School of Information Technology and
Engineering at the ADA University

School of Engineering and Applied Science
at the George Washington University

CNN-BASED STAR TRACKER FOR HIGH PRECISION SPACECRAFT NAVIGATION

A Thesis
Presented to the Graduate Program of Computer Science and Data Analytics
of the School of Information Technology and Engineering
ADA University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Computer Science and Data Analytics
ADA University

By
Farid Guliyev

April, 2024

THESIS ACCEPTANCE

This Thesis by Farid Guliyev
Entitled: *CNN-based Star Tracker For High Precision Spacecraft Navigation*

has been approved as meeting the requirement for the Degree of Master of Science in Computer Science and Data Analytics of the School of Information Technology and Engineering, ADA University.

Approved:

(Adviser)	(Date)
(Program Director)	(Date)
(Dean)	(Date)

ABSTRACT

There are several sensors to detect the attitude determination of the spacecraft. One of the most accurate is the star tracker which is optical equipment that measures the positions of stars. However traditional star identification algorithms are limited by their processing speed and may fail under harsh conditions which can be critical for the spacecraft. Furthermore a large and complex database of star trackers can cause inefficiencies throughout the processing process. An improved star tracker algorithm is proposed to address the issues above. This algorithm will be able to handle these challenging situations and give the most accurate estimation of the spacecraft orientation using state-of-the-art detection.

This paper proposes an improved star detection algorithm for spacecraft attitude detection using convolutional neural networks (CNN), a type of deep learning algorithm that has shown many promising results in various image processing applications. CNN-based star detection algorithm will be able to identify the stars based on the constellations and show the spacecraft's orientation using a less complex database. This will positively affect the accuracy and efficiency of the processing and increase robustness against challenging conditions. Using CNN-based algorithms creates an opportunity continuously improve spacecraft attitude control systems.

Free, open-source databases are being used to train the algorithm to overcome the issue of inaccessible star and constellation images from satellites. Publicly available data from planetariums such as Stellarium and space simulator platforms such as Celestia are being used for training, evaluation, and performance analysis of the proposed algorithm. Moreover various optimization techniques are implemented to improve the performance of the spacecraft attitude determination algorithm. It is believed that this approach will surpass the performance of the traditional algorithm and will lead to a promising direction for the development of advanced star tracker systems. Further research can focus on applying this approach to other areas of spacecraft control such as orbit determination and maneuver planning.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Explanation
AOCS	Attitude and Orbit Control System
CMOS	Complementary Metal-Oxide-Semiconductor
SPR	Star pattern recognition
CNN	Convolutional Neural Network
NLN	Neural Logic Networks
ESA	European Space Agency
RA	Right Ascension
DEC	Declination
GUI	Graphical User Interface
VGG	Visual Geometry Group
ResNet-50	Residual Network 50
CSV	Comma-separated values
GAP	Global Average Pooling
ReLu	Rectified Linear Unit
ADAM	Adaptive Moment Estimation
MSE	Mean Squared Error
RMSE	Root Mean Squared Error
MAE	Mean Absolute Error
HIL	Hardware-in-the-loop
ViT	Vision Transformer

1. INTRODUCTION

Background and Fundamentals

Spacecraft control is one of the most important field in space missions. Its importance comes from the role played by the Attitude and Orbit Control System (AOCS) subsystem. This subsystem has big impact on orientation of the spacecraft and its motion in space. The consequences of errors in AOCS functionality are extreme and can be way more than just operational inconveniences. Even the success of the entire spacecraft mission can potentially be put at risk. As an example an AOCS failure can result is the ability of the spacecraft to control its solar panels which result to limiting the power supply and overall energy control. Another example is that if AOCS does not simplify precise control then alignment of communication antennas with the stations based on Earth could experience delay. This would disrupt important data transmission and reception. Due to all mentioned reasons, the importance of proper design and testing of the AOCS subsystems is crucial. It will test their reliability and how flexible the subsystem is against harsh space conditions.

Various sensors are utilized to achieve high precision control within the AOCS. Each of them carrying a specific function in spacecraft orientation and navigation. The following sensors are: coarse sun, earth, gyroscopes, star trackers, and fine sun sensors [1]. Earth sensors provide basic accuracy in identifying spacecraft orientation in relation to surface of the Earth. They do it with basic complexity in their execution to accommodate varities in atmospheric conditions and orbital position of Earth. Coarse sun sensors provide high accuracy in identifying spacecraft orientation related to the sun which makes them essential for precise solar panel alignment. They usually involve basic complexity during execution when calibration is required and when considering solar radiation change. Gyroscopes have high accuracy when measuring angular velocity, this is crucial for stable spacecraft attitude control. But they often involve high complexity because of calibration and error correction requirements. Fine sun sensors have high accuracy. They help to have accurate solar attitude control. Also, they have high execution complexity because of their sensitivity and calibration requirements. Star trackers provide very high accuracy when calculating spacecraft orientation. Because they accurately find out star positions while having high complexity when executing and processing. The table below illustrates a general overview of the different accuracies and complexities of each of the sensor types:

Table 1.1: Essential sensors of the AOCS

Sensor Type	Function	Accuracy	Complexity
Earth Sensors	Measure roll and pitch	Moderate	Moderate
Coarse Sun Sensors	Measure yaw	High	Moderate
Gyroscopes	Measure angular rates	High	High
Fine Sun Sensors	Measure the Sun’s position	Very High	Very High
Star Trackers	Determine spacecraft attitude	Very High	Very High

As you can see from the table demonstrates the sensors of the AOCS, both the Star Tracker and Fine Sun sensors have the highest accuracy which makes them essential for missions that requires critical high precision spacecraft navigation. Despite the fine sun sensor slightly lower accuracy compared to the star tracker, in most cases both sensors play critical roles ensuring precise spacecraft orientation and control. However for deep space missions

where the sun visibility is limited, the usage of sun sensors becomes impractical, that is why the importance of the Star tracker remains suitable for many space missions.

Star trackers are optical instruments that measure the orientation of spacecraft by detecting the position of stars in the sky. Here is the general working mechanism, in other words, the block diagram of the star tracker [2].

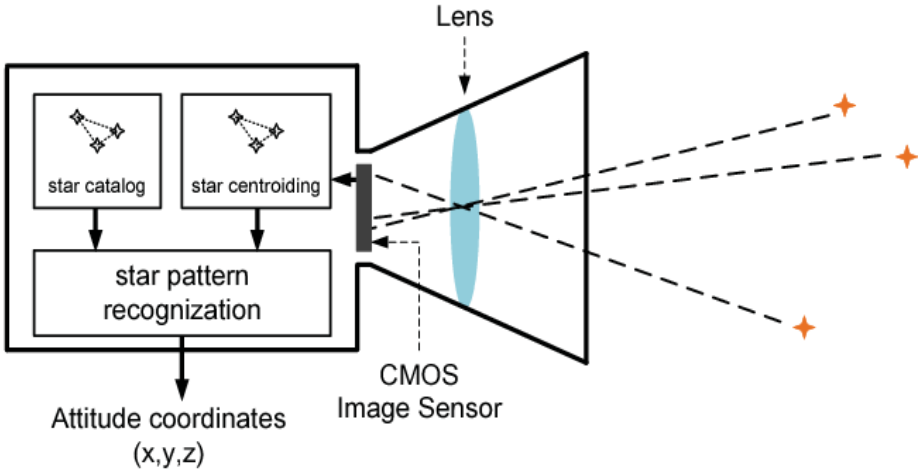

Figure 1.1: Simple block diagram of the star tracker

As you can see from the simple block diagram the star tracker first gets the data using the Complementary Metal-Oxide-Semiconductor (CMOS) Image Sensor. Then it passes the observed stars to the control unit. The star tracker algorithm of the control unit consists of three parts: star catalog, star centroiding and star pattern recognition.

A star catalog is an extensive database encompassing vital information about each star. Typically these catalogs are organized by the magnitude of the stars as the brightest stars are more readily detectable by a CMOS image sensor. This method of organization allows for efficient identification of celestial bodies based on their apparent luminosity. The following table presents the brightest stars in the example star catalog.

Table 1.2: Example of the five brightest stars within the star catalog

Star Name	Constellation	Magnitude	Angular Distance
Sirius	Canis Major	-1.46	8.6 light-years
Canopus	Carina	-0.74	310 light-years
Arcturus	Boötes	-0.05	37 light-years
Vega	Lyra	0.03	25 light-years
Betelgeuse	Orion	0.42	642 light-years

On the other hand the second part of the star tracker which known as star centroiding, determines how the spacecraft is positioned. When the star catalog and star centroiding are put together, they form star pattern recognition (SPR). This system accurately identifies where the spacecraft is located concerning the stars. A more detailed representation of the SPR system of the star tracker can be visualized in the figure below which demonstrates the relationship between the star catalog and general algorithm and how star image is processed through the system.

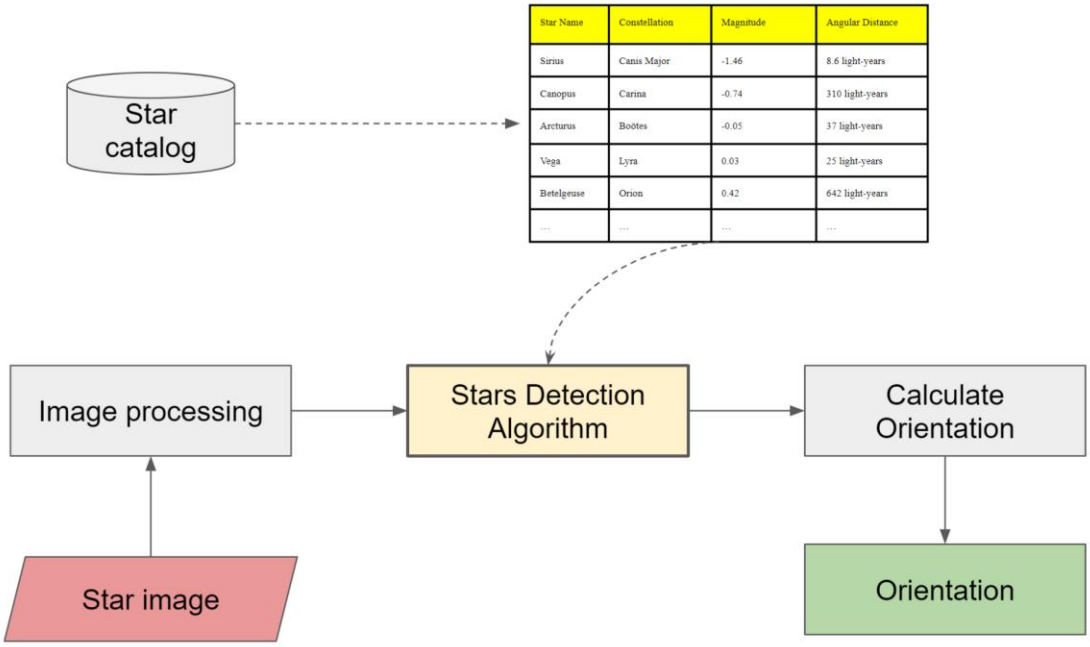

Figure 1.2: Representation of the SPR

Traditional star trackers use hand-crafted algorithms to match observed stars with a star catalog and calculate the spacecraft orientation using star-centroiding approach [3]. These algorithms operates based on predefined rules and heuristics that guide the exploration process which make them typically deterministic meaning that they follow a specific sequence of steps to examine possible solutions systematically. That is why their accuracy capabilities range from a few arc seconds to tens of arc seconds with a unit of angular measurement with one arc-second equaling 1/3600th of a degree. This level of precision allows star trackers to reliably determine spacecraft orientation with exceptional accuracy.

The drawback of traditional star trackers

Despite the exceptional accuracy these algorithms have drawbacks as well. The first drawback is that they are very computationally expensive. Therefore high-performance and very expensive embedded systems needed to overcome this issue. The reason for this drawback is due to a huge star catalogue that contains information about all the stars and their positions. Approaching it in a simple manner where each star that is observed needs to be compared with the stars in a star catalogue. Complexity of time is defined as *O (nm)* - n is the number of stars observed; m is the number of stars in the catalogue (involving a nested loop iterating over each observed star to compare to the ones in the catalogue). The modern star tracker algorithms use indexing or ranked data structures which can come up with a time complexity of *O(nlog(m))*. This is way more optimal than the simple approach. However considering that m is a large number it still takes up too much memory for embedded devices.

The second drawback is the how reliable are the star tracker algorithms. The fundamental features of star tracker operation: it identifies and extracts the brightest stars from captured images; finds their centroids; sorts them and uses their arrangement as a reference coordinate system to calculate the position of the spacecraft. However the presence of false stars can significantly affect the accuracy of this process. This basically leads to incorrect position outputs and can critically affect the mission of the spacecraft.

Proposed approach

This thesis introduces a different method for designing star trackers by applying and using Convolutional Neural Networks (CNN). Instead of relying on a search of the star catalogue, the algorithm can analyses the whole image of observed stars and calculate the positions of the altitude based on features instructed in advance. This will not require the use of the star catalogue.

CNN is an artificial neural network. This network can learn to recognize complex patterns in image data. These neural networks have transformed the way machines recognize and understand visual information. This allows them to learn patterns and features from inside the image data with remarkable precision while being effective [4].

One of the main features that differentiates CNN from other networks is how it uses convolutional layers. These layers put in applicable convolution operations to insert images. All that while still effectively drawing out features like: edges, textures and shapes (from filters). Because it convolves these filters across the input image CNN can capture spatial relationships and local patterns. It does that by moderately drawing an image of the visual features.

Also CNN uses pooling layers to narrow down the feature maps that are produced by convolutional layers. They may help to minimize the large scale of the feature maps. Whilst they still maintain the most important information. This process enhances computational efficiency and robustness to spatial variations that enable CNN to capture invariant features across different regions of an image effectively.

Using CNN on a large dataset of star images can make it possible to create a star tracker model. That star tracker model can rapidly and precisely detect stars in the sky and calculate the direction of the spacecraft with higher precision and lower computational costs. CNN models use the most obvious and basic approach for locating the stars. The networks set probability of occurrence for every possible match and select the one with the highest one. Different factors like noise, false stars, or dynamic situations can have an effect on this process. Hence it is important to carefully build and instruct CNN algorithms to achieve good star detection model. This strategy also shows lowered computing complexity and memory requirements compared to the old method, albeit at the expense of decreased precision.

Recently there has been an increase in interest in CNN-based star trackers as a topic for research. Numerous studies have shown its success in spacecraft navigation. Nevertheless there is an ongoing requirement to create star trackers based on Convolutional Neural Networks (CNN) that can attain exceptional precision while remaining resilient to diverse operational circumstances including alterations in lighting settings, image noise, and camera distortions.

2. REVIEW OF THE LITERATURE

Since the late 20th century many research papers have explored many aspects of the star tracker to enhance its accuracy and make it optimal for spacecraft applications. These studies include the development of the algorithm for star pattern recognition and making it robust for challenging conditions.

Evaluation of the star tracker algorithm from the research mainly considers the three criteria. These criteria are the effectiveness of the feature extraction step, efficiency of the database search and utilization of independent pattern features. They serve the benchmarks to test the performance of the algorithms of the star tracker for successful space missions.

One of the innovative contributions to the star trackers' algorithm is attributed to Liebe's work. This research was presented in the paper titled "Pattern Recognition of Star Constellations for Spacecraft Applications" [6]. He proposed that the angular distance to the first and second stars and along angle between them is more than enough to create the unique star identification. He claims that by using these parameters, star trackers could achieve more accurate and reliable star pattern recognition. The approach is demonstrated in the figure below.

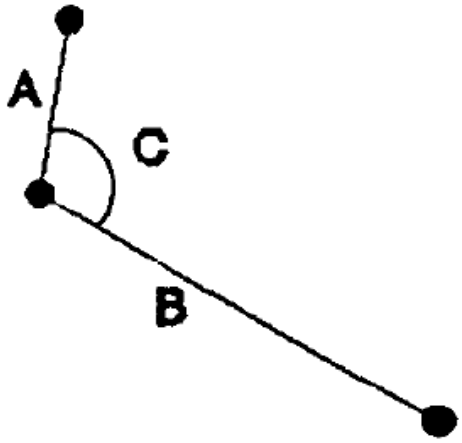

Figure 2.1: The triangular approach by Liebe

In this figure, **A** is the angular distance to the first neighboring star, **B** is the angular distance to the second neighboring star and **C** is the angle between the first and second neighboring star. Carl Christian Liebe claims that the star catalog should be arranged using these three values. He believes that these values are more than enough to detect the stars and determine their position carefully. The analysis presented in the paper demonstrated the developed method's capability to evaluate satellite attitudes with remarkable accuracy which surpass the 15 arcseconds. Additionally by employing averages this method could even achieve accuracies better than 2 arcseconds.

Another pioneering paper in the realm of star trackers by Jian Hong and Julie A. Dickerson introduced an autonomous star identification system that uses fuzzy neural logic networks (NLN) [7]. Their findings indicated that this system can achieve a very good balance between high accuracy and swift recognition speed compared to direct-match algorithms even for large star catalogs. Moreover, the NLN exhibited the capacity to adapt to new patterns by adding additional cluster nodes to the network when novel celestial patterns were introduced to the catalog.

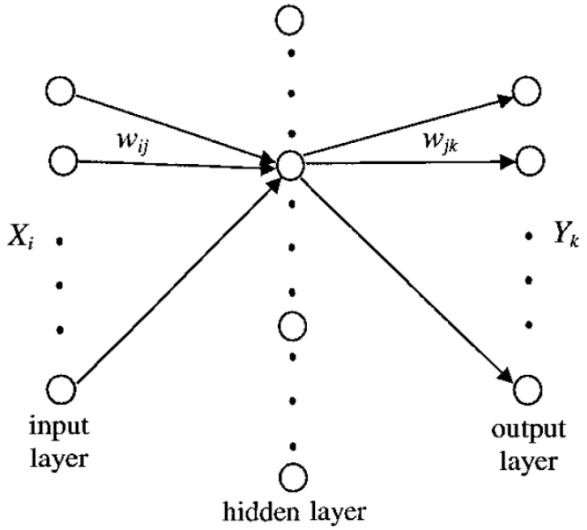

Figure 2.2: Hong's Neutral Network

Further noteworthy contribution in the domain of star trackers is a paper written by M. D. Pham, K. S. Low, Chen Shoushun and Y. T. Xing [8]. This innovative approach effectively addresses the challenge of searching extensive star catalogs by using the tree scheme. The results from their experiments demonstrated considerable performance advantage over the Liebe and Geometric voting methods in terms of execution speed, accuracy and robustness. The most significant achievement of this approach was a remarkable 50 percent reduction in runtime compared to conventional methods. The example of the search tree structure presented in this paper is visualized in the figure below.

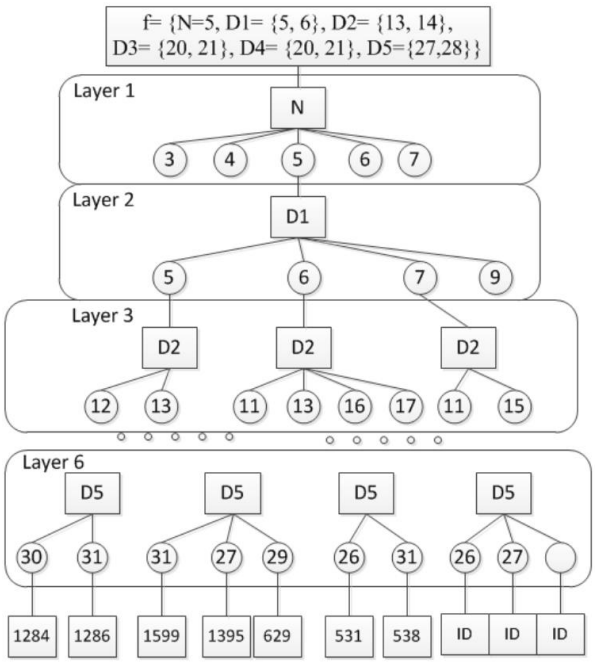

Figure 2.3: Search tree structure

3. DATA COLLECTION

Challenges

Numerous open-source datasets are utilized as valuable resources for development of star tracker algorithms. The release of Gaia Data offered by the European Space Agency (ESA) is a notable example. Gaia is a mission created for space observation. Its purpose is to map Milky Way galaxy in three dimensions. It offers a wide catalogue of stars with their positions, distances and proper motions. These can be used for astronomical purposes like star tracking for spacecraft navigation [9].

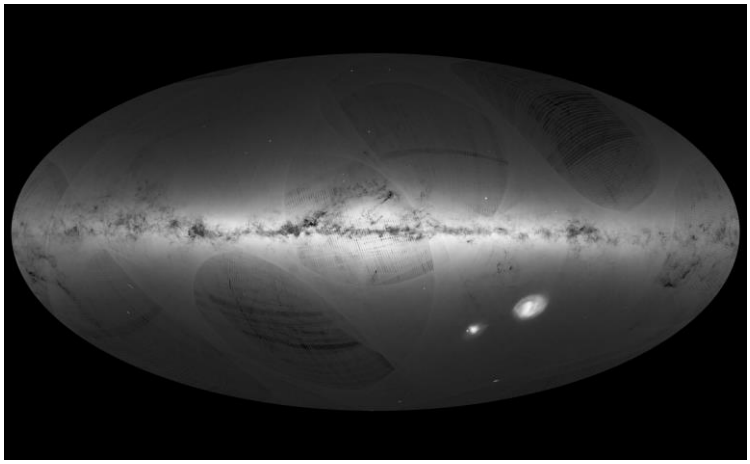

Figure 3.1: An all-sky view of stars by Gaia

While various officially open-source datasets are available for stars that provide information on their brightness and distance like Gaia, these datasets were not suitable for this study. Because the primary purpose in this study is to get the position directly from the image without referring to the star catalog. Therefore the process of data collection presented a significant challenge in this work. Specifically, this paper aims to train a model using sky images that include all of the aspects of the sky. It was necessary to collect the required dataset independently to address this issue.

Hence only two options were available for dataset collection. The first option was web scraping to extract sky images taken by telescopes and people. This approach was ineffective because of considerable difference between the images like different times the image was captured, focal length and visibility of star brightness. As a result, there are not enough datasets that fully and comprehensively capture every region of the sky. That is why this approach is considered useless for gathering data.

The second option involved using simulation programs to generate images for model instruction. This method has its own advantages. As it increases the similarity among simulation and real-world images and also affects flexibility to operate dimensional environments. Nevertheless it still has certain disadvantages. Specifically simulated images tend to be idealized compared to those obtained from star trackers which may be affected by environmental errors. Nonetheless the ability to manipulate simulation environments enables mitigation of such discrepancies. Consequently the decision was made to collect images and data from simulated environments for the data collection process.

Selection of Simulation Software

Therefore several software programs were researched to find the optimal solution for generating and creating images of stars' constellations that precisely estimate real ones. Of the available options, Stellarium emerged as the most suitable candidate for this task, given its open-source nature and compatibility with the GNU General Public License version 2.

Stellarium is reviewed as the best software option for dataset collection. The software offers accessibility, accuracy and versatility. Stellarium has user-friendly interface and large database of celestial objects. This software creates perfect tool for night sky observation. It has high accuracy level in showing the locations and movements of celestial bodies and provides reliable observations and simulations. This gives its users captivating and realistic astronomical experience. It also offers a powerful local programming language that allows users to manipulate the state of the universe, including the positions and appearances of celestial objects and collect data about them. The programming language used by Stellarium is applied to utilize JavaScript and can be accessed through Scripting Console in Stellarium graphical user interface [10].

Users of this software can write scripts to automate certain tasks. These tasks include: taking screenshots, creating animations, and exporting data. The scripting language used also allows to create custom plugins that can extend the functionality of Stellarium software.

Stellarium's scripting capabilities give flexible and customizable platform for generating synthetic data for machine learning and computer vision applications. The ability to control the state of the universe through programming makes it possible to generate large datasets of realistic sky images as well as using available ground truth information. That is why this planetarium can be used to instruct, program and test machine learning models for tasks like recognition of sky images.

Figure 3.2: An all-sky view by Stellarium GUI

Primary Objective in Script

The main purpose of script coding was to get images capturing different perspectives of the night sky with corresponding images of constellations and the overall position and location of every image within the celestial sphere (indicated by global coordinates). Using this method the sky images represented independent values. And constellation images and their respective locations represented dependent values or desired outputs for CNN model. Right Ascension (RA), Declination (DEC), and Theta were used to define the positions of the sky images.

Right Ascension (RA) and Declination (DEC) are celestial coordinates that are used to pinpoint positions of celestial objects inside the celestial sphere. RA measures the angular distance of an object eastward along the celestial equator from the vernal equinox (generally expressed in hours/ minutes/seconds/ degrees). DEC defines angular distance of an object north or south of the celestial equator (similar to latitude on Earth, usually expressed in terms of degrees/minutes/seconds) [11]. The following coordinates as well as Theta (represents angular position of simulated view in a celestial observation program) give a extensive and understandable structure for accurate definition of locations for images of sky.

Utilizing RA, DEC, and Theta is advantageous for position-defining from sky images due to their global nature and adherence to standard astronomical conventions. The following coordinates give a consistent and universally accepted structure for definition of location for celestial object. This basically ensures compatibility and collusion across various astronomical datasets and software platforms. Also RA and DEC ease for the precise localization of celestial objects independant of location of observer or time. This feature makes them valuable tools for machine learning and astronomical research. Hence incorporating RA, DEC and Theta in the script coding process makes sure for a robust and reliable identification for locations of sky images. This is essential for instructing CNN models to detect sky locations and positions.

Script Coding

The script arranges operations to simulate celestial observations and create images. It starts by setting up the environment then enclosing time settings using the factors like: Julian date; atmospheric conditions and landscape setting. Following that it sets the viewport to replicate and recreate star tracker's perspective. Main core functions handle the conversion of angular coordinates to radians. They assess star visibility in the observer's field of view. And they calculate the fraction of visible stars inside constellations.

The task is accomplished by first defining a transformation function called **"transform"**. The following **"transform"** function uses three parameters: Right Ascension (RA), Declination (DEC) and Theta. Then the mentioned parameters are scaled by a constant factor in order to convert them into radians. Inside this given function angular separation and azimuth angle between a given star and the observer's position in the sky are calculated.

Following this function named **"isVisible"** is used to identify if a star is visible in the observer's field of view. This function uses the transformation function and coordinates of the star as its given input and returns value that shows the visibility of the star.

The location of observer and observer's field of view are set by using the **"setView"** function in the simulation. This given function adjusts and sets the parameters by defining them as: Right Ascension, Declination Theta and Field of View (FOV). It does that in order to simulate different observational scenarios.

The **"getLabel"** function is responsible for calculating and computing visibility fraction for every constellation in the sky. It applies **"transform"** function to transform the observer's position. And it uses iteration for each constellation to calculate its visibility fraction.

Finally main loop is used to create simulated sky images. Inside this loop random values for Right Ascension, Declination, Theta and FOV are picked. The **"setView"** function is then used to create and build the simulation. And visibility fractions of constellations are calculated using the **"getLabel"** function. The following fractions as well as index of the image are the output. This loop continues for a specified number of iterations while it captures screenshots of simulated sky images (with and without constellations). And it saves global coordinates to the CVS file.

After the script the completion of the script it is uploaded to the Stellarium GUI. The value of the **"ImageCount"** variable is set to 20000 for the number of images. The script is

then run on Asus ROG Strix laptop. This laptop was chosen because of its i9-12th generation processors and NVidia 3070 Ti GPU. The following are considered sufficient for data collection and model testing.

The image acquisition process was allowed to run for a total of seven whole days. In addition to the acquisition process computer's resources that included its CPU and GPU frequency, speed and other parameters were all monitored to ensure and avoid any issue occurance. The image acquisition process and resource monitoring are shown and illustrated in the figure below.

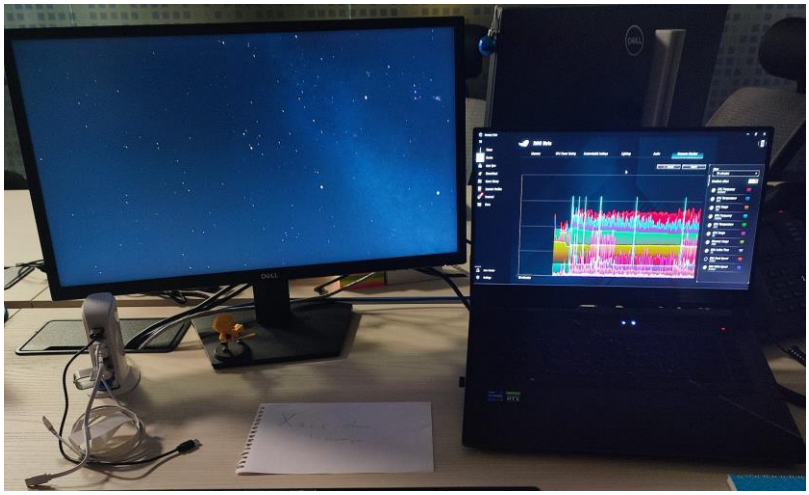

Figure 3.3: Image acquisition and resource monitoring

Generated Images and Sky Positions

As the program has been terminated, by the end of it it has captured a total of 20,000 images for each of the input images and corresponding constellation images and celestial locations. This number is believed to be sufficient and enough for creating and generating a model applicable for testing its ability to predict and calculate the locations in sky images. The figures below illustrate an example of pairing of input and output constellation images.

Figure 3.4: One of the input image

Figure 3.4: Corresponding output image for the above input image

As can be seen from Figure 3.3, one of the input images displays key celestial objects including Rigel, Betelgeuse, Aldebaran and Sirius with others. Additionally global constellations such as Orion and Taurus, and more, formed by these stars, are noticeable in the figure depicted above as well which indicates the correctness of the images from the simulation software.

The table below presents global coordinates using RA, DEC, and Theta corresponding to the initial five input sky images captured during the data collection.

Table 3.1: Corresponding global positions for the first 5 input images

ID	Input Image	RA	DEC	Theta
1		-104.91	-39.93	273.40

2		55.27	4.52	229.25
3		-131.71	-6.72	60.13
4		65.04	18.73	161.01
5		64.47	-4.24	289.98

4. METHODOLOGY

Data Analytics

Data analytics methods are important for analyzing large datasets and extracting and separating significant parts and insights from those datasets. Because of large number of images collected manual inspection of each image is not practical. That is why data analytics techniques can aid in processing images efficiently and automatically detecting constellations among them.

In addition data analytics techniques can aid in processing data before training of convolutional neural network model. Normalization, scaling and feature extraction can be examples for data analytics. These techniques can improve accuracy of the model by reducing noise and enhancing the relevant image features. To summarize the data application for data analytics strategies can aid in improving the quality, efficiency and precision of the data and model [12].

Various data analytics techniques can be applied to the sky images to extract meaningful insights and train a CNN model for detecting constellations. One of the initial steps is to convert the images into an appropriate format for analysis as part of the data preprocessing. This can involve techniques such as resizing the images, converting them to grayscale, or applying filters to get certain features. For the given models all the images were resized to 512x512x3. The following dimensions were specifically chosen because they balance memory efficiency and preserve the details of the image.

Following the processing feature extraction techniques can be used to find relevant patterns and features from images. This also involves determining and calculating specific features of images. These features could include brightness and orientation of stars. These can be used to differentiate between input and target images. The following process can include different algorithms like edge detection, blob detection as well as template matching.

In the figure below, an example of a data analytics technique for the models that was applied to the image is shown. In this technique, pixels below a particular threshold value of 20 were removed from the image and made white to enhance the visibility of the bright stars and constellations. The reason why that technique is also implemented in the target image in the figure is that it is just for demonstration purposes. While training the model, that technique is only used for input images, not target images. The threshold value of the method was chosen after multiple attempts to find the most suitable threshold value.

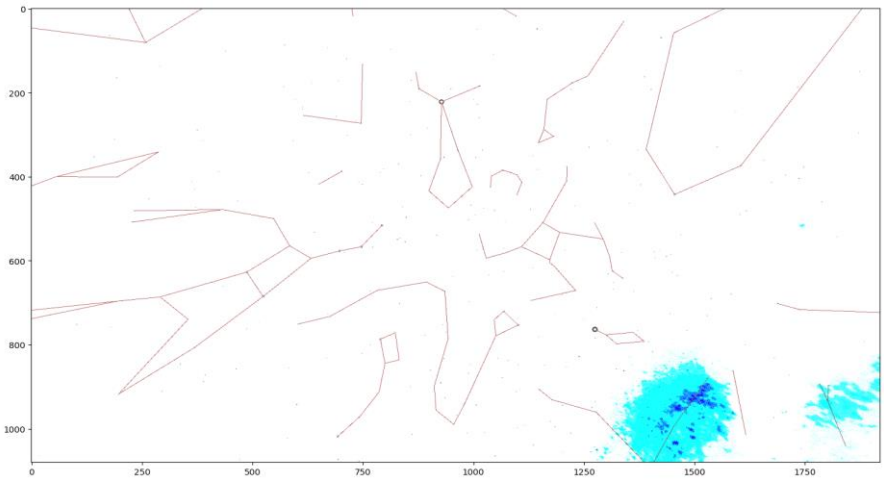

Figure 4.1: Data technique for removing redundant pixels

The blue sections on the left side of the figure depict the Milky Way. The Milky Way was not removed from the environmental setup of the Stellarium script because it can assist in detecting the constellations and positions of stars. For instance, the tail of the Cygnus northern constellation is always on the Milky Way. That is why the Milky Way can be used to determine the position of the Cygnus constellation [13]. This method also applies to many other constellations. The following technique is not only utilized for planetarium applications. It is also used by astronomers in real world. Even though Milky Way is thought to be significant in helping the CNN model analyze and detect constellations and star locations. Given below is the pseudo-code that illustrates implementation of the following technique in the image-loading function.

ALGORITHM 1: Function to load and preprocess the images

```
FUNCTION load_images(image_name):
    img = READ_IMAGE(image_name)
    FOR EACH pixel IN img:
        IF pixel < THRESHOLD_CLEANSING THEN
            pixel = 255
        END IF
    END FOR
    img = RESIZE_IMAGE(img, image_shape)
    RETURN img
END FUNCTION
```

Furthermore the diagram below shows general steps in preprocessing image and instructing CNN model. The following steps start with an input of a sky image then the image is being resized to 512x512x3 dimensions. Later data preprocessing techniques are utilized to eliminate and reduce unnecessary pixels. Depending on the specific application of the model (any of the U-Net, Resnet, or ViT) final output predict either orientation or segmented version of the image itself.

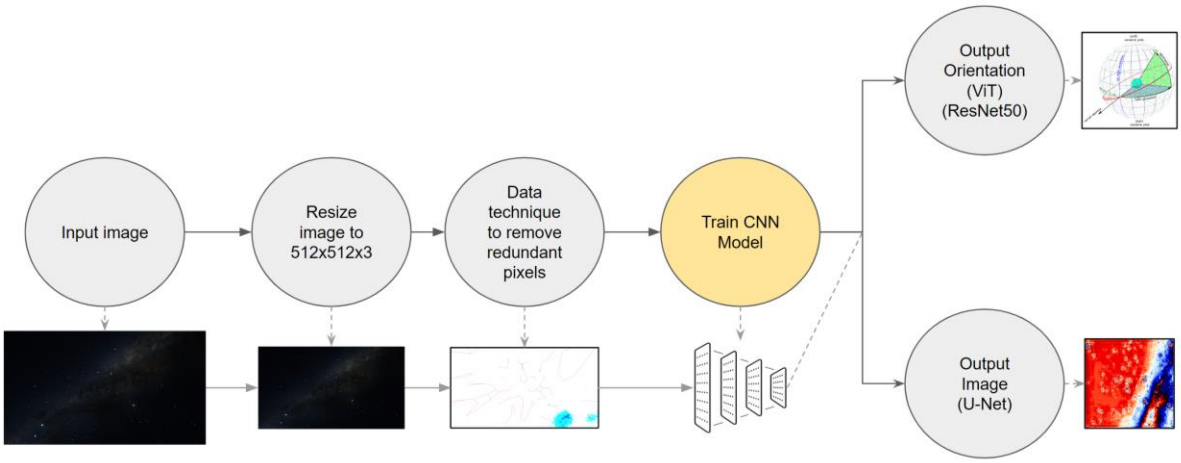

Figure 4.2: Diagram of CNN Model Preprocessing and Training

Developing CNN architecture

To achieve the objective of developing a CNN-based star tracker for high-precision spacecraft navigation three distinct architectures were trained and evaluated. The first architecture U-net operates by taking captured image as input and generating an output image

with constellations of stars. The second approach includes Vision Transformer. This transformer directly takes out the important positional information. For instance Right Ascension (RA), Declination (DEC), and Theta taken from input image. Lastly third approach uses ResNet-50 model. This model inputs captured image and outputs positional coordinates with RA, DEC, and Theta too.

Through careful testing and evaluation of these three models the following thesis aims to analyze and find the most effective approach for CNN-based star tracking based on high-precision spacecraft navigation. In addition this research examines into analyzing the strengths and weaknesses for each model. As well as exploring each model's potential applications and also proposing directions for future research in spacecraft navigation and CNN-based star tracking.

U-Net

Initial strategy is used to implement the task involved image segmentation. The base technique in computer vision is used to split up digital images into specific groups of pixels. The meaning behind using this approach is for constellations to be accurately detected in sky images. Using image segmentation the definition of names and positions of stars would be considerably simplified. This strategy is based on the fact that celestial sphere is divided into 88 recognized international constellations each of them having a unique configuration of stars. Therefore when a specific constellation is successfully defined (e.g. "Cassiopeia") it becomes practical to detect and label its corresponding stars such as Segin, Ruchbah, and Caph. Here is a visualization of the Cassiopeia constellation and its neighboring constellations highlighting the brightest stars [14].

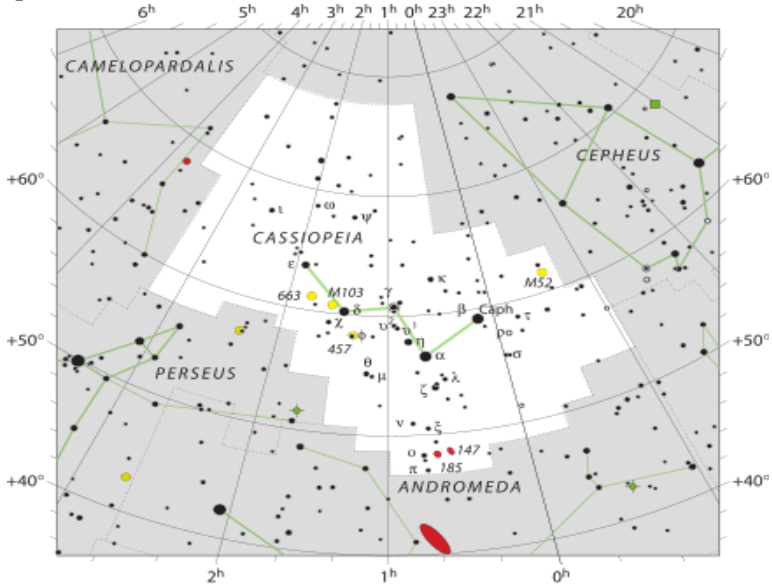

Figure 4.3: Position of the Cassiopeia in the night sky [14]

U-net model was used to carry out the image segmentation component of the task. U-net is a convolutional neural network architecture. It was initially developed for biomedical image segmentation. It is known for its effectiveness to determine object boundaries and segmenting images precisely [15]. Above its initial application in the biomedical field U-net demonstrated favourable results in different image segmentation tasks. Due to the fact that it includes both contracting and expanding pathways to capture complex spatial relationships in images. The general structure of the U-net model is depicted in the figure below. It illustrates its encoder-decoder architecture with skip connections, enabling precise segmentation using integrating features.

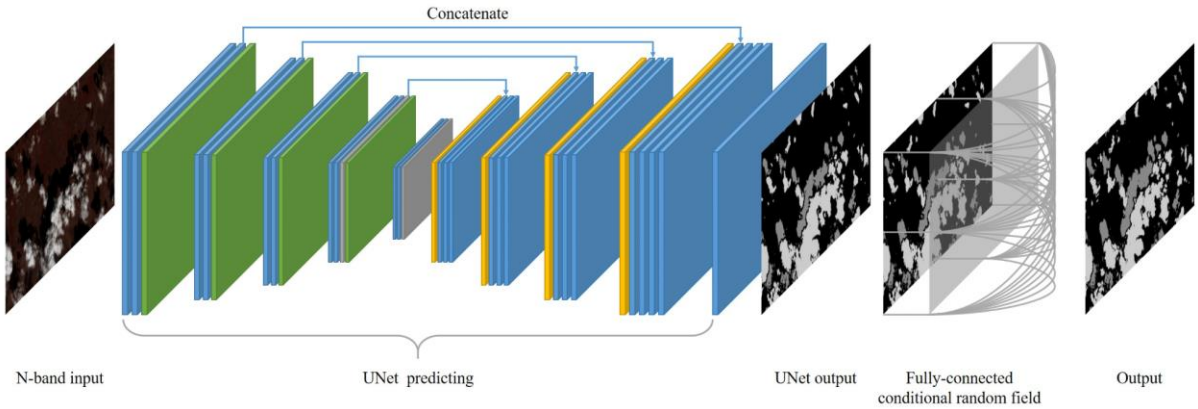

Figure 4.4: General structure of the U-Net model

Specifically for this task U-net architecture was configured with a Resnet34 backbone. It is a variant of the ResNet architecture known for its high learning capabilities in functions like image classification and feature extraction. Pre-trained weights from ImageNet dataset were used to create and train the U-net model.

U-net model was designed using a linear activation function and configured to predict a single channel output. The following output is applicable for sky image segmentation tasks. In addition shape of input of the model was designed to match the dimensions of the input images expected during instruction and inference.

Before fitting the model the image generator function was developed. This prepared it as input for the model fitting process. The given generator function has vital purpose in generating batches of instruction data for the model. This function is run in a loop and yields batches of input as well as related target images. Every batch is calculated when a set of input and target images from specified directories is loaded. The following function utilizes iteration for the instructed data. This ensures that every batch matches to a unique subset of the dataset. The following process keeps going infinitely. This therefore allows continuous instruction data generation during the model-fitting process. By utilizing this generator function the model receives a steady stream of training examples facilitating its learning process and enabling efficient training for multiple epochs. The pseudo-code of the function is illustrated below:

ALGORITHM 2: Function to stream input and target images to the model

```
FUNCTION image_generator():
    WHILE True:
        FOR i FROM 0 TO STEPS_SIZE:
            INITIALIZE batch_x_train as an empty list
            INITIALIZE batch_y_train as an empty list
            FOR j FROM i * BATCH_SIZE TO (i + 1) * BATCH_SIZE:
                image_path_x = INP_IMG_PATH + str(j + 1) + ".png"
                image_path_y = TAR_IMG_PATH + str(j + 1) + ".png"
                LOAD image from image_path_x
                LOAD image from image_path_y
                APPEND loaded image to batch_x_train
                APPEND loaded image to batch_y_train
            YIELD batch_x_train, batch_y_train
END FUNCTION
```

The following model was fitted with a step size of 128 for 70 epochs. The following training lasted for about one week. In order to monitor and observe progress training was periodically stopped every ten epochs to check the performance of the model. At every 10-epoch interval clear improvements were observed.

The Adam optimizer and mean squared error (MSE) loss were utilized to optimize parameters of the model. And to minimize the errors predicted. The following training lasted for multiple epochs. Each epoch was iterated through a predefined number of steps per epoch in order to update weights of the model which was based on training data.

Upon achieving stability in the loss function and observing consistent improvements in model performance, the training process was terminated. A thorough visualization and analysis of the ensuing results were conducted, confirming the effectiveness of the U-net model in segmenting star images for spacecraft navigation applications.

Vision Transformer

In second approach the method of adjusting transformer's architecture and structure to directly extract positional information. This positional information included: Right Ascension (RA), Declination (DEC), and Theta (from the input image). For this, Vision Transformer (ViT) was chosen as architecture to test. Because it provides remarkable performance when handling image data.

ViT transformer is a profound learning architecture. It has received attention for how effective it is when applied in different computer vision tasks. Compared to regular convolutional neural networks (CNN) that rely on convolutional layers for feature extraction ViT uses self-attention mechanism to help it capture global dependencies in input image [16]. This enables VIT to effectively model long-range interactions and extract meaningful positional embeddings which makes it well-suited for tasks requiring precise localization. The general architecture of the ViT is described in the figure below.

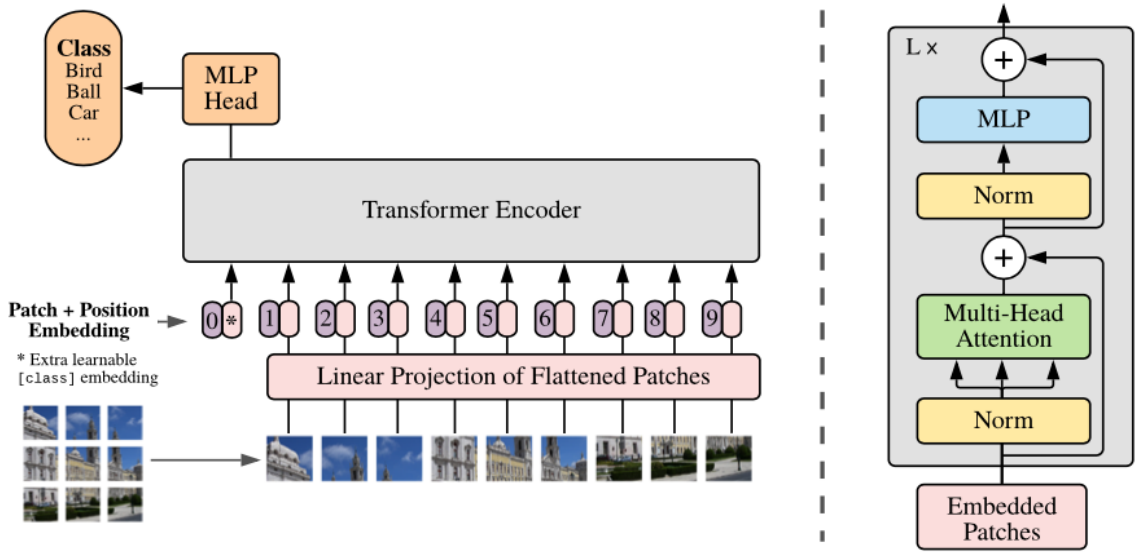

Figure 4.5: General architecture of the ViT

ViT model was applied using a pre-trained version available in Python. This offered a range of models that were designed for transfer learning. One of these options was L/16 variant. It was selected based on its exceptional performance on the CIFAR-10 dataset. The following performance achieved accuracy of 99.36%. CIFAR-10 dataset is widely used benchmark dataset that consists of 60,000 32x32 color images. These images are spread across ten classes. This makes it suitable for assessing the classification performance of the model [17]. L/16

variant was chosen as it demonstrated high accuracy on this dataset. It was expected that ViT would show strong generalization capabilities when used for star position estimation [18].

Pre-trained ViT model was loaded. Following step was to adjust it to specific requirements of the star tracking problem. In order to make this adaptation easier additional layers were added to the base model. These layers were added to tailor it to specifically provided task. Specifically, a dense ReLU layer with 256 nodes was introduced to enhance the model's capacity to capture complex patterns in the input data. Subsequently an output layer comprising three nodes, each corresponding to RA, DEC, and Theta, was appended to the model architecture. This output layer was activated using a sigmoid function to constrain the predicted values within the desired range.

As a result, the positions in the target CSV file were normalized using min-max normalization with appropriate coordinate boundaries. The normalized values of the five target values are presented in the table below.

Table 4.1: Normalized target positions of the images

	RA	DEC	Theta
Sky Image 1	0.208573	0.278189	0.759450
Sky Image 2	0.653541	0.525090	0.636799
Sky Image 3	0.134132	0.462668	0.167021
Sky Image 4	0.680660	0.604051	0.447248
Sky Image 5	0.679084	0.476440	0.805497

After architecture of the model was defined it was compiled using the Adam optimizer and MSE loss function. This was done in order to make efficient training simpler. In addition callback mechanism was applied to monitor performance of the model during training. Also the following mechanism was applied for saving the weights of model to disk in case any improvements were observed. At first, the model was configured to train for 100 epochs with a batch size 32.

However because of computational demands of training a deep neural network each epoch required approximately 8 hours for its completion. Efforts of optimizing the efficiency of the training were applied. This was done by freezing pre-trained ViT model layers. Nevertheless training process lasted more over several weeks. Ultimately, the training was terminated due to memory and CPU limitations on the hardware platform highlighting the challenges of training deep learning models in resource constrained environments.

ResNet-50 Model

For following model selection and training of CNN-based star tracker several architectures were researched. These were carefully evaluated to find a suitable balance between complexity and accuracy trade-off. The following evaluation consisted of popular architectures. Those included: VGG, MobileNet, Inception and ResNet. Each of these offer unique advantages and trade-offs. After broad experimentation and analysis ResNet-50 model became most suitable choice for the task [19]. The following decision was made based on different factors. Those included model's ability to handle complex features extracted from images effectively.

Moreover deep remaining structure of ResNet-50 played important role in minimizing issues associated with vanishing gradients. This is a common challenge in training deep neural networks. Using skip connections ResNet-50 makes flow of gradients during backpropagation easier. This allows for smoother and more stable training. Especially this applies to other network architectures [20]. The following characteristic was important for application of star

tracker. This characteristic applies when model needs to accurately capture patterns and variations in star configurations among different sky images.

In addition pre-trained weights of ResNet50 on large image datasets gives a significant advantage. For example on datasets like ImageNet. This is done by initializing model with acquired definitions of visual features. The following pre-training also aided to speed up process of the model during training on the star image dataset. This led to faster training times and improved generalization to unseen data.

Furthermore, the computational efficiency of ResNet-50, relative to deeper architectures like ResNet-101 or ResNet-152, made it the best choice for practical implementation which ensure reasonable inference times without compromising on performance. It also makes it particularly important for real-time applications such as spacecraft navigation where computational resources are often limited. Here is the general scheme of the Resnet-50 model architecture.

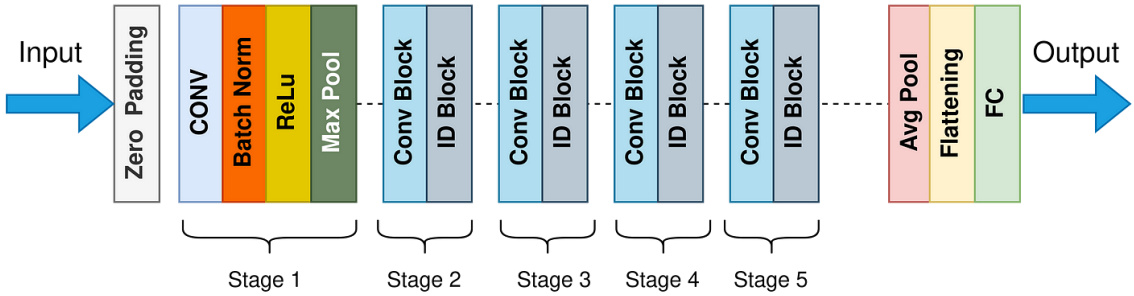

Figure 4.6: General scheme of ResNet50 model

Few layers were included and adjusted in ResNet50 model. This was done to make it fit as star tracker algorithm. Initially shape of the input layer was adjusted to 512x512x3. This aligns with the resized image dimensions. Global Average Pooling (GAP) operation was integrated into the output layer to enhance the model. GAP is a critical step as it helps summarize the features learned by the convolutional layers. It also reduces spatial dimensions and aids to prepare data for further processing. It aids in preventing overfitting and improving generalization capability of the model as well.

Following this a fully connected dense layer was introduced into the network. This layer included 256 nodes as well as a Rectified Linear Unit (ReLU) activation function. The reason that ReLu is applied is that it introduces non-linearity. It also lets the model learn complex patterns and representations in the data. This layer adds depth and complexity to the network and makes it capable of capturing intricate features in the images. Finally output layer was configured with three neurons and a sigmoid activation function. The choice of the sigmoid activation function over the linear function improves the ability of neural network to learn complex relationships in the data. Also, it simplifies gradient flow during training. And it makes sure that the model output aligns with desired scale for the problem.

At last model is compiled with ADAM optimizer and MSE loss function. Total of 15 thousand images are used to instruct the model. Because of memory constraints input images were fed into the network in batches of 32. This was done because loading all images directly into memory was not be optimal. The training process continued for over 100 epochs and took more than two weeks. The loss function was closely monitored throughout each epoch. The best model was saved using a checkpoint. The behavior of loss function after training process is shown in the figure below.

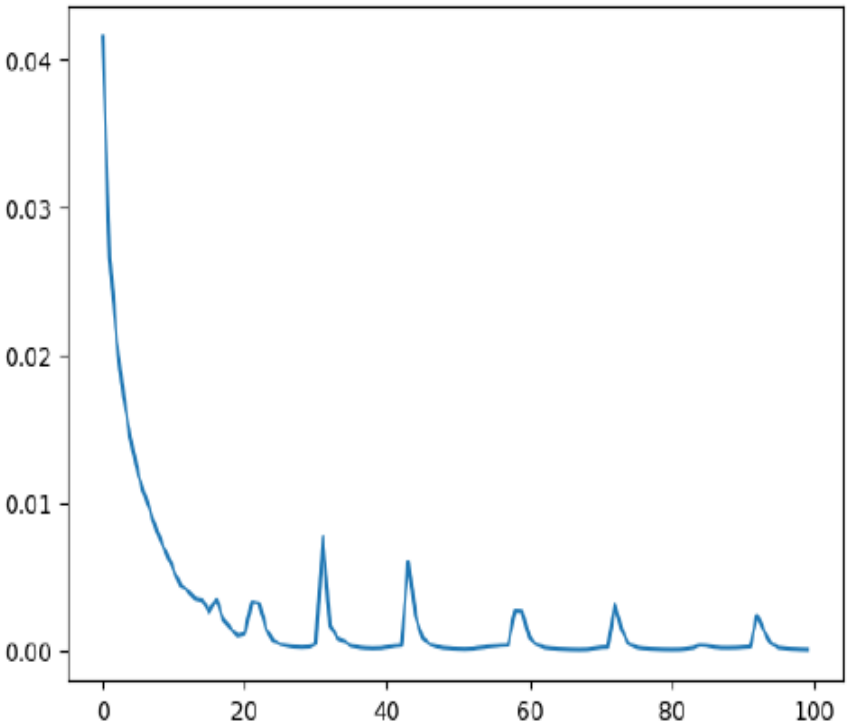

Figure 4.7: The plot of the loss function over each epoch

The figure above illustrates how the loss function value decreases significantly over the epochs. Initial dramatic drop in loss shows rapid learning. At the same time minor changes demonstrate the adjustments of the model to clarify its predictions. This loss declining pattern shows an increase proficiency of the model to determine position arguments. These arguments are necessary for precise astronomical tracking and navigation. On the other hand the following changes also illustrate potential for further optimization. This is possible if it is either done through hyper parameter adjustments or additional training data. These are done to achieve even more reliable and flexible performance in real applications.

Model Evaluation

In order to evaluate the performance of U-net model predicted images were shown using various categories of color maps. These included those from assorted color maps like flag and prism [21]. On the other hand Vision Transformer and ResNet assessment surrounded 2 distinct approaches. In the beginning regression metrics and residual analysis were used (using histograms). This was done to evaluate the effectiveness of the model. As well as its primary role as a regression model. Also regression metrics provided insights into the model's ability to precisely estimate the celestial positions of stars in the captured images. These evaluations were done for all three variables like RA, DEC and Theta.

Second approach transformed the regression task into a classification scenario. This included introducing different degrees of uncertainty to simulate real-world conditions. As well as evaluating the performance of the model (in classification metrics). By considering evaluation as classification task model's flexibility and ability to handle uncertainties in celestial observations were examined. This approach gave valuable insights into how different levels of uncertainty and variability in star positions are well generalized by the model.

Both regression and classification evaluations were needed to get a comprehensive understanding of the performance of star tracker algorithm. Regression evaluation helps to assess the performance of the model as regression task. This applies when predicting precise celestial positions. This is also necessary for high-precision spacecraft navigation. However

classification evaluation also gave insights into the model's ability to handle uncertainties and variations in star positions. This is important for real applications. Observations could be affected by factors like atmospheric conditions and sensor noise. Both approaches are used to test the overall assessment of star tracker algorithm's performance to see if it was achieved. This shows how fit it is for different operational scenarios in spacecraft navigation.

Regression Evaluation

As mentioned two regression evaluations are employed for testing performance of the given model. The first regression focuses on applying regression metrics. Second regression method focuses in assessing residuals and their histograms.

Regression metrics

Four fundamental metrics were used to assess the performance of the model. This is done in the context of regression: Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE) and R-squared (R2). The following metrics have specific purposes in assessing the accuracy and precision of the prediction of the model. These metrics give comprehensive evaluation of the performance of the model to predict celestial positions. They also provide valuable insights into accuracy, precision and overall effectiveness in the context of regression [22].

MAE calculates average absolute differences between predicted and actual values. This offers insight into average magnitude of prediction errors. Lower MAE values demonstrate high accuracy levels. And zero demonstrates perfect predictions.

MSE measures average of the squared differences between the predicted and actual values. The following operation of squares increases the impact of larger errors. The following makes it especially sensitive to outliers in data. RMSE defines square root of MSE. This gives a measure of the standard deviation of prediction errors. Expressed in the same units as the target variable lower RMSE values show improved better model performance.

Lastly R2 is a metric of goodness-of-fit. It shows proportion of difference in target variable (explained by the model). Higher R2 value (ranging from 0 to 1) suggests that model catches more important proportion of difference in target variable. This indicates better alignment between predictions and actual observations of the model.

Residual Analysis

For better understanding of error distribution and assessment of model's performance (from alternative perspectives) histograms were used. This was done to visualize the residuals for RA, DEC and Theta. For visualization of error distribution (using histograms) gives valuable insights into the characteristics of predictions for the model. Also it helps assess its performance from different perspectives [23]. This is done by examining the residuals for RA, DEC and Theta. This gives a better understanding of errors for different celestial coordinates.

Histograms provide graphical representation of frequency and distribution of errors. This allows to identify patterns, trends and outliers in predictions of the model. As example symmetric and narrow distribution of residuals (centered around zero). These can suggest that model shows consistent and accurate predictions with minimal uncertainty.

Classification Evaluation

Classification performance evaluation is important in the context of the star tracker model. This is especially true because of its role in spacecraft navigation. Accuracy of the star tracker directly affects precision of celestial coordinate predictions. This also affects the spacecraft's ability to maintain its given trajectory and achieve mission objectives (due to space conditions reliability is essential).

Spacecraft relies on precise navigation systems to cross large distances and reach their destinations safely. The star tracker is important component of these navigation systems. It provides important information about orientation relative to celestial bodies. Prediction errors can lead to deviations from planned trajectories. Also, potentially jeopardize success of space missions and make it dangerous.

Because of high stakes in space exploration its essential to constantly assess star tracker model's classification performance. Robust classification metrics provide valuable insights into the model's ability to accurately classify celestial objects and predict their positions. Metrics like accuracy, precision, recall and F1-score give comprehensive measures. They measure performance across different classes of celestial objects of the model.

Accuracy is fundamental metric. It assesses how right are predictions of the model [24]. High accuracy levels shows that the model correctly identifies most celestial objects. It also shows that it minimizes the occurrence of misclassification errors.

5. RESEARCH RESULTS

U-Net

As mentioned earlier to test the performance of the U-net model, the predicted images were visualized using a flag and prism. The following figures present comparisons between the desired image and predictions generated using the flag and prism colormaps.

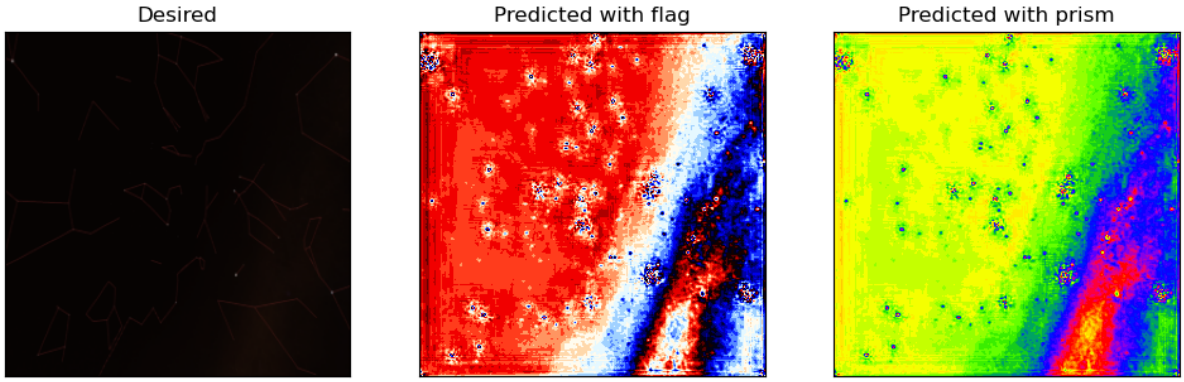

Figure 5.1: First test image predicted using the U-net model

As you can see, the figure above shows that the model accurately identifies primary celestial objects as well as the Milky Way. It successfully detect the Leo Constellation which is positioned at the center of the image and Gemini constellation which was positioned on the right side of it. Impressively, the model also accurately recognizes the important stars such as Castor and Pollux which are essential components of the Gemini.

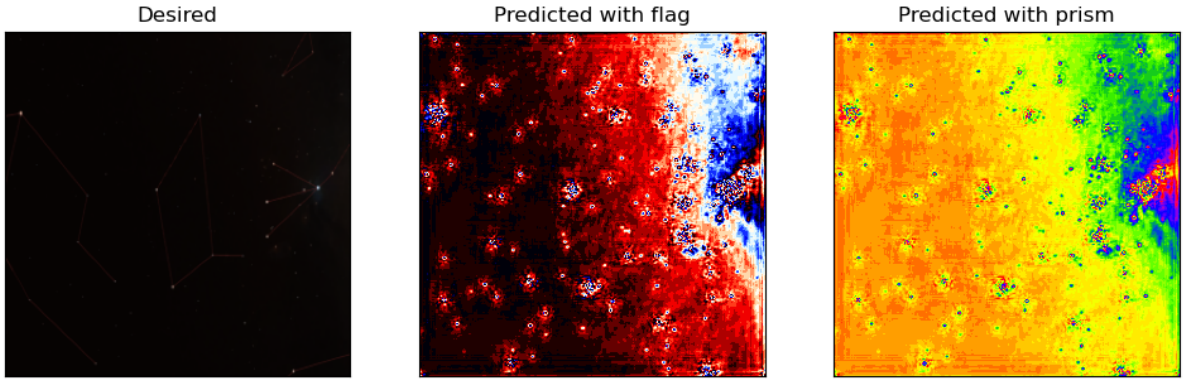

Figure 5.2: Second test image predicted using the U-net model

The ability of the model is also further evident in the second figure. It successfully identify the Scorpios constellation which is located on the right side of the image. Moreover model begins to outline a constellation line that extends from the Antares (central star) to the other stars which basically indicates its ability to establish relationships among celestial objects. This depiction underscores the capacity of the model which not only recognizes individual stars but also understand the connections between entities.

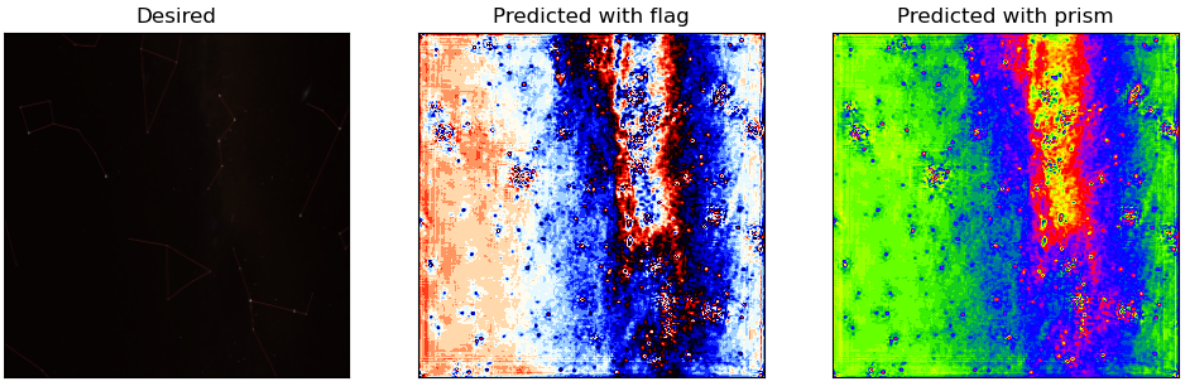

Figure 5.3: Third test image predicted using the U-net model

In the final example image, the model successfully detects the Cassiopeia constellation along with all its main stars. Notably, the model accurately recognizes connecting lines among the stars prominently visible at the center of the image.

All these results from the figures show us that with long training, the model can establish such lines for many of the constellations, and enhance its predictive capabilities and overall performance in celestial identification.

Vision Transformer

Regression Evaluation Results

Regression Performance Metrics

To assess the model's performance in the regression context four metrics were employed. These are Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE) and R-squared (R2). The outcomes of these metrics for the test images are presented in the table below which was predicted using ViT model.

Table 5.1: Regression evaluation of the ViT using test images

	RA	DEC	Theta
MAE	57.024	22.624	82.849
MSE	6268.532	869.441	9526.947
RMSE	79.174	29.486	97.606
R2	0.434	0.421	0.113

As you can see from the results the evaluation of the ViT model revealed considerable discrepancies between the predicted and actual values. MAE values indicate considerable average differences across all three variables. Furthermore the MSE values highlight the extent of the errors of RA, DEC and Theta with values of 6268.532, 869.441 and 9526.947 respectively.

It is evident from the RMSE values that the model predictions deviate significantly from the ground truth values. On the other side, the values of R2 scores of RA, DEC and Theta are 0.434, 0.421 and 0.113 which reflect relatively low levels of explained variance. It basically suggests that the predictions of the model account for only a limited percentage of the variance observed in the data.

Residual Analysis Findings

As mentioned in the methodology histograms were used to visualize the residuals of RA, DEC and Theta. The main target of this evaluation is to understand the error distribution and evaluate the performance of the model from different perspectives.

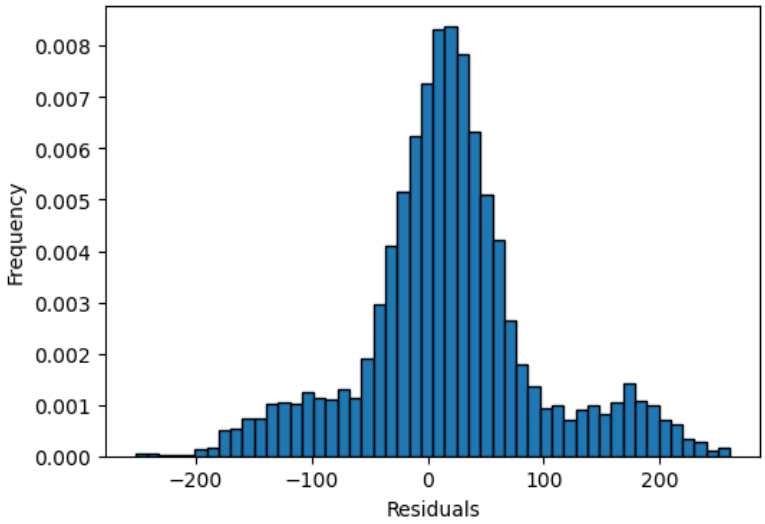

Figure 5.4: RA Residuals Histogram (ViT)

Generally we know that the ideal residuals should be a normal distribution. However from the above histogram we can see that the RA residuals histogram show a distribution with a slight right skewness. The skewness basically indicates a potential bias in model that may overestimate the RA values.

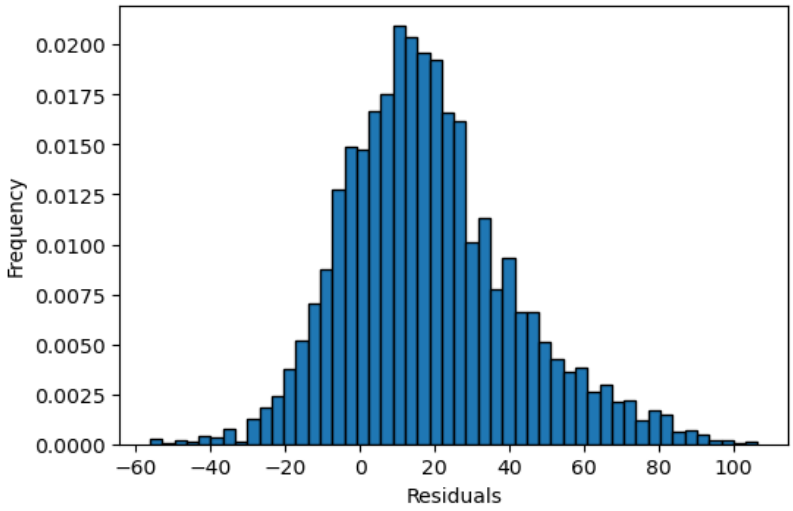

Figure 5.5: DEC Residuals Histogram (ViT)

The histogram of the DEC residuals shows a highly irregular distribution. It lacks a clear central peak which indicates a well-behaved data set. This irregularity may suggest that the predictions of DEC are inconsistent and unreliable. Also the spread of the data points indicates a serious variance which could lead to unpredictable errors in the output.

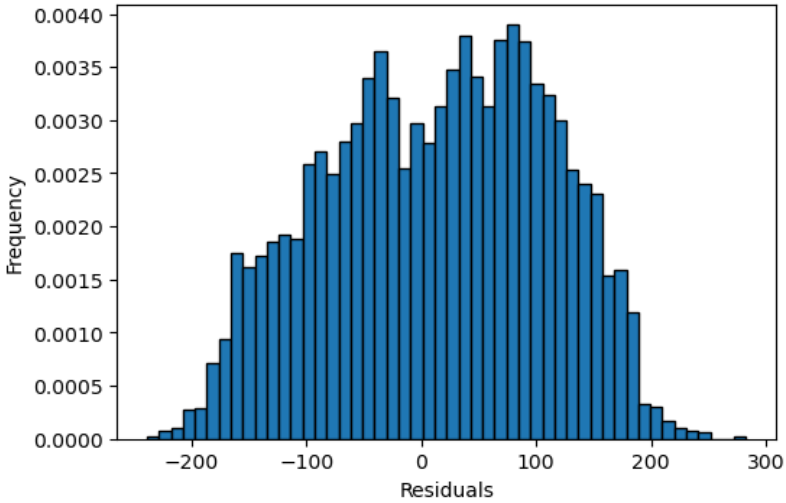

Figure 5.6: Theta Residuals Histogram (ViT)

In the final figure, the Theta residuals histogram shows noticeable irregularities as well. These irregularities could indicate systematic errors in the model in the prediction. Also the deviations from the shape of the histogram raise concerns about the model's accuracy for Theta.

Classification Evaluation Results

As mentioned to evaluate the performance of the model in the classification context the model's accuracy on the residuals of the test data was evaluated across a number of uncertainty values. The outcomes of these evaluations are showed in the below table.

Table 5.2: Accuracy with varied degree uncertainty levels (ViT)

	RA	DEC	Theta
5° ±*	3.17%	8.9%	1.68%
10° ±*	5.98%	18.39%	3.19%
15° ±*	8.19%	27.14%	4.53%
20° ±*	10.80%	35.60%	6.11%

As you can see the results obtained from the ViT classification evaluation exhibit significant differences across all uncertainty degrees. The accuracy rates for predicting RA, DEC and Theta are very low. It ranges from 3.17% to 10.80% for RA, 8.9% to 35.60% for DEC and 1.68% to 6.11% for Theta.

There can be many reasons for this poor performance. First is that, I believe, the ViT transformer is not suitable for this kind of task to predict the celestial coordinates. Because it has limitations in capturing spatial relationships between pixels of the input image. Unlike other convolutional neural network models, transformers rely on self-attention mechanisms to process input data. Also the complexity of the celestial patterns in the image can affect the performance of the model in the worst possible way.

Moreover pre-trained transformers were typically trained on large-scale datasets such as ImageNet thus they may not adequately represent the nuances that is present in celestial images. That is why additional fine-tuning and domain-specific training is needed to get the desired accurate predictions.

ResNet-50 Model

Regression Evaluation Results

Regression Performance Metrics

The evaluation of the ResNet50 model as a regression model using the same metrics is presented in the table below.

Table 5.3: Regression evaluation of the ResNet-50 using test images

	RA	DEC	Theta
MAE	16.933	5.274	19.609
MSE	1850.741	60.167	2007.549
RMSE	43.0202	7.757	44.806
R2	0.833	0.960	0.813

As you can see, based on the evaluation metrics, the performance of the model for predicting the Right Ascension, Declination and Theta appears to be quite good but with some small errors in performance.

For RA, the 16.93° MAE indicates that the model's predictions differ by this amount from the real values. The MSE of 1850.74° and RMSE of 43.02° mean a moderate spread in the prediction errors. However the R2 value of 0.83 indicates that the model captures considerable portion of the variance of the data. It basically demonstrates a reasonably good fit.

In the case of DEC the model performs pretty well as well. The MAE of 5.27° signifies that the predictions are very close to the actual values on average. Both the MSE 60.17 and the RMSE 7.76 are relatively low indicate the minimal prediction errors. On the other hand, the high R2 value of 0.96 indicates an excellent fit which suggest that the model accurately predicts Declination values.

The model performs well for Theta as well but with slightly more error than RA and DEC values. The 19.61° MAE indicates predictions differ with this amount from the actual values which is more than both RA and DEC. The MSE of approximately 2007.55 and RMSE of approximately 44.81 suggest a normal spread in prediction errors. The 0.81 R2 indicates that the model explains a considerable portion of the variance in the data.

In general, the model shows strong predictive capability for Declination followed by Right Ascension while Theta prediction although good exhibits slightly more error. Finally, the R2 values of these three values suggest that the model captures a significant portion of the variability of the data. It, basically, indicates the effectiveness of the model in predicting celestial coordinates.

Residual Analysis Findings

Here are the histograms that were utilized to visualize the residuals for RA, DEC, and Theta to test the performance of the Resnet50 model.

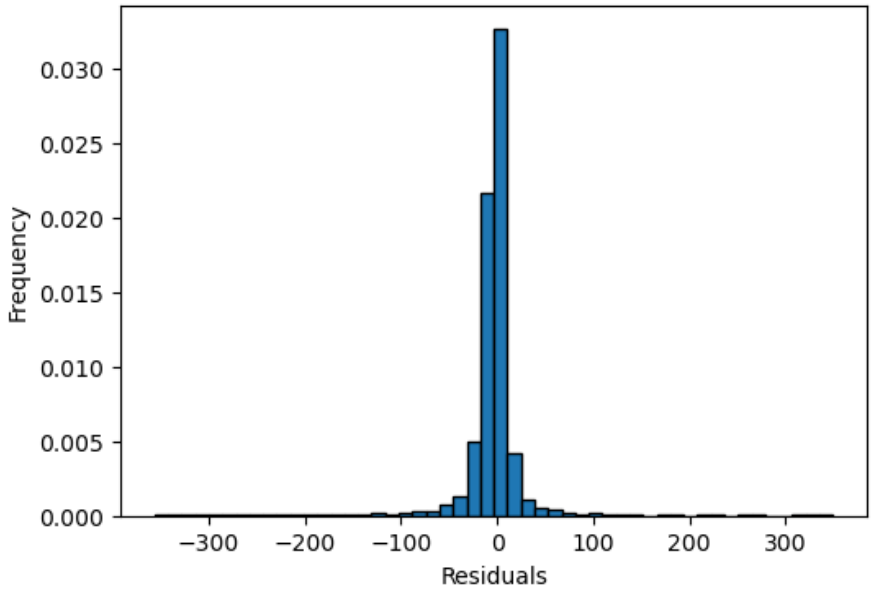

Figure 5.7: RA Residuals Histogram (ResNet-50)

The histogram of RA residuals shows a normal distribution which is a good sign from the perspective of the residual analysis. This indicates that the errors are randomly normal distributed and also the model is well-specified. The peak zero suggests that the model does not systematically overestimate as well as underestimate the RA values. However residuals far from zero value could indicate points where the model predictions are off.

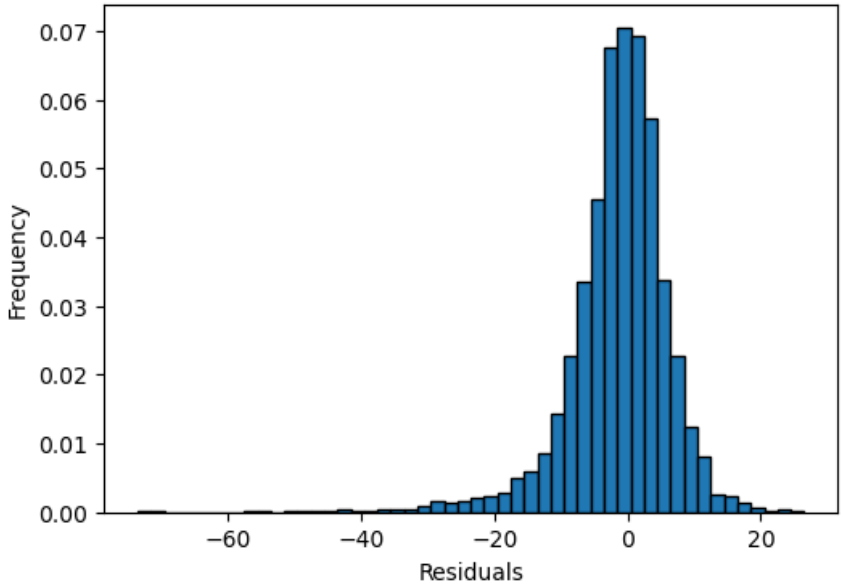

Figure 5.8: DEC Residuals Histogram (ResNet-50)

On the other hand, the DEC residuals histogram shows a very low spread compared to the RA which is located in between -60 and 20. This basically suggest that there are no very high error outliers that the model could not predict. That is why the DEC results are way better than RA residuals.

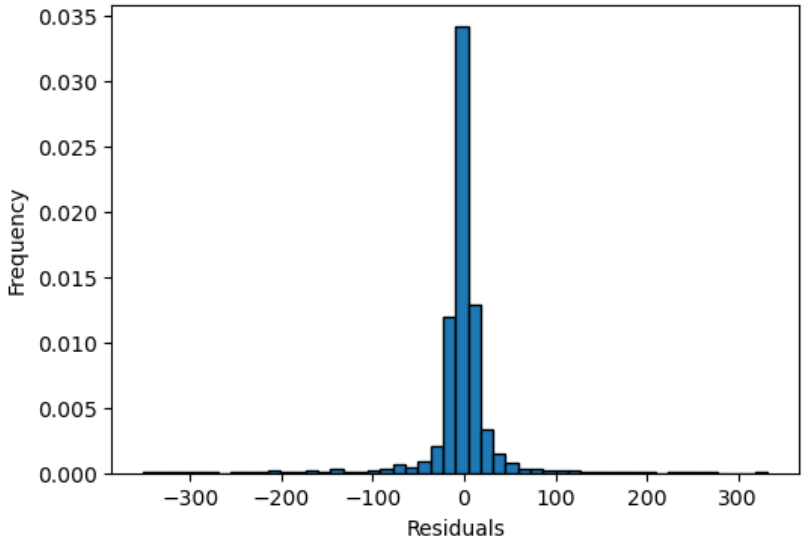

Figure 5.9: Theta Residuals Histogram (ResNet-50)

The Theta residuals histogram shows a distribution similar to the RA residuals where most residuals are close to zero. This suggests that the model's predictions for Theta are generally accurate. However, residuals far from zero could indicate outliers where the model's predictions are less precise like with the RA residuals.

Finally, as evident from the histograms, a near-normal distribution is observed in which errors are centered around a mean of close to zero. This distribution proves that our model is progressing in the right direction to detect the positions from sky images.

Classification Evaluation Results

The table below presents the classification evaluations of the Resnet50 model with different uncertainty degrees.

Table 5.4: Accuracy with varied degree uncertainty levels (ResNet-50)

	RA	DEC	Theta
5° ±*	40.85%	62.32%	34.00%
10° ±*	65.16%	87.44%	58.77%
15° ±*	77.08%	94.83%	73.52%
20° ±*	84.17%	97.48%	80.57%

For the uncertainty of 5° the model achieved an accuracy of 40.85% for RA, 62.32% for DEC and 34.00% for Theta. These results suggest that the predictions of the model for RA and DEC are relatively accurate while DEC exhibits the highest accuracy among the three variables. However the accuracy of Theta is lower than others. It basically indicates that the model may struggle to predict the Theta variable accurately at that uncertainty degree of 5°.

As the uncertainty increases to 10°, there is an improvement in the performance for all three variables which is expected. The accuracy for RA increases to 65.16%, DEC increases to 87.44% and Theta to 58.77%. These results show that the predictions become more accurate because of the increased uncertainty to predict the coordinates.

Further increasing the uncertainty degree to 15° and 20° leads to continuous model performance enhancements. At 15° uncertainty, the model achieves accuracies of 77.08% for RA, 94.83% for DEC, and 73.52% for Theta, and for 20° uncertainty, the accuracies improve to 84.17% for RA, 97.48% for DEC, and 80.57% for Theta. These results indicate that the

predictions of the model improve for increasing levels of uncertainty. However, for every uncertainty level, DEC shows the highest accuracy among the variables.

6. DISCUSSION AND CONCLUSIONS

The research that was presented in this paper shows that the model could perform well for typical CNN problems. However we know that the star tracker requires sub-degree-level accuracy which is why the models are not optimal for this particular problem. Consequently, this research is still in progress to achieve sub-degree-level accuracy which is pretty complicated for any deep learning task.

Indeed despite its current limitations, these models can be used for small satellites like CubeSats. Because for this kind of satellite, we don't need sub-degree-level accuracy thus our models can be suitable for these space missions.

In summary, this thesis has introduced the application of the Convolutional Neural Network to the star tracker sensor of the spacecraft. Before detailing the methodology, this paper reviewed the traditional star tracker algorithms and their current limitations. It explained why the CNN-based approach can be more suitable because of its state-of-the-art working principles. Then the process of simulating data collection was presented, followed by model construction and training phases using different architectures such as U-net, VIT and ResNet50. Furthermore, the evaluation of the models using regression and classification methods was discussed with an analysis of the performance of each architecture.

Additionally it should be also mentioned that these proposed models can be used for other applications as well. Examples can be orbit determination and maneuver planning. In orbit determination star trackers are crucial for determining the position and velocity of the satellite. With an updated algorithm the CNN-based approach can enhance the accuracy significantly. Similarly in maneuver planning, the optimal trajectory should be determined to achieve the desired changes in the orbit of the spacecraft thus the high accuracy and reliability of the star tracker are needed for maneuver planning as well. That is why it is still believed that the CNN-based star tracker algorithm can provide more precise and reliable data to lead the better maneuver planning and optimal efficient use of fuel and resources.

7. FUTURE WORK

As highlighted in above sections important part of future work includes further enhancing the instruction of the model and configuration to improve its accuracy and precision. This will be achieved through strategies. Those include: fine-tuning architecture of the model; optimizing hyperparameters; and expanding the instructed dataset to get a wider range of celestial scenarios.

Mixed strategies include both the strengths of traditional algorithms and CNN-based approaches. These strategies can also be explored and used. These approaches can involve integration of traditional methods for starting star identification and attitude determination. This is followed by use of CNN-based methods for further clarification and improvement in accuracy and precision. These types of mixed approaches can therefore probably improve the total performance of star trackers. Specifically this strategy/approach works sufficiently in challenging environments where traditional methods may struggle.

Once the model reaches sub-degree-level performance plan, this plan includes conducting Hardware-in-the-Loop (HIL) testing to check and measure its effectiveness in real scenarios. HIL testing is an important component in the following: simulating spacecraft's

operational conditions; offering necessary insights into capabilities of the model. This allows for: star tracker integration with other spacecraft components; facilitated testing of their interactions (in a controlled environment); and assessment under various simulated scenarios (these include disturbances caused by vibrations/temperature fluctuations/ radiation).

Hence HIL testing allows to identify and resolve potential issues before spacecraft launch. This type of testing saves time and resources over time. Because of the provided platform for identifying and debugging any potential algorithmic issues HIL testing serves a crucial role in ensuring the reliability, precision and accuracy of the system. It also promotes the development of more advanced algorithms by offering a platform for testing and validation.

One paper that significantly contributed to validating the star tracker model's effectiveness and reliability for practical space missions through HIL testing is "Real-Time Hardware-in-the-Loop Tests of Star Tracker Algorithms" [25]. This research searches into the careful testing and validation procedures that are conducted in HIL setups. These procedures aim to clarify the model and make sure its robustness, accuracy and precision in challenging space environment is at the highest performance.

This paper describes and shows HIL testing setup that is used in the experiments. These experiments involved a star tracker camera mounted on a two-axis gimbal and connected to a custom-built real-time processor. The camera took sky images that were later processed by star tracker algorithm in real-time. The following processed results were then compared to the actual positions of the stars that were obtained from star catalog. These served as the ground truth for evaluating the accuracy and precision of star tracker algorithm. The figures below show an example of HIL testing configuration and testing process the same as described in the research paper which was mentioned in the previous paragraph.

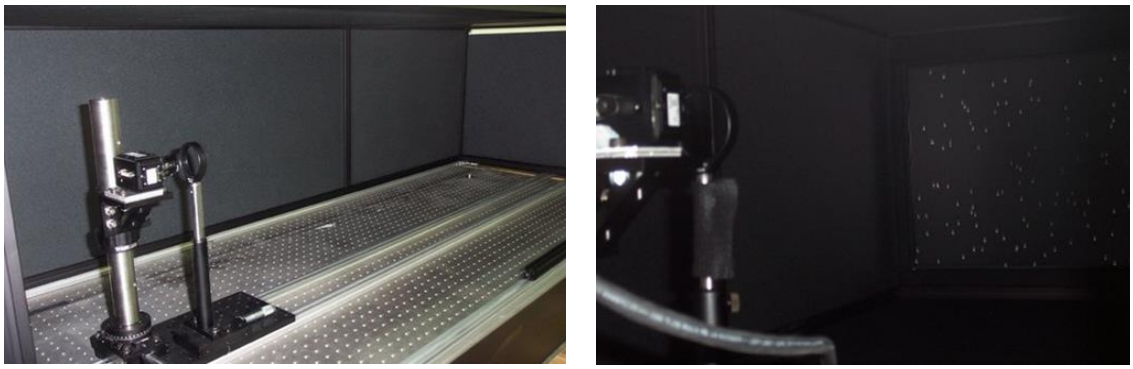

Figure 7.1: Configuration and testing star tracker in HIL testing environment

REFERENCES

[1] NASA. 2021. "Guidance, Navigation & Control." Retrieved May 2023 from https://www.nasa.gov/wp-content/uploads/2021/10/5.soa_gnc_2021.pdf

[2] TY-Space. "A good helper for satellites – Star Sensors." Retrieved May 2023 from https://www.ty-space.net/a-good-helper-for-satellites-star-sensors/

[3] Liebe, C. C. "Star trackers for attitude determination." IEEE Aerospace and Electronic Systems Magazine, 10(6), 10-16, 1995.

[4] IBM. 2022. "Convolutional Neural Network." Retrieved June 2023 from https://www.ibm.com/topics/convolutional-neural-networks

[5] B. B. Spratling, IV and D. Mortari, "A Survey on Star Identification Algorithms," Algorithms, vol. 2, no. 1, pp. 93-107, Jan. 2009.

[6] C. C. Liebe, "Pattern recognition of star constellations for spacecraft applications," IEEE Aerosp. Electron. Syst. Mag., vol.8, no.1, pp.31-39.

[7] J. Hong and J. Dickerson, "Neural-Network-Based Autonomous Star Identification Algorithm," Journal of Guidance, Control, and Dynamics, vol.23, no.4, July 2000.

[8] M. D. Pham, K. S. Low, Chen Shoushun and Y. T. Xing, "A star pattern recognition algorithm for satellite attitude determination," 2012 IEEE Symposium on Industrial Electronics and Applications, Bandung, 2012, pp. 236-241.

[9] "GAIA Catalogues," Wikipedia. Retrieved June 2023 from https://en.wikipedia.org/wiki/Gaia_catalogues

[10] Stellarium. Stellarium Astronomy Software. Available at https://www.stellarium.org. Accessed: June 2023.

[11] NASA. "Reference Systems," Course Chapter. Retrieved May 2023 from https://science.nasa.gov/learn/basics-of-space-flight/chapter2-2/

[12] Isahit. "Image Preprocessing in Deep Learning." Retrieved May 2023 from https://www.isahit.com/blog/what-is-the-purpose-of-image-preprocessing-in-deep-learning

[13] "Cygnus Constellation," Wikipedia. Retrieved June 2023 from https://en.wikipedia.org/wiki/Cygnus_(constellation)

[14] "Cassiopeia Constellation," Wikipedia. Retrieved June 2023 from https://en.wikipedia.org/wiki/Cassiopeia_(constellation)

[15] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and computer-assisted intervention (pp. 234-241). Springer, Cham.

[16] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner (. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. arXiv preprint arXiv:2010.11929, 2020.

[17] Alex Krizhevsky. 2009. CIFAR-10 Dataset. University of Toronto. Retrieved January 2024 from https://www.cs.toronto.edu/~kriz/cifar.html

[18] Google Research Github Repository. "Vision Transformer" Retrieved January 2024 from https://github.com/google-research/vision_transformer

[19] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark Analysis of Representative Deep Neural Network Architectures," IEEE Access, vol. 6, pp. 64250-64264, 2018.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770-778.

[21] Matplotlib. "Colormaps in Matplotlib." Retrieved June 2023 from https://matplotlib.org/stable/users/explain/colors/colormaps.html

[22] Padhma M. 2022. "An Introduction to Evaluating Regression Models." Retrieved August 2023 from https://www.analyticsvidhya.com/blog/2021/10/evaluation-metric-for-regression-models/
[23] Srikanth Varma. 2023. "Residual Analysis." Retrieved September 2023 from https://www.scaler.com/topics/data-science/residual-analysis/
[24] Sun, T., Xing, F., Wang, X. et al. An accuracy measurement method for star trackers based on direct astronomic observation. Sci Rep 6, 22593 (2016).
[25] Rufino, D. Accardo, M. Grassi, G. Fasano, A. Renga, and U. Tancredi, "Real-Time Hardware-in-the-Loop Tests of Star Tracker Algorithms," IEEE Transactions on Aerospace and Electronic Systems, vol. 55, no. 3, pp. 1436-1451.