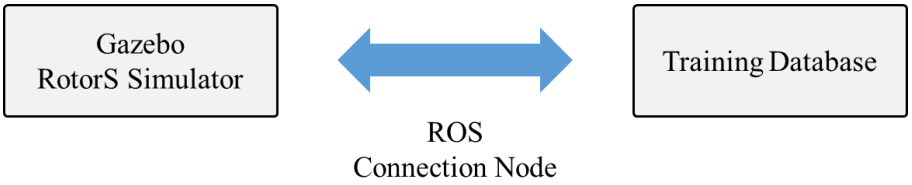
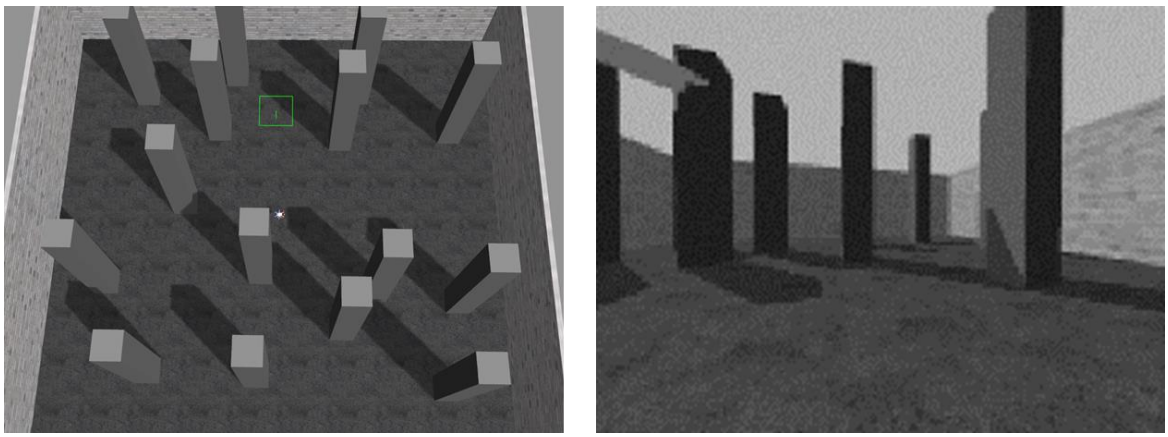


Figure 11 Data Acquisition System



(a) Training Environment (b) Drone Image Example

Figure 12 Training Environment and Image Example

The training images were obtained at 15 frames per second, and each image was down-sampled to a grayscale image. The image resolution was 120x160, and a total 2 hours of training data was acquired. The training data specifications are shown in Table 1.

Table 1: Training Data Specifications

Parameters	Scale
Image Resolution	120x160
Channel	1 (grayscale)
Sampling Rate	15 fps
Control Directions	3
Total Duration	2 hours

One corresponding control input from the human expert was saved into the database with a timestamp so that it could be matched with the image data during the training process. For simplicity, three directions in total: left, right, and straight was used, and the control input directions were applied to the drone in an incremental manner so that the heading angle of the drone increases or decreases depending on the extent or duration of the previous control input directions. One-hot encoding was applied to the control input data before the training process was done. One-hot encoding is a way of encoding the categorical integer features in a one-hot numeric array [29], which is widely used throughout machine learning in many cases where the categorical variables do not have an ordinal relationship.

After the training data was fully collected, the total dataset was divided into a training and test dataset with a ratio of 7:3, i.e., 70% of the data is only used for the training process whereas 30% of the data is only used for testing purposes and is not used during the training process in order to prevent and effectively test overfitting issues.

Proposed Framework

The final layout of the proposed imitation learning framework for obstacle avoidance is shown below in Figure 13. The system mainly consists of three phases. First the training videos and its corresponding expert decision data is aggregated into a certain dataset, where preprocessing is done for the ease of training. Next, the collected data is used to the train the 3D-CNN model, and applied to the simulated drone when the model is successfully trained. Below in Figure 13 is an illustration of the human data-based imitation learning framework for UAV obstacle avoidance in this study.

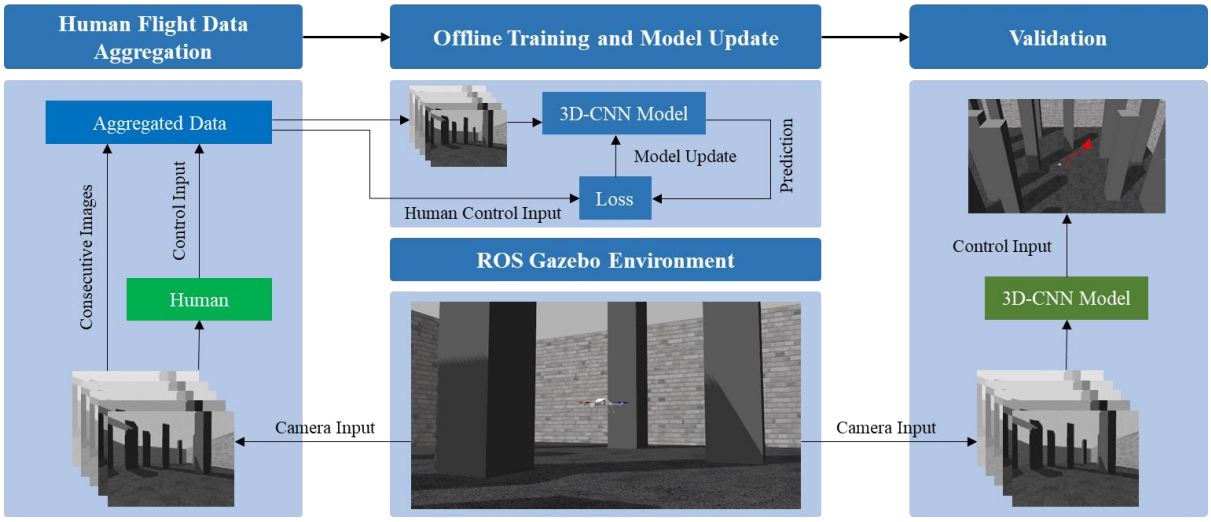


Figure 13 Proposed Imitation Learning Framework

Network Architecture and Training

The architecture of the 3D CNN used in this study is shown in Figure 14. Four convolutional layers, two max pooling layers, and one fully connected layer was used in the model. The size of the convolution filter and pooling kernel were 3x3x3 and 2x2x2, respectively. Two convolutional layers were used as a set prior to the pooling layers, where 8 channels were used in the first set of convolutional layers and 16 channels were used in the second set of convolutional layers.

The input of the neural network was reshaped into a 15x120x160x1 tensor, where 15 represents the time step of the input videos, 120x160 represents the input dimension, and 1 represents the number of color channels, indicating that a single channel grayscale image was used. Despite the given time step of 15, we connected the most recent image to the fully connected layer as the decision for obstacle avoidance is mainly affected by more recent states. The architecture of the 3D CNN is illustrated in Figure 14.

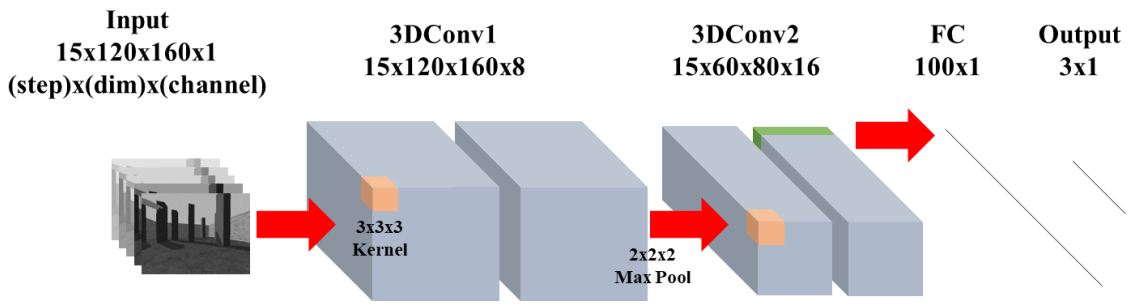


Figure 14 Architecture of 3D CNN Network

Loss Function

Among various types of loss functions, cross entropy with logits is one of the most commonly used in classification problems for deep learning. The cross entropy [30] is as shown below in Equation (8).

$$H(p,q) = -\sum_i p_i \log q_i = -y \log\hat{y} - (1-y)\log(1-\hat{y}) \,, \tag{8}$$

Here, p_i and q_i represent the true probability and predicted values of the given model, where $p_i \in \{y, 1-y\}$, and $q_i \in \{y, 1-y\}$. By using logits with the cross entropy, we are able to use a non-negative cost function and stretch the values for correct and incorrect predictions. This loss function is optimized with respect the weights and biases of the neural network given the training data.

Hyperparameters for Training

The hyperparameters for the training of the 3D CNN are as shown below in Table 2. The total number of iterations was set to 5,000 and a batch size of 50 was selected to feedforward the input data. The neural network was optimized with the Adam optimizer with a learning rate of 0.005, and the cross entropy with logits function was used as the loss function as described in the previous section.

Table 2: 3D CNN Training Parameters

Parameters	Value
Iterations	5,000
Batch Size	50
Learning Rate	0.005
Loss Function	Cross entropy with logits
Optimizer	Adam

Results and Validation

In this section, the training results of the 3D CNN is analyzed and compared with a CNN and fully connected DNN (Deep Neural Network). The three networks will be tested and compared in terms of prediction accuracy, and the obstacle avoidance performance with respect to the success rate in each test scenario will be validated as well.

Experimental Setup

The training of the neural networks were done offline on a separate machine learning server. The server runs Ubuntu 16.04 with 3 NVIDIA TITAN Xp GPUs, and the learning is implemented in Python 3.6 with Tensorflow 1.4 GPU.

The weights and biases of the trained models are transferred to the local machine that runs the simulation, and in addition to the accuracy evaluation of each model, real-time obstacle avoidance success rates will be compared. In order to determine the obstacle avoidance capability of each neural network, four additional test scenario maps were designed. The obstacle layouts were randomized in a way that patterns from the test maps were not foreseen during the training phase. The test scenario maps are shown below in Figures 15 and 16.

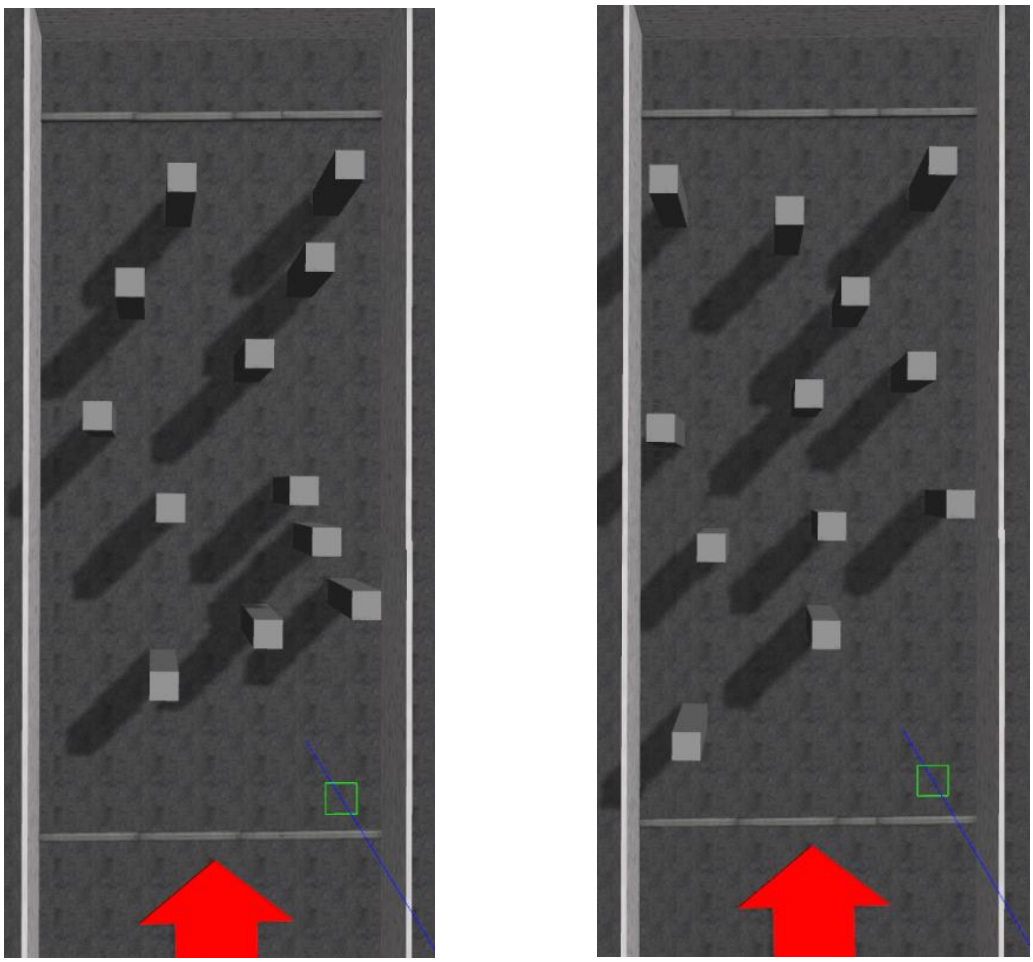


Figure 15 Test Scenario Map 1 and 2

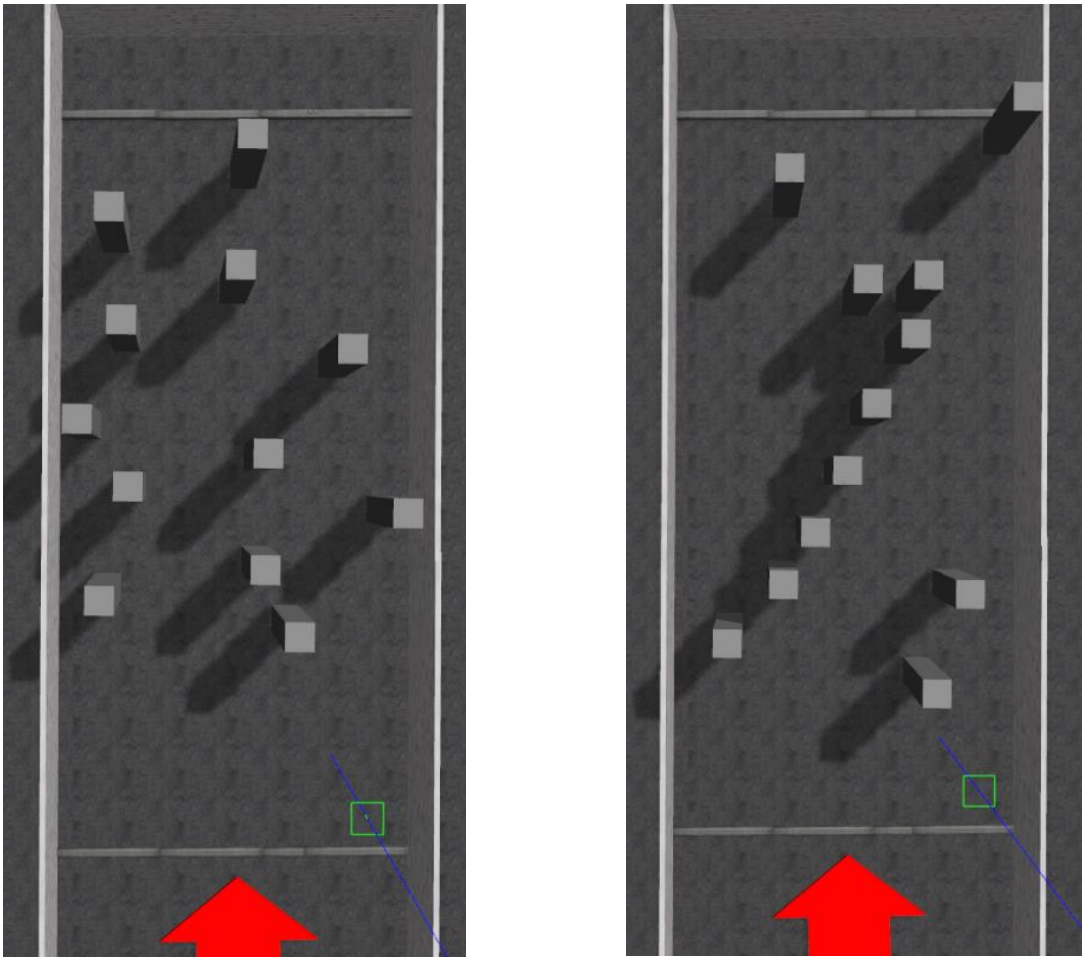


Figure 16 Test Scenario Map 3 and 4

Training Results

The imitation loss of the 3D CNN proposed in this study is shown below in Figure 17. The loss converges within 1,500 iterations, and the accuracy of the three networks are compared in Table 3.

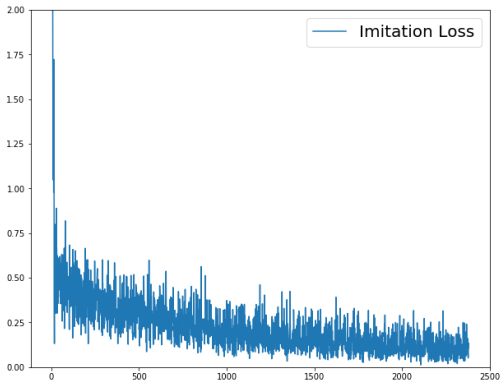


Figure 17 Training Loss of 3D CNN

The accuracy of each network is calculated based on the number of correctly predicted values for a given amount of samples. For each network, a total set of 500 input batches randomly selected from the test dataset were feedforward, where the 3D CNN exhibited the highest accuracy of the three, followed by the CNN and DNN. However, in terms of total accuracy, the 3D CNN and CNN did not exhibit a significant difference.

Table 3: Accuracy Comparison by Neural Network Type

	3D CNN	**CNN**	**DNN**
Training Accuracy	96.9%	95.2%	82.5%
Test Accuracy	82.7%	81.9%	73.4%

3D CNN

Based on the 3D CNN architecture discussed above, the confusion matrix is as shown below in Table 4. The results of the confusion matrix for each neural network was determined based on the 500 input batches that were used to calculate the classification accuracy.

Table 4: Confusion Matrix of 3D CNN

	Predicted Left	Predicted Right	Predicted Straight
True Left	0.92	0.03	0.07
True Right	0.01	0.95	0.06
True Straight	0.07	0.11	0.98

CNN

The CNN architecture shown below in Figure 18 was used for the CNN model. The confusion matrix for the CNN is shown in Table 5.

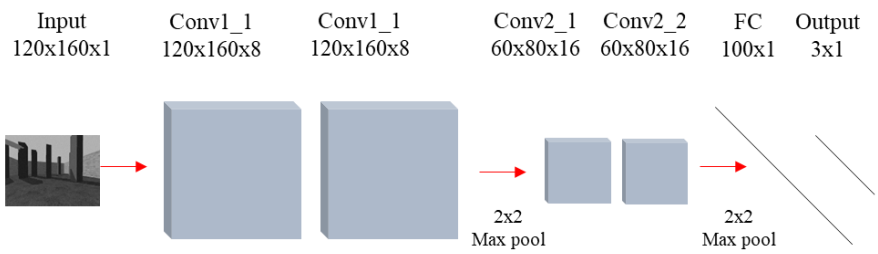


Figure 18 CNN Architecture

Table 5: Confusion Matrix of CNN

	Predicted Left	Predicted Right	Predicted Straight
True Left	0.93	0.07	0.13
True Right	0.01	0.86	0.17
True Straight	0.04	0.12	0.96

DNN

The architecture of the DNN model used for the comparison in this study is shown in Figure 16. Three fully connected layers with 100 nodes each were used for the DNN, and the input was reshaped into a 120*160 vector for the neural network. Among the three neural networks, the DNN exhibited the lowest classification accuracy. The confusion matrix for the DNN is shown in Table 6.

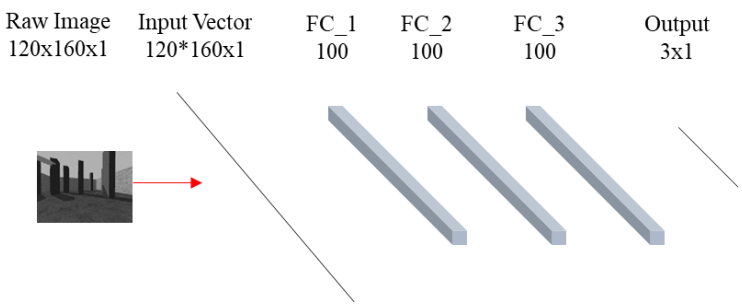


Figure 19 DNN Architecture

Table 6: Confusion Matrix of DNN

	Predicted Left	Predicted Right	Predicted Straight
True Left	0.77	0.03	0.23
True Right	0.01	0.83	0.01
True Straight	0.23	0.05	0.94

Performance Evaluation

Success Rate Evaluation

The neural networks were tested upon ten times each with respect to the four test scenario maps. The collision avoidance success or failure was determined by the distance between the drone and the obstacles. If the drone is positioned within 0.05 meters of any obstacle, the scenario is regarded as a failure and the simulation is stopped.

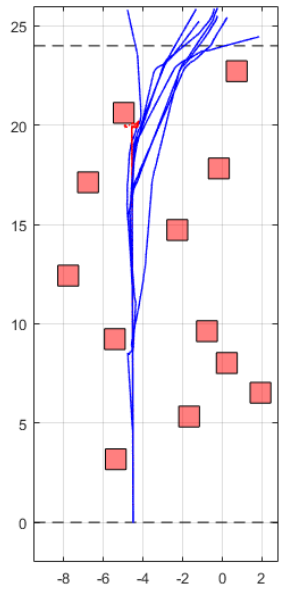
The success rates of each neural network regarding the different test maps are shown in Table 7. The 3D CNN exhibited the highest success rate among the three neural networks. For the test maps 1, 2, and 3 the 3D CNN showed near 100% success rates. However for test map 4, due to the unseen layout of obstacles, only the 3D CNN succeeded at least once, indicating that test map 4 was a great challenges for each of the neural networks.

Table 7: Success Rate by Neural Network Type and Test Map

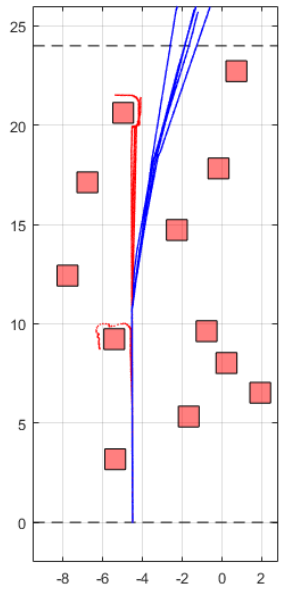
	Test Map 1			**Test Map 2**			**Test Map 3**			**Test Map 4**		
	3D CNN	**CNN**	**DNN**	**3D CNN**	**CNN**	**DNN**	**3D CNN**	**CNN**	**DNN**	**3D CNN**	**CNN**	**DNN**
Success Rate	80%	50%	10%	80%	40%	20%	90%	50%	10%	30%	0%	0%

Test Map 1 Trajectories

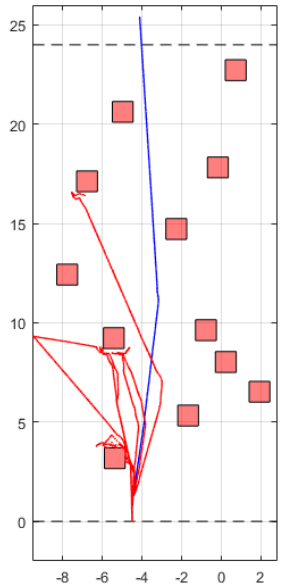
The trajectories of each neural network for test map 1 are shown in Figure 20. The red lines indicate failed cases, whereas the blue lines indicate the successful trajectories of the drone. The position information of the drone was sampled at 15 Hz.



(a) 3D CNN



(b) CNN

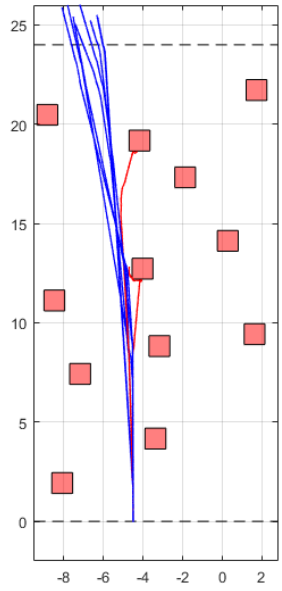


(c) DNN

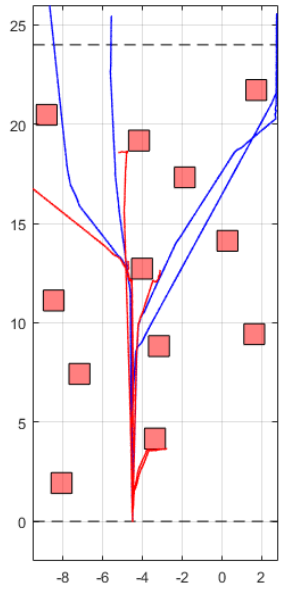
Figure 20 Test Map 1 Trajectories

Test Map 2 Trajectories

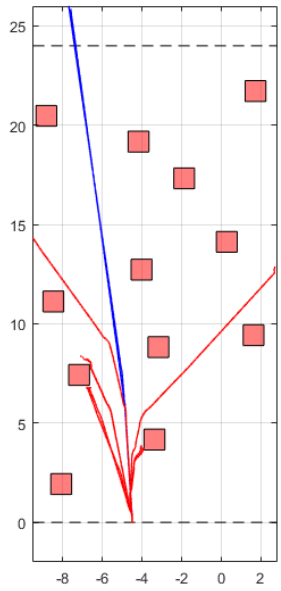
The trajectories of each neural network for test map 2 are shown in Figure 21.



(a) 3D CNN



(b) CNN

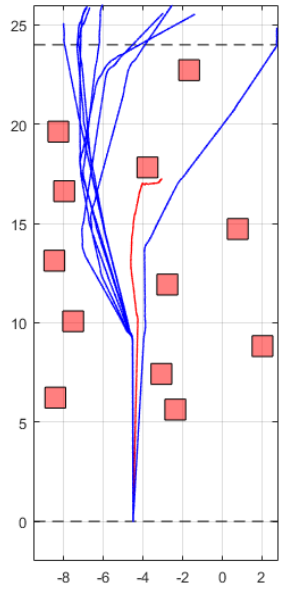


(c) DNN

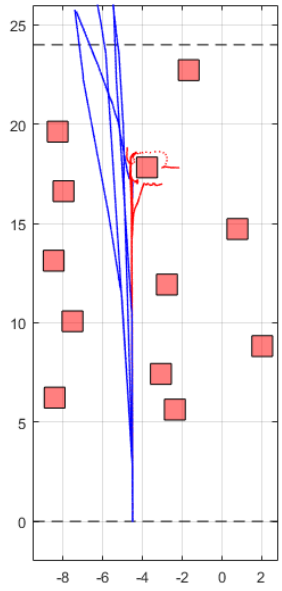
Figure 21 Test Map 2 Trajectories

Test Map 3 Trajectories

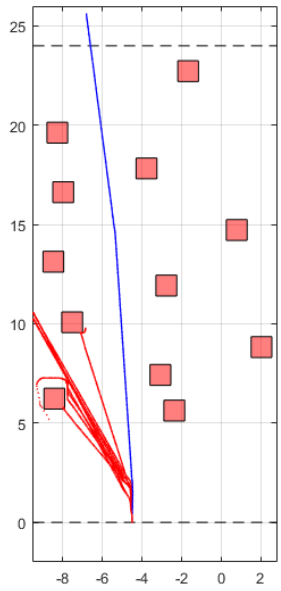
The trajectories of each neural network for test map 3 are shown in Figure 22.



(a) 3D CNN



(b) CNN

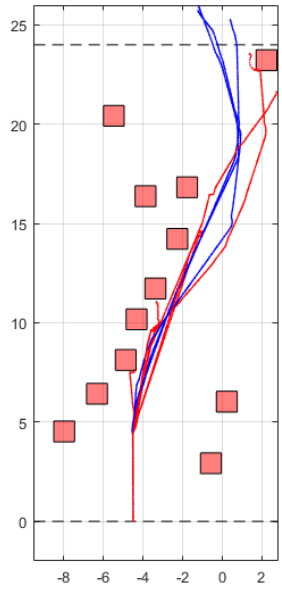


(c) DNN

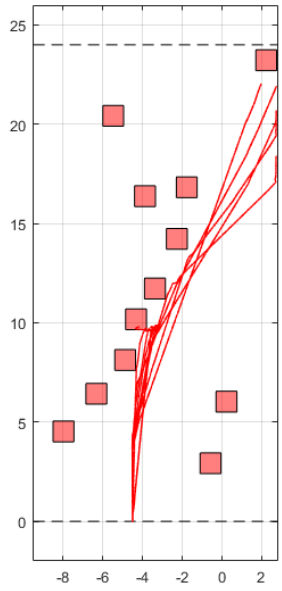
Figure 22 Test Map 3 Trajectories

Test Map 4 Trajectories

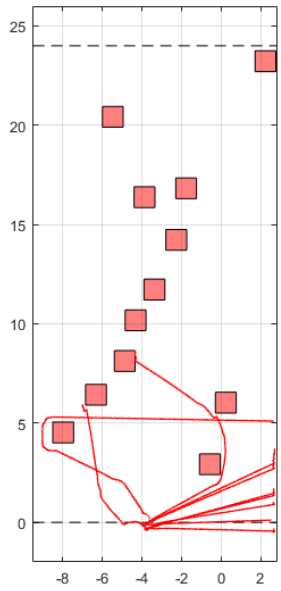
The trajectories of each neural network for test map 4 are shown in Figure 23.



(a) 3D CNN



(b) CNN



(c) DNN

Figure 23 Test Map 4 Trajectories

Farthest Point Evaluation

In addition to the overall success rates and individual trajectories of each trial, we analyzed the trajectories of the drones with respect the distribution of the farthest points for each network in each map. Because the success rate only determines the success and failure of the test scenario in a Boolean manner, we are able to more effectively analyze the performance of each network by visualizing the distribution of the farthest points as the distance traveled upon each trial reflects the obstacle avoidance capabilities of the network.

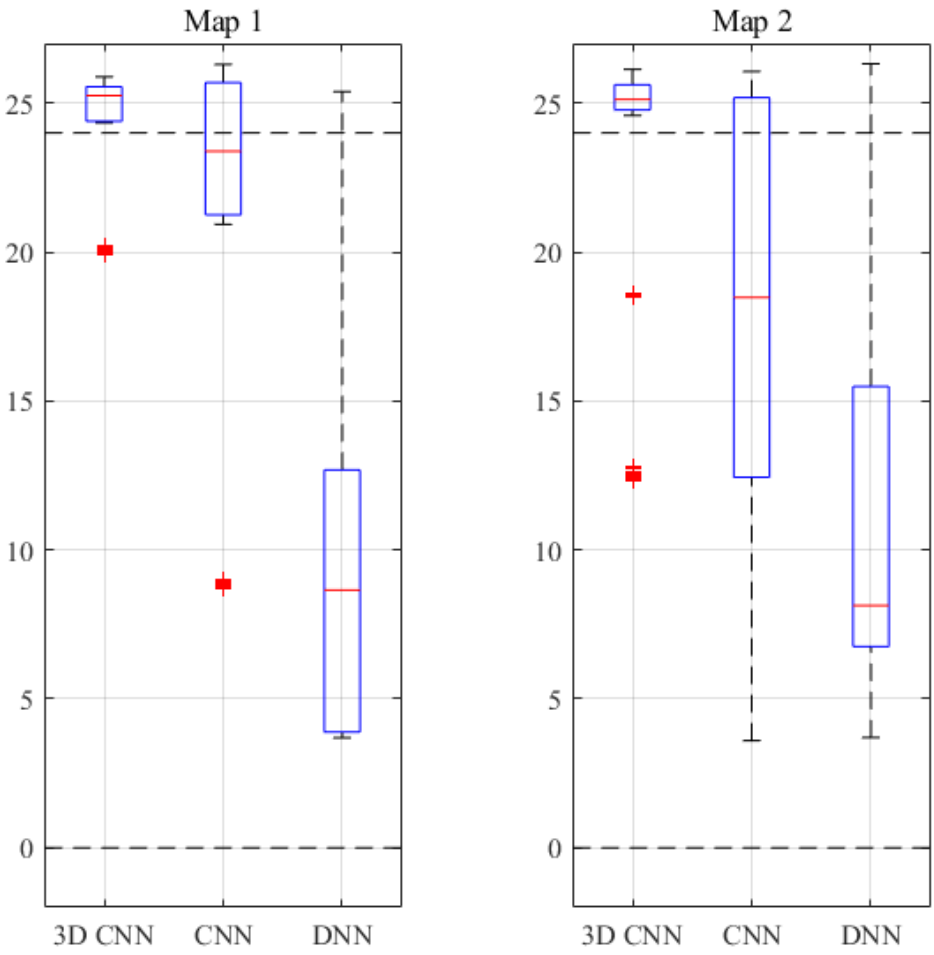


Figure 24 Box Whisker Plot for Farthest Points in Maps 1 and 2

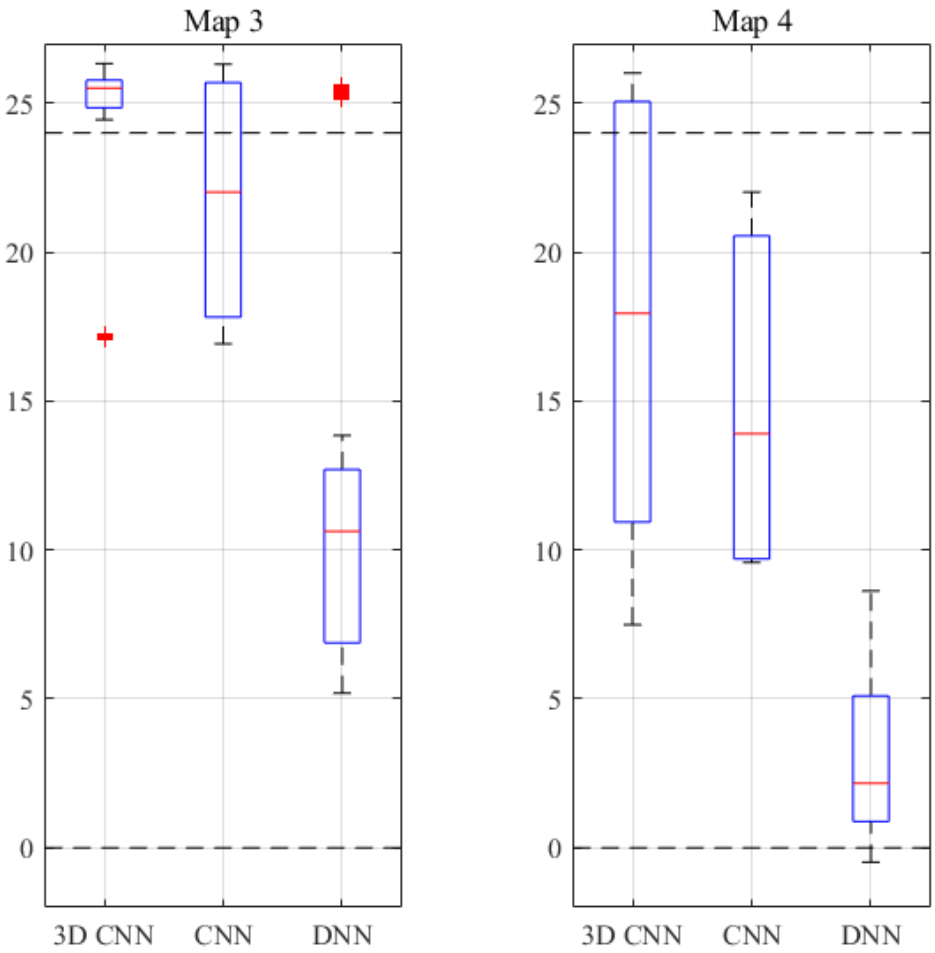


Figure 25 Box Whisker Plot for Farthest Points in Maps 3 and 4

From the box whisker plots of the three neural networks for each map, the 3D CNN exhibited the best performance in terms of traveled distance as well as consistency in results. The horizontal lines in each figure represent the starting and end points, with 0 as the starting point and 24 as the end point; if the farthest traveled point is beyond the value 24, the drone or neural network has successfully finished the trial. The upper and lower limits of each whisker represents the maximum and minimum values, and the red line indicates the mean value. The upper and lower edges of the box represent the upper and lower quartile of the data distribution. In maps 1, 2, and 3 the CNN also showed that it was capable of crossing more than half of the map in most trials. The DNN however, was not able to successfully demonstrate obstacle avoiding capabilities as most of the farthest traveled points were located in the first half of the map.

Based on the distributions obtained for each scenario, the mean values of the maximum traveled distance for each case were compared. The actual values are shown in Table 8, and the values are visualized for comparison purposes in Figure 26.

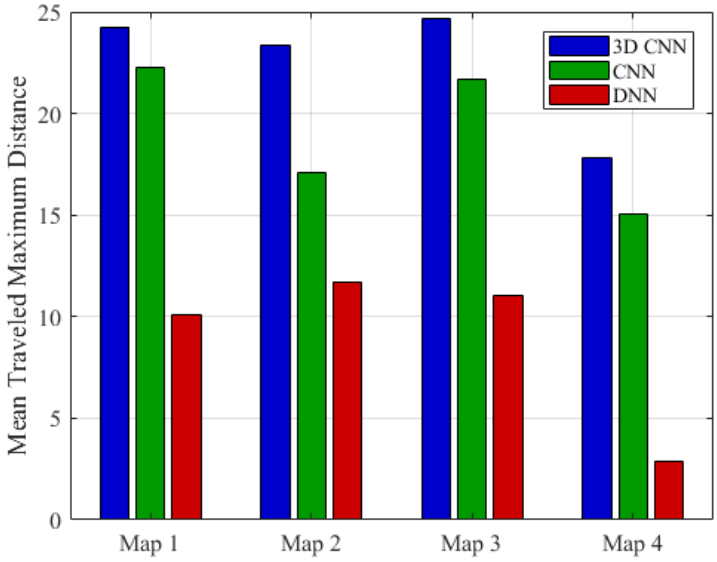


Figure 26 Mean Traveled Maximum Distance Comparison for all Scenarios

Table 8: Mean Traveled Maximum Distance Comparison for all Scenarios

	Test Map 1			**Test Map 2**			**Test Map 3**			**Test Map 4**		
	3D CNN	**CNN**	**DNN**	**3D CNN**	**CNN**	**DNN**	**3D CNN**	**CNN**	**DNN**	**3D CNN**	**CNN**	**DNN**
Mean Max. Distance	24.25	22.27	10.08	23.38	17.07	11.72	24.64	21.69	11.06	17.85	15.05	2.91

Trajectory Point Distribution Analysis

One additional metric to evaluate the performance of each neural network is to obtain an overall distribution of the points of the trajectory for each scenario. The map was divided into square grids with 0.8 as the length of each edge. Each square grid was regarded as bin, where the total number of points that fall into the bin were counted. Based on the counted points, the values were normalized and visualized in the following figures. The bins that were used to obtain the point distribution are shown below in Figure 27. The density of each bin is represented with a color value between blue and yellow, with yellow being the highest value, i.e., yellower bins indicate that the bin is denser with trajectory points compared to others.

Image data in particular is especially sensitive to factors such as noise and light, hence a robust image-based control algorithm should be able to return consistent values. For a neural network to exhibit ideal obstacle avoidance capabilities, more yellow bins should be present throughout the navigation path since consistent performance is an important criterion.

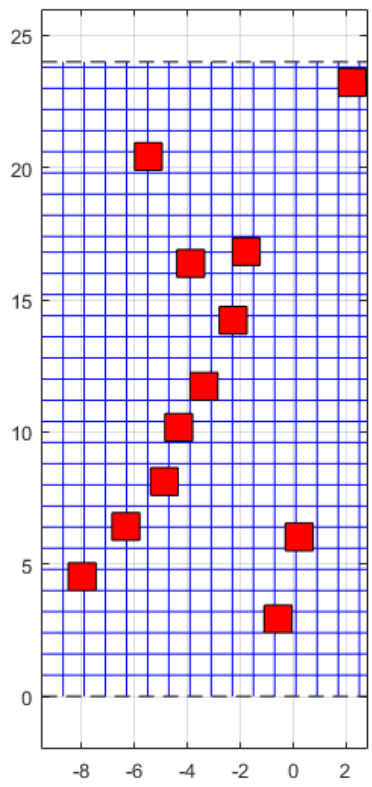
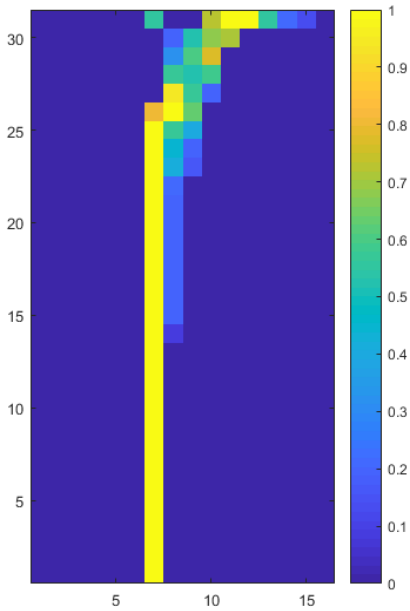
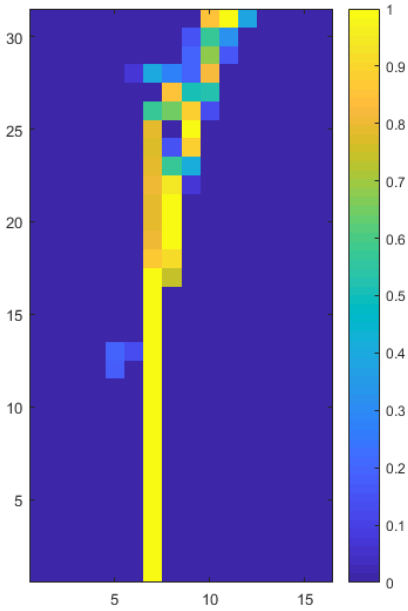
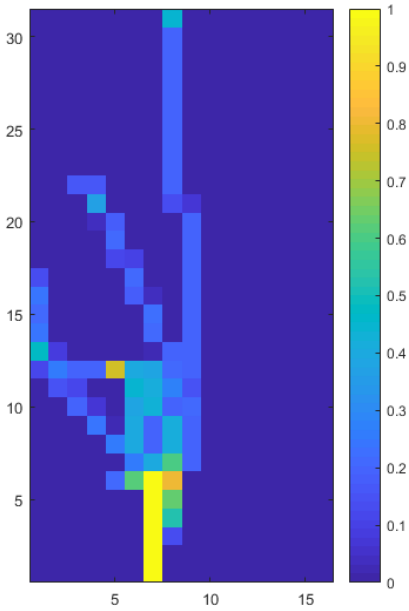


Figure 27 Grid for Point Distribution Calculation



(a) 3D CNN



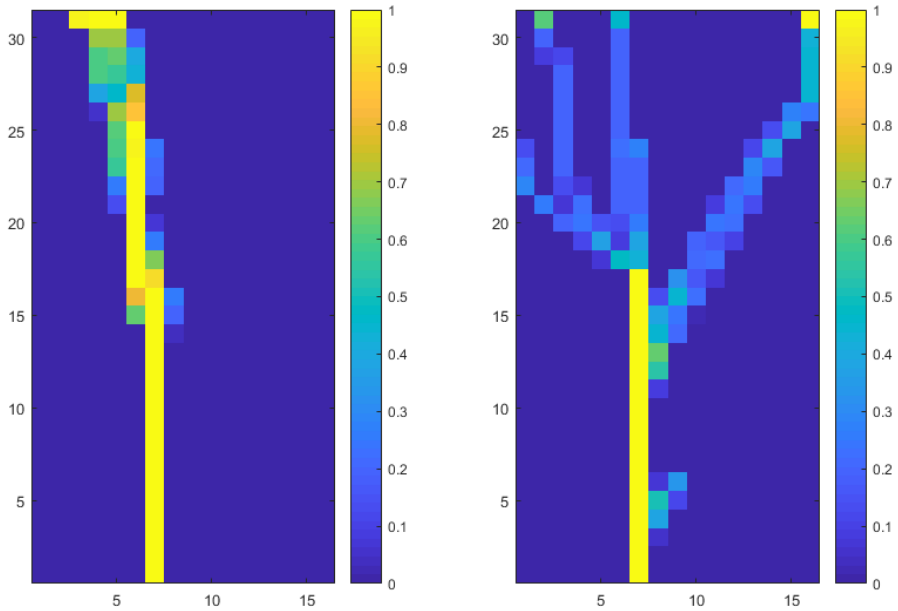


(b) CNN (c) DNN

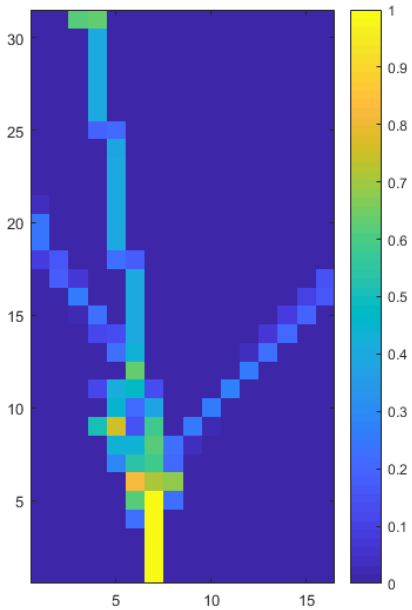
Figure 28 Distribution of Trajectory Points throughout Map 1

From Figure 28, it can be noticed that the 3D CNN and CNN both exhibit consistent performances considering the point distribution of the trajectory points that fall into each bin. Nevertheless, the trajectory point distribution of the CNN had some variations throughout the last quarter of the map,

which implies a weaker consistency with respect to the performance of the 3D CNN. The DNN on the other hand, showed values lesser than 0.4 for all of the points beyond the half point of the map, indicating that not many trajectory points were located further throughout the map.

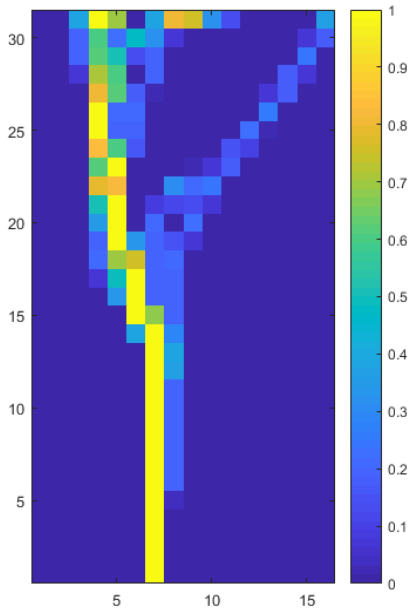


(a) 3D CNN (b) CNN

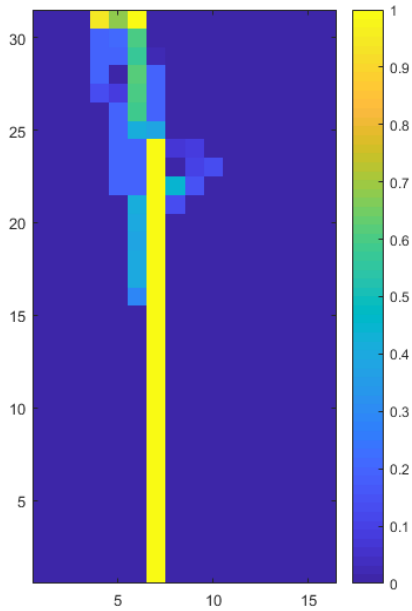


(c) DNN

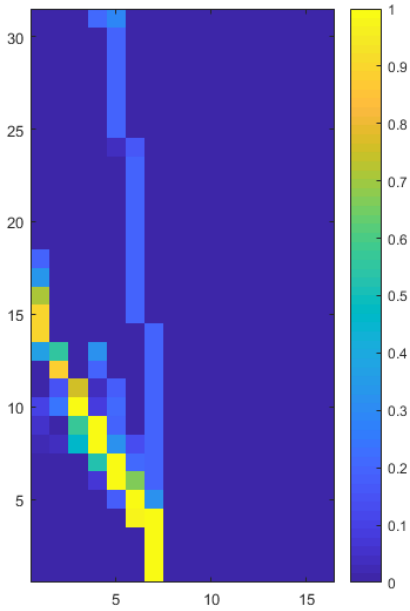
Figure 29 Distribution of Trajectory Points throughout Map 2



(a) 3D CNN

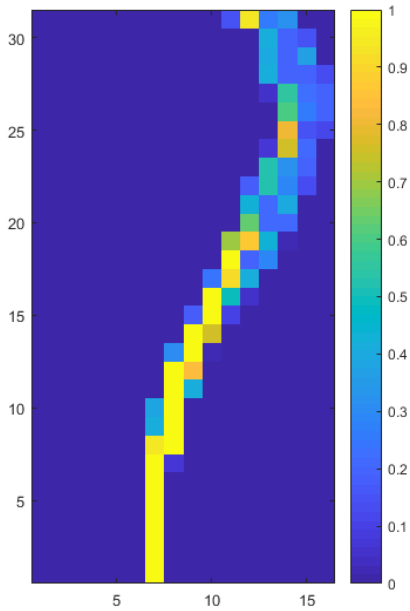


(b) CNN

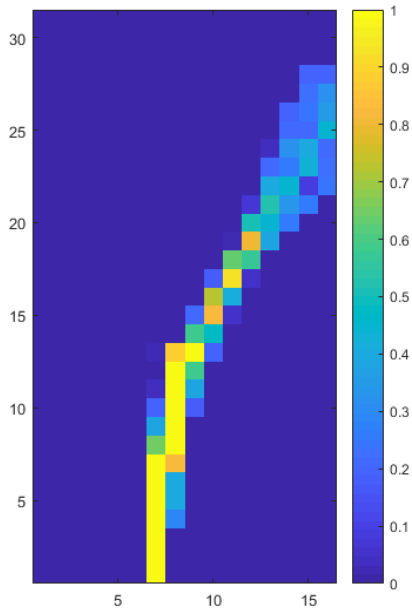


(c) DNN

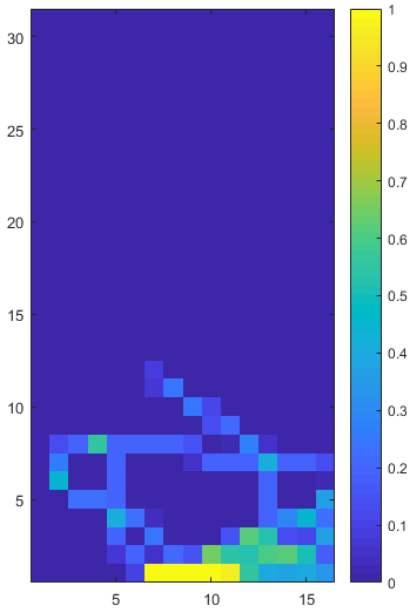
Figure 30 Distribution of Trajectory Points throughout Map 3



(a) 3D CNN



(b) CNN



(c) DNN

Figure 31 Distribution of Trajectory Points throughout Map 4

In Figure 29, the difference between the 3D CNN and CNN is more noticeable. The CNN showed point distributions with lower values near the end of the map, indicating that variances regarding the performance was present. However, in Figure 30, the 3D CNN exhibited more variance than the CNN model. We can infer that the 3D CNN may have fluctuations regarding its performance in certain obstacle layout conditions.

For the results for map 4 in Figure 31, as the 3D CNN was the only neural network capable of avoiding obstacles, the trajectory point distribution was 0 for most of the values near the end point in the CNN and DNN case. Although the 3D CNN had only exhibited a success rate of 30 % for map 4, it can still be noticed that the performance of the neural network was consistent.

CONCLUSION AND FUTURE WORK

An imitation learning framework using 3D CNNs for obstacle avoidance has been investigated in this paper. Through our experiments with a drone in a simulated environment, the 3D CNN model exhibited human-like behavior, and at the same time we were able to achieve acceptable results for obstacle avoidance without the utilization of complicated neural network structures, proving the feasibility of applying imitation learning to drones to conduct a certain task. Through comparison with a CNN and DNN, the 3D CNN exhibited superior results with respect to the success rate, consistency, and mean traveled distance.

Deep learning technologies have been able to outperform many of the existing algorithms through various fields by automatically extracting features for classification and regression. Yet, the largest drawback of deep learning is that it is considered a black box model, making it hard to interpret and analyze.

For future studies, the analysis of the neural networks should be done in order to make improvements to the current system in terms of performance. By analyzing the activated neurons within the hidden layers, we expect to obtain an understanding of the automatically selected features and rules from the neural network. In addition, as a goal point is not considered in the current task, we plan to implement goal-based obstacle avoidance and navigation, in a more complex environment. Our ultimate goal is to apply the imitation learning framework to an actual system with real images.

REFERENCES

[1] Chu, K., Lee, M., & Sunwoo, M. (2012). Local path planning for off-road autonomous driving with avoidance of static obstacles. IEEE Transactions on Intelligent Transportation Systems, 13(4), 1599-1616.
[2] Borenstein, J., & Koren, Y. (1988). Obstacle avoidance with ultrasonic sensors. IEEE Journal on Robotics and Automation, 4(2), 213-218.
[3] Borenstein, J., & Koren, Y. (1991). The vector field histogram-fast obstacle avoidance for mobile robots. IEEE transactions on robotics and automation, 7(3), 278-288.
[4] Souhila, K., & Karim, A. (2007). Optical flow based robot obstacle avoidance. International Journal of Advanced Robotic Systems, 4(1), 2.
[5] Song, K. T., & Huang, J. H. (2001). Fast optical flow estimation and its application to real-time obstacle avoidance. In Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on (Vol. 3, pp. 2891-2896). IEEE.
[6] Michels, J., Saxena, A., & Ng, A. Y. (2005, August). High speed obstacle avoidance using monocular vision and reinforcement learning. In Proceedings of the 22nd international conference on Machine learning (pp. 593-600). ACM.
[7] Huang, B. Q., Cao, G. Y., & Guo, M. (2005, August). Reinforcement learning neural network to the problem of autonomous mobile robot obstacle avoidance. In Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on (Vol. 1, pp. 85-89). IEEE.
[8] Kahn, G., Zhang, T., Levine, S., & Abbeel, P. (2017, May). Plato: Policy learning using adaptive trajectory optimization. In Robotics and Automation (ICRA), 2017 IEEE International Conference on (pp. 3342-3349). IEEE.
[9] Pomerleau, D. A. (1989). Alvinn: An autonomous land vehicle in a neural network. In Advances in neural information processing systems (pp. 305-313).
[10] Sammut, C., Hurst, S., Kedzier, D., & Michie, D. (1992). Learning to fly. In Machine Learning Proceedings 1992 (pp. 385-393).
[11] Ross, S., Gordon, G., & Bagnell, D. (2011, June). A reduction of imitation learning and structured prediction to no-regret online learning. In Proceedings of the fourteenth international conference on artificial intelligence and statistics (pp. 627-635).
[12] Ross, S., Melik-Barkhudarov, N., Shankar, K. S., Wendel, A., Dey, D., Bagnell, J. A., & Hebert, M. (2013, May). Learning monocular reactive uav control in cluttered natural environments. In 2013 IEEE international conference on robotics and automation (pp. 1765-1772). IEEE.
[13] Kim, D. K., & Chen, T. (2015). Deep neural network for real-time autonomous indoor navigation. arXiv preprint arXiv:1511.04668.
[14] McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 5(4), 115-133.
[15] Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. Psychological review, 65(6), 386..
[16] Minsky, M., Papert, S. A., & Bottou, L. (2017). Perceptrons: An introduction to computational geometry. MIT press.
[17] Rumelhart, D. E., McClelland, J. L., & PDP Research Group. (1987). Parallel distributed processing (Vol. 1, p. 184). Cambridge, MA, USA:: MIT press.
[18] LeCun, Y., Kavukcuoglu, K., & Farabet, C. (2010, May). Convolutional networks and applications in vision. In ISCAS (Vol. 2010, pp. 253-256).

[19] Han, J., & Moraga, C. (1995). The influence of the sigmoid function parameters on the speed of backpropagation learning. From Natural to Artificial Neural Computation, 195-201.
[20] Hahnloser, R. H., Sarpeshkar, R., Mahowald, M. A., Douglas, R. J., & Seung, H. S. (2000). Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. Nature, 405(6789), 947-951.
[21] Glorot, X., Bordes, A., & Bengio, Y. (2011, June). Deep sparse rectifier neural networks. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (pp. 315-323).
[22] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.
[23] Khvostikov, A., Aderghal, K., Benois-Pineau, J., Krylov, A., & Catheline, G. (2018). 3D CNN-based classification using sMRI and MD-DTI images for Alzheimer disease studies. arXiv preprint arXiv:1801.05968.
[24] Lan, S., Yu, R., Yu, G., & Davis, L. S. (2018). Modeling Local Geometric Structure of 3D Point Clouds using Geo-CNN. arXiv preprint arXiv:1811.07782.
[25] Barea, R., Pérez, C., Bergasa, L. M., López-Guillén, E., Romera, E., Molinos, E., ... & López, J. Vehicle Detection and Localization using 3D LIDAR Point Cloud and Image Semantic Segmentation.
[26] Ren, J., Reyes, N. H., Barczak, A. L. C., Scogings, C., & Liu, M. (2018, June). An Investigation of Skeleton-Based Optical Flow-Guided Features for 3D Action Recognition Using a Multi-Stream CNN Model. In 2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC) (pp. 199-203). IEEE.
[27] Shah, A. K., Ghosh, R., & Akula, A. (2018, September). A spatio-temporal deep learning approach for human action recognition in infrared videos. In Optics and Photonics for Information Processing XII (Vol. 10751, p. 1075111). International Society for Optics and Photonics.
[28] Cashmore, M., Fox, M., Long, D., Magazzeni, D., Ridder, B., Carrera, A., ... & Carreras, M. (2015, June). ROSPlan: Planning in the Robot Operating System. In ICAPS (pp. 333-341).
[29] Harris, David and Harris, Sarah. Digital design and computer architecture (2nd ed.). San Francisco, Calif.: Morgan Kaufmann. p. 129. ISBN 978-0-12-394424-5.
[30] Bishop, C. M. (2006). Pattern recognition and machine learning. springer.