A picture containing text

Description automatically generated

An investigation into the use of artificial intelligence with photogrammetry to predict unity object primitives.

Paul Kerins

T00188667

Table of Contents

[Chapter 2: Literature review 2](#_Toc152860096)

[2.1 Investigating Photogrammetry 2](#_Toc152860097)

[2.1.1 What is Photogrammetry 2](#_Toc152860098)

[2.1.2 Principles of Photogrammetry 2](#_Toc152860099)

[2.1.3 Uses of Photogrammetry 3](#_Toc152860100)

[2.1.4 Aerial Photogrammetry 3](#_Toc152860101)

[2.1.5 Terrestrial Photogrammetry 3](#_Toc152860102)

[2.1.6 Close Range Photogrammetry 4](#_Toc152860103)

[2.2 Investigating Artificial Intelligence 4](#_Toc152860104)

[2.2.1 What is Artificial Intelligence 4](#_Toc152860105)

[2.2.2 What is Machine Learning 4](#_Toc152860106)

[2.2.3 Supervised Learning 4](#_Toc152860107)

[2.2.4 Unsupervised Learning 5](#_Toc152860108)

[2.2.5 Reinforced Learning 5](#_Toc152860109)

[2.2.6 What is Neural Networks 6](#_Toc152860110)

[2.2.7 Keras MNIST 6](#_Toc152860111)

[Chapter 3: Methodology 7](#_Toc152860112)

[3.1 Research Undertaken 7](#_Toc152860113)

[3.2 Research Question 7](#_Toc152860114)

[3.3 Proposed Project Implementation 7](#_Toc152860115)

[3.4 Functional Design 7](#_Toc152860116)

[3.4.1 Risk Analysis 7](#_Toc152860117)

[3.5 Function Specification 8](#_Toc152860118)

[3.6 Prototype 1 8](#_Toc152860119)

[4. Implementation 9](#_Toc152860120)

[4.1 Sprint 1 9](#_Toc152860121)

[Task 1 9](#_Toc152860122)

[Task 2 10](#_Toc152860123)

[Results 11](#_Toc152860124)

[Abbreviations 12](#_Toc152860125)

[References 13](#_Toc152860126)

# Chapter 2: Literature review

## 2.1 Investigating Photogrammetry

### 2.1.1 What is Photogrammetry

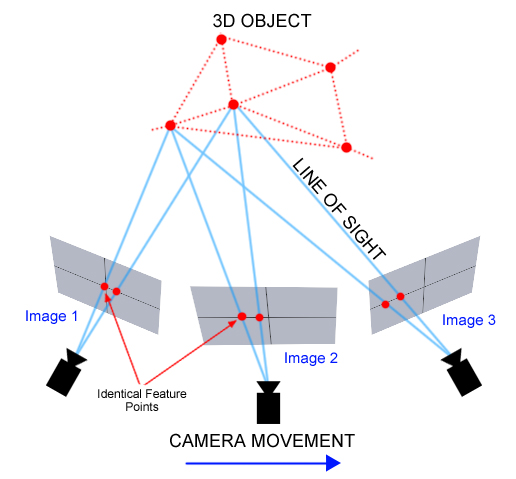
According to the book Small-format Aerial Photography “Photogrammetry is the technology of obtaining reliable information about physical objects and environment through processes of recording, measuring, and interpreting photographic images” (James S.Aber, 2010). Photogrammetry main goal is making precise measurements of three-dimensional objects and terrain using photos taken of the object or terrain. Ther photos taken are used to gather measurements from the photo interpretation and geometric relationship. From the data gathered from photogrammetry, maps and 3D models of real-world scenes can be created.

There are three types of photogrammetry:

* Aerial
* Terrestrial
* Close Range

The two forms of photogrammetry are:

* Interpretive
* Metric



*Figure 2.1.1.1 Photogrammetry Diagram*

### 2.1.2 Principles of Photogrammetry

Photogrammetry uses the concept of triangulation. “Triangulation involves taking pictures from a minimum of two different locations” (Marketing, 2023). Using the pictures taken creates lines of sight from each camera to specific points on the photographed object. The intersection of these lines helps produce 3D coordinates of the specific points. There are necessary aspects for any photogrammetric models. These features are:

* Tie points
* Ground control points (GCP)
* Bundle adjustment

Tie points are coordinates that are linked across multiple overlapping images. These points are present in all the photos of the object. Tie points help the photo adjust with the shared coordinates.

Ground control points help to the image’s orientation in relation to the Earth's surface. GCP uses known coordinates to position the image in the real world.

Bundle adjustment helps to remove any distortion that may occur within a set of images. It helps to reduce the errors from real and predicted image points.

### 2.1.3 Uses of Photogrammetry

Photogrammetry is used in a wide range of different industries. Photogrammetry is commonly used for creating maps out of aerial photos. Orthomosaics, Digital surface models and digital terrain models are products that come from photogrammetry. Orthomosaic is a birds-eye view of a terrain that adjusts for distortion and can span over wide landscapes. “Digital surface models and digital terrain models represent surface levels and elevations” (Marketing, 2023). Surface models includes the whole environment like trees and buildings whereas the terrain model gets only just the height of the earth while ignoring the rest of the features. Photogrammetry is used in land surveying, engineering, real estate, military intelligence, medicine, film and entertainment, forensics, construction and mining, sports, agriculture, and forestry.

A computer screen shot of a mouse

Description automatically generated

Figure 2.1.3.1 Example of the use of Photogrammetry in 3D modelling

### 2.1.4 Aerial Photogrammetry

Aerial photogrammetry is a technique for creating two dimensional or three-dimensional models from aerial photographs, which are usually taken from an airplane or airborne craft such as a drone. Aerial photogrammetry is usually taken from two or more angles of the same area to map the area accurately. Aerial photogrammetry is mainly used to create topographical maps. “These maps may be used as a basis for, or in conjunction with, Geographic Information System data” (Goodman, 2023). Aerial photogrammetry mapping methods are commonly used in a variety of different industries such as architecture, land development, environmental studies of terrain like watershed and deforestation research. The reason for the photographs being taken by a plane is due to the flight pattern. The flight pattern allows for each area to have multiple pictures taken from different angles. Photogrammists need multiple angles of the area to determine an accurate position of objects within the photographs.

### 2.1.5 Terrestrial Photogrammetry

Terrestrial photogrammetry is a powerful and versatile technique used in geospatial mapping, surveying, and 3D modelling of objects and landscapes on the Earth's surface. It involves the acquisition and analysis of photographs taken from fixed terrestrial positions, allowing for the precise measurement of distances, angles, and the three-dimensional coordinates of various features. By utilizing multiple images of the same area from different viewpoints, photogrammetry software can triangulate and calculate the positions of points, making it an essential tool for applications such as urban planning, architecture, civil engineering, and environmental monitoring. Terrestrial photogrammetry has become increasingly advanced with the integration of digital cameras and sophisticated software, making it an invaluable resource for accurate data collection and visualization in a wide range of industries.

### 2.1.6 Close Range Photogrammetry

Close Range Photogrammetry (CRP) is defined as photogrammetric data collection and processing is less than one thousand feet away. The data collection method can be both aerial or ground based, and the final output can be rendered in two dimensional or three dimensional. CRP has seen massive interest from major tech company Google for its project Tango as it wishes to use 3D photogrammetry. As Tango will incorporate depth-sensing and motion-sensing cameras into an Android smartphone. The use of CRP will allow for the collection of 3D data and the producing of 3D imagery.

## 2.2 Investigating Artificial Intelligence

### 2.2.1 What is Artificial Intelligence

Artificial intelligence (AI) refers to the simulation of human intelligence in machines, enabling them to perform tasks that typically require human intelligence. This encompasses various technologies and techniques like machine learning, deep learning, natural language processing, and computer vision, which equip AI systems to learn from data, adapt to changing situations, and make decisions or solve problems autonomously. AI finds applications in fields such as healthcare, finance, autonomous vehicles, and recommendation systems, transforming how businesses and industries operate (Russell, 2021).

### 2.2.2 What is Machine Learning

Machine Learning is the most common type of artificial intelligence. It uses algorithms trained from data sets to help create models to allow the machines to perform tasks that would otherwise only be possible for humans to complete. These tasks are ones such as predicting price fluctuations, analysing data, and categorizing images. Machine learning uses the algorithms to train the machines “to create self-learning models that are cable of predicting outcomes and classifying information with human intervention” (Coursera, 2023). Even though artificial intelligence and machine learning are often interchangeable in today’s world. The two terms are very distinct. As artificial intelligence is the general attempt to make machines have human like abilities, machine learning is specifically for the use of data sets and algorithms to make the machines have human like cognitive abilities. Machine learning is commonly used around the world, one of the main examples of machine learning is the recommendation engines that suggest shows, products or songs to the use, these engines are found on Netflix, Amazon, and Spotify.

There are three types of machine learning.

* Supervised Learning
* Unsupervised Learning
* Reinforced Learning

### 2.2.3 Supervised Learning

Supervised Learning is a type of machine learning where an algorithm learns from a labelled dataset, making predictions or decisions based on that data set. The reason why it is called supervised learning is due to there being a supervisor or teacher present to provide the algorithm with the correct outcomes or answers for a set of example inputs. In supervised learning, you start with a dataset that includes pairs of input data and their corresponding correct output, which is often referred to as the target or label. You then use a learning algorithm to analyse and learn from the training dataset. The goal is for the algorithm to recognize patterns, relationships, and features in the data that can help it make accurate predictions. Once the algorithm has been trained on the labelled data, it can be used to make predictions or classifications on new, unseen data. It uses the patterns and knowledge it gained during training to provide an output, which can be a classification label, a numeric value, or any other relevant prediction.

### 2.2.4 Unsupervised Learning

“Unsupervised learning is a type of machine learning technique that uses artificial intelligence algorithms to identify patterns in data sets that are neither classified nor labelled” (S.Gillis, 2023).In unsupervised learning, the algorithm’s objective is to discover patterns, structures, or relationships within the data on its own. It is often used for tasks such as clustering, dimensionality reduction, and density estimation. Unlike supervised learning, unsupervised learning algorithms work with a dataset that lacks labelled output. The data consists of features or attributes but doesn’t have corresponding target labels or categories. The algorithm’s goal in unsupervised leaning is to uncover hidden structures or patterns within the data. It does this through various techniques, with the most common ones being clustering and dimensionality reduction.

* Clustering algorithms group similar data points together based on the inherent similarity or proximity of data points.
* Dimensionality reduction methods reduce the complexity of the data by mapping it to a lower-dimensional space while preserving relevant information.

The output of unsupervised learning is typically the identified clusters, reduced dimensional representations of the data, or some other form of structure or pattern that the algorithm has discovered.

### 2.2.5 Reinforced Learning

Reinforcement learning is a type of machine learning where an agent learns to make sequences of decisions by interacting with an environment. The agent receives feedback in the form of rewards or penalties based on the actions it takes, and the agent’s objective is to learn a strategy that maximize the cumulative of rewards over time. Reinforcement learning is often used in applications where the agent needs to learn how to make a sequence of decisions to achieve its goal, such as playing a game and robotics. The key components of reinforcement learning are:

* The agent, that interacts with the environment. The agent observes the current state, selects actions, and receives feedback in the form of rewards.
* The environment, this is the external system that the agent interacts with.
* The actions, all the possible steps that can be taken by the agent.
* The state, which is the current condition returned by the model.
* The reward, this is to help the model move in the right direction, the reward is given to it for making the correct action.
* The policy, this determines how an agent will behave. It acts as a mapping between the action and the state. (Simplilearn, 2023)

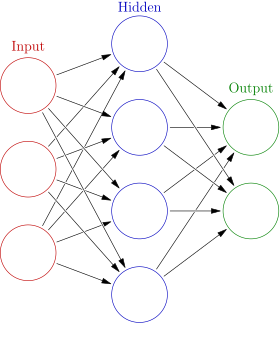
The main goal of the agent is to learn the policy, by mapping from states to actions, that earns the greatest amount of reward over time. For this to happen, the agent will have to explore different actions to understand the consequences of that action and using the knowledge gained to make better decisions. Reinforcement learning has been applied in a wide range of domains, including game playing with games such as Minecraft and Pokémon, robotic control, and recommendation systems, where learning from interaction and feedback.

### 2.2.6 What is Neural Networks

“Neural networks, also known as artificial neural networks (ANN) or simulated neural networks (SNN), are a subset of [machine learning](https://www.ibm.com/topics/machine-learning) and are at the heart of [deep learning](https://www.ibm.com/topics/deep-learning) algorithms” (IBM, 2023). ANN are a class of machine learning models inspired by the structure and function of the human brain. Neural networks consist of interconnected layers of artificial neurons and are capable of learning complex mapping from input data to output data.

The key components of neural networks are:

* Neurons are the basic building blocks of a neural network. “They receive one or more input signals. These input signals can come from either raw data set or from neurons positioned at a previous layer of the neural network” (McCullum, 2020).
* Weights, which are the “real values that are attached with each input/feature and they convey the importance of that corresponding feature in predicting the final output” (Ganesh, 2020).
* Biases, which is that each neuron has an associated bias, which helps the model account for situations where all inputs are zero.
* Summation Function “is to bind the wights and inputs together and calculate their sum” (McCullum, 2020).
* Activation Function “is used to introduce non-linearity in the model” (McCullum, 2020).
* Neurons are organized into layers. There are three primary types of layers:
* Input Layer: Receives the initial data.
* Hidden Layers: One or more Layers between the input and output layers. They extract and learn from the data.
* Output Layer: Produces the final output or predictions.



*Figure 2.2.6.1: Architecture of Artificial Neural Network*

### 2.2.7 Keras MNIST

Keras is a popular deep learning framework that is associated with supervised learning. It allows the user to easily build and train neural networks and is often used for image classification takes like the MNIST dataset. The MNIST dataset is a dataset of handwritten digits, consisting of 28x28 pixel greyscale images pf handwritten numbers from 0 to 9. The task for the machine is to build a model that can classify these images into the correct digital category. It falls under supervised learning as the dataset is labelled with the correct digit for each image, which allows the model to be trained to predict the correct digit label given a new image.

# Chapter 3: Methodology

## 3.1 Research Undertaken

The research that was taken was to gain an understanding of what photogrammetry and AI, and to gain an understanding of if they can be used together.

## 3.2 Research Question

An investigation into the use of artificial intelligence with photogrammetry to predict unity object primitives.

## 3.3 Proposed Project Implementation

Creation of a Neural Network that can predict certain values of a unity game object based on data passed in.

## 3.4 Functional Design

### 3.4.1 Risk Analysis

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Description** | **Likelihood** | **Impact** | **Mitigation Activity** | **Plan of Action** |
| **Data cannot be acquired** | Low | High | Look for an alternative way to gain the data | Create new approach of gaining data |
| **Not Enough data is available** | Low | High | Export more data from unity | Allow for longer periods of time to gain data |
| **Data does not provide a convincing output** | Medium | Medium | Investigate a different approach to the problem | Use a different neural network model approach |

Table 1 Risk Analysis Table

## 3.5 Function Specification

|  |
| --- |
| Must Have |
| Unity Scene |
| Unity Game Object Data |
| Neural network to be trained on exported data |
| Should Have |
| Second Algorithm/ Model to compare efficiency |
| Sending of messages or warnings based on comparison of loads |
| Testing on dataset with breakdowns |
| Could Have |
| Data based on real time |
| Won't Have |
| Data based on real life objects |

#### Table 2 Function Specification Table

## 3.6 Prototype 1

For prototype 1 a simple unity scene with a game object Cube was used with sphere objects placed at each corner of the Cube with a C# script to print the 2D and 3D values of the cube to a CSV file. A sklearn neural network model was then developed on collabs to try predicting the XYZ values based on the data exported to the CSV file. The neural network used the 2D values to try predicting the XYZ values. A breakdown of the tasks involved to complete prototype 1 is as follows:

|  |  |  |
| --- | --- | --- |
| **Task No.** | **Description** | **Status** |
| **1** | Set up a unity scene | Complete |
| **2** | Create Cube and sphere game objects. | Complete |
| **3** | Create C# script to export 2D and 3D variables | Complete |
| **4** | Export Cube and sphere values to CSV file | Complete |
| **5** | Set up google colab worksheet | Complete |
| **6** | Import dataset & libraries:   * Keras * Pandas * Seaborn * Pyplot * Numpy * sklearn | Complete |
| **7** | Train model with X, Y columns | Complete |
| **8** | Extract all the 2D values to predict XYZ values with statistics such as mean square error | Complete |

Table 3 Prototype 1 Task Breakdown Table

# 4. Implementation

This chapter outlines the development of the prototypes into a final solution for the problem. It consists of 4 sprints.

## 4.1 Sprint 1

This sprint deals with creating a unity project that exports the data of the cube object to a csv file which was then used in colabs to train a neural network to predict the XYZ values of the cube from the 2D screen points.

|  |  |  |
| --- | --- | --- |
| Task | Description | Status |
| 1 | Create Unity project | Complete |
| 2 | Create Cube game object | Complete |
| 3 | Create game objects for each corner of cube | Complete |
| 4 | Export data to csv file | Complete |
| 5 | Import Data to colabs | Complete |
| 6 | Train neural network | Complete |

Table 4 Sprint 1 Task Breakdown Table

### Task 1

Implementing a unity project with a cube game object. While setting up small sphere game objects to each corner of the cube to gather the position of each corner. Having to create a C# script that would gather all the relative data from the cube and print it correctly to a csv file.

In Figure 4.1.1 shows the Unity project set up with the cube and CubeData.csv file produced in the assets folder.

A screenshot of a computer

Description automatically generated

Figure 4.1.1 Unity Project Prototype

### Task 2

Creating a Google colabs and importing the correct libraries for training a neural network. Using the imported libraries on colabs to train a neural network to predict the XYZ values of the cube by passing in the 2D XY values of the cube. After the neural network was trained to predict the XY values, in Figure 4.1.2 and 4.1.3 it shows a graph of the accuracy and loss while training the neural network to demonstrate the accuracy of the network.

A screen shot of a graph

Description automatically generated

Figure 4.1.2 Colabs Graph of the loss in the training

A screen shot of a graph

Description automatically generated

Figure 4.1.3 Colabs Graph of the accuracy of the training

The neural network was then used to predict the XY values of the cube game object which can be seen in Figure 4.1.4.

A graph of different colored lines

Description automatically generated

Figure 4.1.4 Colabs Graph of the Network predicting the XY values of the Cube object.

### Results

A unity project that produces a csv file filled with data based on the cube object values and used that data to train a neural network on google colabs. With the neural network trained, it was able to predict the XY values and display the percentage accuracy and loss of the training.

## 4.2 Sprint 2

This sprint deals with producing new data of the cube by using two new cameras. The two new cameras are positioned above the cube and to the side of the cube, this will assist the neural network in learning the size and position of the cube from different perspectives. The cube will stay in one position and rotate as the previous settings the cube could be placed outside the view of the cameras. The model was modified to be less re-defined in order for the neural network to learn itself.

|  |  |  |
| --- | --- | --- |
| Task | Description | Status |
| 1 | Add new Cameras | Complete |
| 2 | Adjust the code of the cube | Complete |
| 3 | Adjust Neural network model | Complete |
| 4 | Export data to csv file | Complete |
| 5 | Import Data to colabs | Complete |

Table 5 Sprint 2 Task Breakdown Table

### Task 1

Adding two camera objects to the unity scene. The cameras were positioned above and to the side of the cube to give the neural network different perspectives on the cube.

In Figure 4.2.1 shows the Unity cameras set up in position to the cube.

A screenshot of a video game

Description automatically generated

Figure 4.2.1 Unity Cameras Set up.

### Task 2

During this sprint the cubes code was adjusted to allow the cube to stay in frame of each camera. The adjustment to the code stops the cube from being randomly positioned outside of the view of the camera. Having the cube stay in the frames of each camera prevents data values from being null.

### Task 3

To make the neural network model more capable of predicting the position of the cube, it was adjusted to make it less defined by reducing the Dense of the model.

A screenshot of a computer

Description automatically generated

Figure 4.2.3 Colabs Model adjusted to be less defined.

### Results

The added cameras added extra data to the csv file allowing for a more precise prediction of the cubes positioning. Redefining the model allowed for it to learn by itself to get a more accurate prediction.

## 4.3 Sprint 3

This sprint produced errors that halted the progression of the project. While trying to train the neural network on the new data from the three cameras added in the previous sprint, the neural network was producing measurement values that were too high and null values.

|  |  |  |
| --- | --- | --- |
| Task | Description | Status |
| 1 | Create new CSV file | Complete |
| 2 | Train neural network on data from three cameras | Incomplete |
| 3 | Run new data against old data | Incomplete |

### Task 1

Creating a new CSV file was created to contain the data from the three cameras. The new data has the position of the cubes X, Y, Z values from the perspectives of the cameras. This will give the neural network a more precise position to be able to have a more accurate prediction.

### Task 2

When passing data into the neural network and running the model, the outputs from the model showed that it was not learning to predict the positions of the cube. The values being outputted by the model were not like they were in previous sprints as the loss value was in the high 20s and the accuracy value was null.

A screenshot of a computer

Description automatically generated

Figure 4.3.1 – Neural network model outputs.

Task 3

As the model was producing unusable values, testing the new data against the model would have been pointless as the model needed to be fixed before running the new data against the old data.

Results

This sprint was incomplete as the error occurring with the neural network model halted the progression within the sprint. With the error occurring the neural network was unable to be trained and needed to be fixed.

## 4.4 Sprint 4

-Problems – Loss flatlined – had to tweak model to re-adjust- loss was in high 20s and accuracy was 0

- Fixed by adjusting script to normalise data before passing to neural net – takes away res

- Made cube move again within cameras view.

This sprint consisted of resolving errors that occurred when fitting the model with the new data given from the cube from the changes made in the previous sprint. This was due to the cube staying in the same place while rotation it was giving the neural network nothing to learn or predict from as the values where constantly repeating. Due to the lack of learning for the neural network, it was producing very high loss values on the data and having no accuracy. The objective of this sprint was to train the neural network on the data in relation to the different camera perspectives.

|  |  |  |
| --- | --- | --- |
| Task | Description | Status |
| 1 | Make cube move to random places withing view of cameras | Complete |
| 2 | Adjust cube script to normalize values | Complete |
| 3 | Train neural net on new normalized data | Complete |
| 5 |  | Complete |

Table 5 Sprint 2 Task Breakdown Table

### Task 1

The first task of this sprint was adjusting the cubes code to allow for it to spawn in random positions on the screen. The code was made to only allow the cube randomly to spawn in certain parameters so the cube will stay in view of all three cameras. This stops there being any null values in the CSV file.

A screen shot of a computer code

Description automatically generated

Figure 4.4.1

### Task 2

In this sprint, the code for the values displayed in the CSV file was changed to normalise the values to be in between 0 to 1. Having normalized data produces more comparable numbers as it changes the value from pixel point positions to a more normal number.

### Task 3

Once the errors were resolved, the new data was trained on the neural network with the normalized data. The loss and accuracy values produced better values then previously showed. The below images show the accuracy from the neural network when comparing the actual X, Y, Z values to the predicted X, Y, Z values.

A blue line with numbers and lines

Description automatically generated

Figure 4.4.2 - scatter graph of the actual X values and the predicted X values

A screen shot of a graph

Description automatically generated

Figure 4.4.3 - scatter graph of the actual Y values and the predicted Y values

A screen shot of a graph

Description automatically generated

Figure 4.4.4 - scatter graph of the actual Z values and the predicted Y values

### Results

Resolving the issues that occurred during the last sprint took up a lot of this sprints time but also allowed for new ways to approach the project and make small adjustments such as normalizing the values and restricting the cubes parameters to be spawned in. From the data given in this sprint, the neural network has been able to predict the X, Y, Z values very accurately as shown in the above images.

– Started first week of easter break. Add Dates and duration to each sprint. Going back two weeks for each

## 4.5 Sprint 5

The objective of this sprint was to use an unseen dataset to test the neural network. Using a test dataset will give a more accurate representation of how good the model is at predicting the position of the cube.

|  |  |  |
| --- | --- | --- |
| Task | Description | Status |
| 1 | Adjust script to produce a test dataset | Complete |
| 2 | Run the test data with the model | Complete |
| 3 | Compare the data of the training data with the test data | Complete |

Table Sprint 5 Task Breakdown Table

### Task 1

In order for the unity project to produce a different dataset the C# code was to be modified to print out a CSV file with a different name. To make sure that the original training data was not modified will the test data was being creating the code was modified by commenting out the training data variable and changing the variable being passed into the StreamWriter. In order to really test the model, the code was modified to all the cube to have a greater range of movement than previously had in the training data. By having unseen values being passed in will show the models true accuracy.

A computer screen shot of a program code

Description automatically generated

Fig – Modified C# code to produce test dataset.

A screen shot of a computer program

Description automatically generated A screen shot of a computer program

Description automatically generated

Fig – Training data values Fig – Test data values

### Task 2

Once the test data csv file was created by the Unity project, the data was passed into the neural net trained on the training data. The graphs below show each of the X, Y, Z values from predicting them from the test data.

A blue line with white background

Description automatically generated A blue line with white text

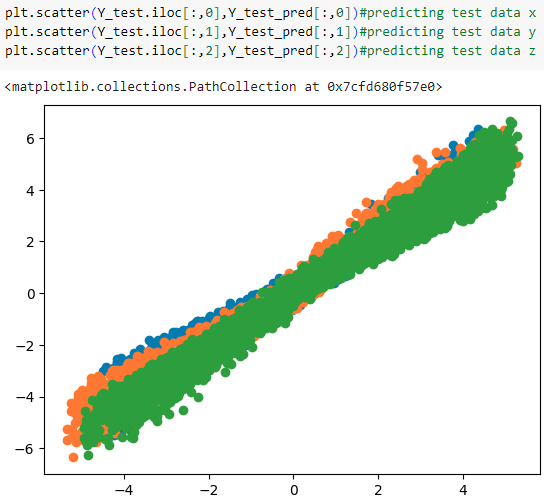
Description automatically generated with medium confidence A blue line with numbers and a graph

Description automatically generated with medium confidence

*Fig – Graph of predicted X values. Fig – Graph of predicted Y values. Fig – Graph of predicted Z values.*

### Task 3

In the graphs below you can see that even though the test data graph isn’t as straight as the training data, but it is very similar showing that the model is relatively accurate at predicting the position of the cube even with values that the model hasn’t seen before.

 A screen shot of a graph

Description automatically generated

*Fig – graph of X, Y, Z values of predicted test data Fig – graph of X, Y, Z values of predicted training data*

### Results

From the progress made in this sprint, resulted in having a successful sprint in testing the accuracy of the model with unseen data and values within the data. As seen in the graphs above it is shown that the model is accurate when it comes to predicting the 3D position of the cube from the 2D points.

## 4.6 Sprint 6

Following the success of the last sprint, this sprint was dedicated to training a model on the rotation values. In this sprint, there was a similar model to the position model created to predict the rotation values. Once the rotation model is trained the test data will be passed in to inspect the accuracy of the model with unseen data.

|  |  |  |
| --- | --- | --- |
| Task | Description | Status |
| 1 | Create a model to handle the rotation values | Complete |
| 2 | Modify code to print more dependable rotation values | Complete |
| 3 | Run training data with the rotation model | Complete |
| 4 | Run the test data with the rotation model | Complete |
| 5 | Compare the data of the training data with the test data | Complete |

### Task 1

To create the rotation model, it was the same kind of model as the position model. As the rotation values are more unpredictable values the loss and accuracy of the model won’t be as good as the position model.

### Task 2

After running the data on the rotation model there was a concern on the out put of the model based on the type of angles being processed by the C# script. The script was the modified to change the angle output from Euler Angles to Quaternion angles. The reason for changing to Quaternion Angles is due to them not suffering from gimbal lock as it makes them more stable for interpolating rotations and performing complex rotations. They also produce more precise rotation calculations which is needed for the neural network to gain as much knowledge as possible. For the model to accept the Quaternion values, it had to be modified to produce 4 values instead of 3.

A screen shot of a computer code

Description automatically generated

*Fig – rotation neural network model*

### Task 3

Once the code was modified, the training data was then passed through the model. As expected, the loss and accuracy of the model isn’t as high as the position model as the rotation values are a lot more difficult to predict.

A screen shot of a graph

Description automatically generated

*Fig – Graph of predicted rotation values based on training data.*

### Task 4

The next task in this sprint was to run the model with the test data to see the accuracy of the model with unseen data.

A screen shot of a graph

Description automatically generated

*Fig – graph of predicted rotation values based on test data.*

### Task 5

From the graph below, it shows that the prediction accuracy of the model is consistent as both training and test data are very similar to each other as shown below.

A screen shot of a graph

Description automatically generated *Fig – Comparison Graph of training and test data*

### Result

As the model produced a similar graph with the test date to the training data as shown above, the results of this sprint were a success in training and testing the model to predict the rotation of the cube.

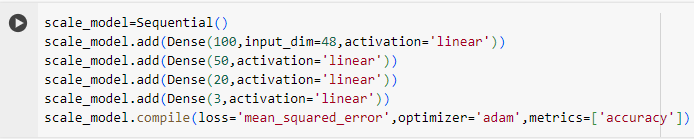
## 4.7 Sprint 7

The objective of this sprint was to train and test a neural network in predicting the scale values of the cube. The model was tested in a similar way to the position test by modifying the values of the C# script to introduce values that the model has never seen before and see how accurate it predicts the values.

|  |  |  |
| --- | --- | --- |
| Task | Description | Status |
| 1 | Create a model to train with the scaling values | Complete |
| 2 | Run training data with the rotation model | Complete |
| 3 | Modify the C# script with unseen values for testing | Complete |
| 4 | Run the test data with the rotation model | Complete |
| 5 | Compare the data of the training data with the test data | Complete |

### Task 1

The model created to be used with the training data was the same layout as the position model. As the scaling values would have been similar to the position values. The accuracy of the model was expected to be high. Below is the neural network created.



*Fig – scaling neural network model.*

### Task 2

Once the model was created the training data was passed through to train the model to predict the scaling values. The accuracy was high, and the loss was low. From the graphs below it shows the accuracy of the model predicting values based on the training data.

A graph of blue dots

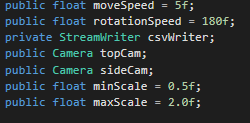
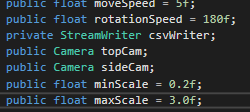
Description automatically generated A screen shot of a graph

Description automatically generated A screen shot of a graph

Description automatically generated*Fig – Graph of predicted X values Fig – Graph of predicted Y values Fig – Graph of predicted Z values*

### Task 3

Once the model has been trained, the code was needed to be modified in order to generate values in the test dataset that the model hasn’t seen before. In order to do that, the C# code was altered to increase the maxScale variable and the decrease the minScale in order to give the cube more of a range to scale up and down to.

*Fig – C# training value code Fig – C# testing values code*

### Task 4

After the code was modified and the test dataset was created, the data was then run to test the accuracy of the model. The graphs produced by the test data wasn’t as accurate as the training data but produced a similar but more disburse graph as seen below.

A screen shot of a graph

Description automatically generated *Fig – Graph of predicted test X values*

A screen shot of a graph

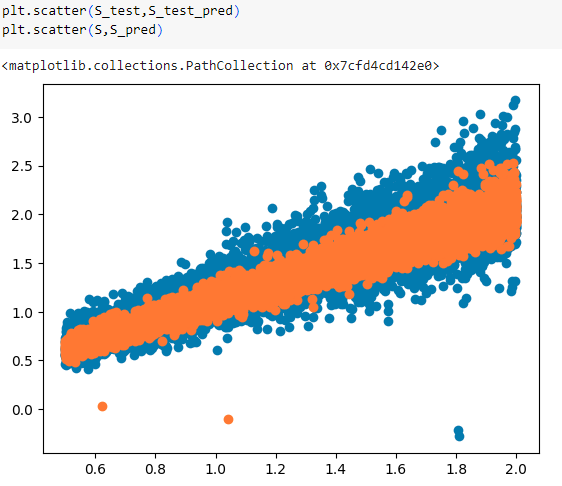
Description automatically generated *Fig – Graph of predicted test Y values*

A screen shot of a graph

Description automatically generated *Fig – Graph of predicted test Z values*

### Results

The results from this sprint were a model that was trained and tested to accurately predict the scaling values of the cube. The model was relatively accurate between the testing and training predictions as seen in the graph below. The predictions on the test data was expected to be more disburse as there was values the model had not seen before and had to train itself when seeing them.



*Fig – Comparison of training data prediction to test data prediction.*

## Abbreviations

AI - Artificial intelligence

GCP Ground control points

ANN - Artificial Neural Networks

SNN- Simulated Neural Networks

# References

Coursera. (2023). What Is Machine Learning? Definition, Types, and Examples. *Coursera*, 1.

Ganesh, K. S. (2020, July 24). *What’s The Role Of Weights And Bias In a Neural Network?* Retrieved from towardsdatascience: https://towardsdatascience.com/whats-the-role-of-weights-and-bias-in-a-neural-network-4cf7e9888a0f

Goodman, D. (2023, September 15). *All the Science*. Retrieved from All the Science: https://www.allthescience.org/what-is-aerial-photogrammetry.htm?expand\_article=1

IBM. (2023). *IBM*. Retrieved from IBM.com: https://www.ibm.com/topics/neural-networks

James S.Aber, I. M. (2010). *Small-Format Aerial Photography.*

Marketing, T. (2023). *TOPS Take Off Professionals*. Retrieved from What is Photogrammetry: https://www.takeoffpros.com/2020/06/16/what-is-photogrammetry/

McCullum, N. (2020, June 28). *Deep Learning Neural Networks Explained in Plain English*. Retrieved from freecodecamp.org: https://www.freecodecamp.org/news/deep-learning-neural-networks-explained-in-plain-english/#:~:text=Neurons%20in%20deep%20learning%20models,layer%20of%20the%20neural%20net.

Russell, S. N. (2021). *Artificial Intelligence: A Modern Approach.* Pearson.

S.Gillis, A. (2023, May). *TechTarget*. Retrieved from TechTarget.com: https://www.techtarget.com/searchenterpriseai/definition/unsupervised-learning#:~:text=Unsupervised%20learning%20is%20a%20type,are%20neither%20classified%20nor%20labeled.

Simplilearn. (2023, Feb 17). *Simplilearn*. Retrieved from Simplileran.com: https://www.simplilearn.com/tutorials/machine-learning-tutorial/reinforcement-learning#what\_is\_reinforcement\_learning