

ICS3206 - Course Project

Detection of Constellations in Astronomical Images



Name: **Andrea Filiberto Lucas**
Course: Artificial Intelligence (AI)
ID No: 0279704L

Statement of completion

Item	Completed (Yes/No/Partial)
Data set collection	Yes
Data set augmentation	Yes
Implementation of a template-matching system	Yes
Good evaluation of the template-matching system	Yes
Investigation of deep learning approaches	Yes
High-quality report	Yes

Dataset GitHub Repository

Due to the large size of the dataset, it could not be uploaded directly onto the VLE. However, the dataset is accessible via the following GitHub link: GitHub Repository (<https://github.com/AFLucas-UOM/Constellation-Finder>). The dataset is organized within the **‘8CD’ folder, which stands for 8 Constellation Dataset**. This folder contains all the images and resources used in this project.

Abstract

Constellation identification in astronomical images presents unique challenges due to variations in image quality, noise, and the complexity of star arrangements. This report explores and evaluates two distinct approaches for constellation recognition: Template-Based Matching (TBM) and Deep Learning (DL). The TBM approach employs classical computer vision techniques such as ORB feature detection, FLANN-based matching, and RANSAC for homography estimation, leveraging a custom *8CD Dataset* that includes augmented astronomical images to enhance robustness. Despite achieving a modest accuracy of 50% in identifying constellations within the top-3 predictions, TBM demonstrated reliability for constellations with distinct patterns but struggled with ambiguous or overlapping features.

In contrast, the deep learning evaluation highlights the work of [4], where a ResNet-like Convolutional Neural Network (CNN) achieved an F1 score of 0.927 using a dataset of over 1.2 million images. This architecture exhibited strong adaptability to noise and distortions, outperforming TBM in both accuracy and robustness. While CNNs require extensive datasets and computational resources, their ability to generalize and handle complex variability makes them a more promising solution for constellation identification.

This report underscores the strengths and limitations of each approach, providing a comprehensive analysis of their methodologies, evaluation metrics, and results. The findings emphasize the potential of deep learning to advance constellation recognition, while also identifying opportunities to refine classical methods for scenarios with limited computational resources. The comparative insights offer valuable guidance for future research and development in astronomical image analysis.

1 Introduction

This project explores the development of a system designed to detect constellations in astronomical images using template-based matching techniques and a custom augmented dataset. Template-based matching (TBM) identifies objects within an image by comparing them to predefined templates, making it a suitable approach for recognizing structured patterns like constellations.

To achieve this, ORB (Oriented FAST and Rotated BRIEF) was employed for feature detection and description, enabling the identification of key points in both the templates and input images. FLANN-based matching (Fast Library for Approximate Nearest Neighbors) was then applied to efficiently compare these feature descriptors. Lowe's Ratio Test was integrated into the pipeline to filter out weak matches, ensuring higher reliability in the results. Additionally, RANSAC (Random Sample Consensus) was used to compute homography, providing robust alignment and eliminating outliers. These combined methods allowed the system to accurately match templates to celestial patterns, even in the presence of variations in scale, rotation, and other transformations.

To enhance the system's robustness, a custom dataset named **8CD (8 Constellation Dataset)**, was created using screenshots of eight constellations from four online planetariums. The dataset was augmented with techniques such as noise addition, brightness and contrast adjustments, blurring, skewing, and perspective transformations. These augmentations increased the dataset's diversity, ensuring that the system could handle a wide range of image conditions and improve overall performance. Each constellation was systematically varied to support more reliable and efficient processing.

The system ranks matching results based on computed scores, which take into account the number and quality of matches. Visualizations were used to evaluate the system's accuracy and performance, providing insights into the effectiveness of the implemented methods. By combining classical image processing, feature detection, and data augmentation, this project demonstrates a reliable and efficient approach to constellation detection, showcasing the potential of classical computer vision (CV) techniques in addressing structured pattern recognition tasks in astronomy.

2 Background

In this chapter, an overview of the selected constellations and their relevance to template-based matching (TBM) is provided, highlighting their diverse patterns and complexities. Additionally, a brief comparison between classical CV and deep learning (DL) approaches is discussed, emphasizing their respective strengths, challenges, and applicability to tasks like constellation detection.

2.1 Selected Constellations & Their Relevance to TBM

The selected constellations were chosen for their diverse patterns and complexities to thoroughly evaluate the system's template-based matching with feature detection. Corona Borealis, a small semicircular constellation symbolizing a crown, tests structured, curved patterns. Lacerta, with its faint zigzag shape, challenges the detection of less-defined structures. Leo Minor's simple triangular arrangement evaluates basic geometric patterns, while Lepus, depicted as a hare with a loose quadrilateral shape, offers moderate complexity.

Norma, a rectangular constellation, tests alignment capabilities under transformations. Pisces, with its V-shaped pattern representing two fish, and Sextans, a faint constellation with sparse features, present challenges in detecting linked or minimalistic patterns. Finally, Virgo, one of the largest constellations, provides a prominent, scalable pattern for robust testing.

These constellations, with their varying shapes and complexities, were selected to validate the system's ability to detect structured patterns under diverse image conditions using template-based matching.

2.2 Difference between Classical CV & Deep Learning Approaches

In CV, techniques have evolved from traditional methods to advanced DL approaches, driven by improvements in technology and computational power. Classical CV methods are based on clearly defined algorithms and processes. Examples include ORB for feature detection, and FLANN for feature matching. These methods are efficient, reliable, and easier to understand, making them ideal for tasks with structured and predictable input data.

Classical methods are particularly effective in scenarios with limited computational resources or where simplicity and clarity are crucial. They handle variations in scale, rotation, and noise by using techniques like homography estimation with RANSAC. This makes them suited for constellation detection, as the relative positions of stars tend to remain consistent despite varying image conditions.

DL approaches differ from classical methods by using models like convolutional neural networks (CNNs) that learn patterns directly from data rather than relying on predefined rules. This allows them to handle complex tasks, such as detecting constellations in images with varying backgrounds or distortions. Unlike classical methods, CNNs automatically identify important features during training. However, DL requires large datasets, making augmentation techniques like rotation, scaling, and brightness adjustments essential. Training also demands significant time and computational resources, including powerful hardware like GPUs.

Studies on DL for constellation detection indicate that these methods can achieve greater accuracy and adapt to a wider range of image conditions compared to classical approaches. However, these benefits come at the cost of higher computational requirements, more complex training processes, and reduced interpretability compared to traditional methods.

3 Literature Review & Investigation

This literature review explores two prominent approaches for constellation identification: TBM and the DL approach (CNNs). The review also compares these methods, highlighting their respective strengths and limitations, with a focus on accuracy, robustness, and computational demands.

3.1 Template-Based Matching

TBM is a widely used technique in image processing that identifies specific patterns or objects within an image by comparing them to predefined templates [1]. In the context of constellation identification, TBM compares the spatial arrangement of stars in an astronomical image with templates of known constellations. This approach automates the recognition process, eliminates the need for manual identification, and provides an objective and consistent method for analysing celestial patterns [1].

The TBM process begins with image preprocessing, where raw astronomical images are prepared for matching. Gray scale conversion simplifies the image data by focusing on star brightness, while thresholding separates stars from the background, creating a binary representation. Noise reduction techniques, such as Gaussian smoothing, further enhance the clarity of star positions by minimizing random image distortions. These preprocessing steps ensure that the input image is suitable for accurate matching against constellation templates [1].

A key component of TBM is the construction of a constellation database, which houses the templates. These templates can range from simple binary images, where star positions are marked as white pixels on a black background, to more sophisticated data structures that include star attributes like brightness, relative coordinates, and connection lines. Advanced systems may also incorporate neighbourhood information, indicating the spatial relationships between constellations in the celestial sphere, enhancing the identification process. This comprehensive database is vital for ensuring accurate and reliable recognition [2].

The template matching process itself employs various algorithms to compare the preprocessed image with the templates. Normalized Cross-Correlation (NCC) is a common method that produces a correlation map to locate potential matches. Peaks in the correlation map indicate areas where the template aligns well with the image. Additional techniques, such as region labelling and star ranking, further refine the process by identifying connected regions in the binary image and prioritizing stars based on their brightness. To account for scale and rotation variations, algorithms compare the brightest stars in the image with those in the template, adjusting the template's scale and orientation for a better match [3].

Advantages of TBM include its automation capabilities, allowing large-scale data processing without the need for human intervention. This makes it particularly suitable for tasks requiring objectivity and consistency. TBM methods are also computationally efficient when optimized for specific tasks, enabling relatively quick processing times [1].

However, TBM has several limitations. Its performance is sensitive to image conditions such as noise, lighting variations, and distortions caused by atmospheric effects or imaging systems. Partial constellation visibility, where parts of the pattern are obscured, poses additional challenges as incomplete patterns may not match the templates sufficiently. The accuracy of TBM also heavily depends on the quality and completeness of the template database. A missing or inaccurate template can result in misidentification [2].

To address these challenges, robust techniques are employed. Adaptive thresholding and background subtraction mitigate lighting variations, while image correction methods help manage distortions. Systems capable of recognizing partial matches or utilizing constellation neighbourhood information can improve performance under occluded or challenging conditions. For example, Ji et al. [1] demonstrated a constellation detection system with an accuracy rate of 92.8% while processing images in 85 seconds, showcasing the potential of TBM in real-world applications.

3.2 Deep Learning Approaches for Constellation Identification

CNNs are a highly effective DL architecture for tasks involving image recognition and pattern detection, making them particularly well-suited for constellation identification. CNNs leverage their ability to automatically learn spatial hierarchies of features, such as edges, textures, and complex patterns, enabling them to distinguish between star arrangements in astronomical images [4]. This section explores the potential of CNNs for this task, focusing on their architecture, dataset requirements, training process, evaluation methods, and expected outcomes.

CNNs excel at feature extraction by automatically learning the relevant features directly from the input images without manual engineering. For constellation recognition, this capability is critical for capturing subtle variations in star positions, brightness, and inter-star relationships that define constellations. Additionally, their hierarchical structure allows the network to progressively learn more complex features across layers, starting from simple edges to intricate star arrangements [5]. Furthermore, CNNs exhibit translation invariance, enabling them to recognize constellations irrespective of their position in the image. This characteristic is particularly important for astronomical data, where constellations can appear anywhere in the sky.

The choice of architecture would involve a moderately deep CNN, such as ResNet or VGG, to balance the complexity of the network and the available dataset size [4]. Pre-trained models, such as those from ImageNet, could be fine-tuned for this task, significantly reducing the dataset size requirement. While CNNs typically benefit from large datasets, fine-tuning allows effective learning even with moderate datasets, provided that data augmentation is applied extensively [6].

Data augmentation plays a vital role in improving CNN performance by artificially increasing the size and diversity of the dataset. Techniques such as rotation, scaling, flipping, and adding noise can simulate variations in star arrangements, brightness, and imaging conditions. These augmentations help the CNN generalize better to unseen data and improve its robustness to distortions and noise. Additionally, preprocessing the data by converting images to greyscale and applying thresholding and noise reduction ensures that the model focuses on the star patterns while minimizing background interference [5].

Training a CNN for constellation identification requires significant computational resources and time, particularly if a custom architecture is used. The training process involves iteratively optimizing the network's parameters to minimize the error between predicted and actual labels. The availability of GPUs is crucial for accelerating this process, as they allow parallel processing of computations. While training time depends on the dataset size, network depth, and hardware capabilities, the use of pre-trained models can substantially reduce this time. It is also important to monitor for overfitting by using techniques such as dropout and early stopping [7].

The **evaluation of the CNN** would focus on its accuracy and robustness in identifying constellations in test images. A held-out test set containing labelled images not seen during training would be used to assess the model's performance. Metrics such as accuracy, precision, recall, and the F1-score would provide a comprehensive evaluation of the model's effectiveness. A confusion matrix could

further analyse errors, identifying specific constellations that are easily confused. Additionally, robustness testing would involve evaluating the CNN on images with variations in noise, lighting, and distortion to ensure it generalizes well to real-world conditions [4].

Compared to template-based matching, CNNs are expected to offer superior performance in terms of accuracy and robustness. While TBM relies heavily on the completeness and quality of the template database, CNNs can learn directly from the data, making them less dependent on predefined templates. CNNs are also better equipped to handle variations in image conditions, such as noise, distortion, and lighting changes, due to their ability to generalize from diverse training data. However, the computational cost and training time for CNNs are significantly higher than for TBM, making them less efficient in resource-constrained environments [8].

3.3 TBM vs CNN

A well-trained CNN is expected to surpass template-matching methods in constellation identification due to its scalability, adaptability, and precision. CNNs can handle larger and more complex datasets, enabling the development of more comprehensive recognition systems. Unlike template matching, which relies on predefined templates, CNNs automatically learn and extract relevant features, making them more adaptable to variations in noise, lighting, and distortions. While template matching is precise under ideal conditions, CNNs generally achieve higher accuracy, especially when working with noisy or imperfect images. These advantages make CNNs a more promising approach for handling the complexities of astronomical data, providing robust and reliable predictions that address the limitations of traditional template-matching methods [4], [8].

4 Methodology

In this chapter, a detailed overview of the methodology used to create and utilize the 8CD Dataset for constellation detection is provided. The data collection process is described, emphasizing the use of online planetarium platforms and preprocessing to produce clean and annotated image variations. The application of diverse data augmentation techniques to improve dataset robustness is also discussed. Finally, the TBM approach is outlined, highlighting key steps such as feature detection, descriptor matching, homography computation, and validation criteria for accurate constellation detection.

4.1 Data Collection (*8CD Dataset*)

The **8CD Dataset** was created by capturing screenshots of eight constellations from four online planetarium platforms: [9], [10], [11], and the *Night Sky* app [12] available on the iOS App Store. These platforms were chosen for their detailed and interactive star maps, offering diverse visual representations of constellations.

Screenshots were taken for each constellation in two variations: clean versions (with minimal details, such as no lines, limited stars, and no names) and annotated versions (with lines, labels, and names). When annotations could not be removed directly via the planetarium tools, manual preprocessing was employed. Tools like *Cleanup.pictures* [13] were used to erase unwanted elements like lines, and *Shottr* [14] was utilized for tasks such as blurring or removing names, ensuring dataset consistency.

This approach resulted in a dataset capturing a wide range of image conditions, enhancing robustness for both testing and training. By including both clean and annotated versions, the dataset enables a comprehensive evaluation of the system's performance under varied and realistic scenarios.

4.2 Data Augmentation

To enhance dataset robustness and the system's ability to manage diverse image conditions, a custom Python notebook, *DataAugmentor.ipynb*, was used to perform extensive data augmentation on the *8CD Dataset*. Various transformations simulated realistic variations found in astronomical imaging, with the augmented images saved in the *ConstellationName_augmented* directory within the *8CD* folder.

The augmentation techniques included brightness and contrast adjustments to simulate lighting variations, noise addition to mimic sensor artifacts or low-light imaging conditions, and Gaussian blurring to replicate atmospheric distortions or focus issues. Geometric transformations, such as scaling, rotation, and skewing, were applied to account for differences in size, orientation, and projection distortions. Perspective transformations further tested the system's ability to detect constellations under non-orthogonal projections. Flipping operations, both horizontal and vertical, introduced mirrored versions of the constellations to ensure robustness against reversed patterns. Additionally, colour jittering simulated variations in colour balance caused by imaging devices or environmental factors. These transformations were applied systematically, often in combination, to create diverse and realistic variations for each image.

The augmentation pipeline was implemented using libraries such as OpenCV, NumPy, and PIL. It was designed to batch-process directories of images, generating multiple augmented versions of each original image. By generating augmented data, the dataset was effectively expanded, providing the TBM system with more examples to analyse.



Figure 1: Corona Borealis with Constellation Lines [9]



Figure 2: CoronaBorealis Cleaned [9]

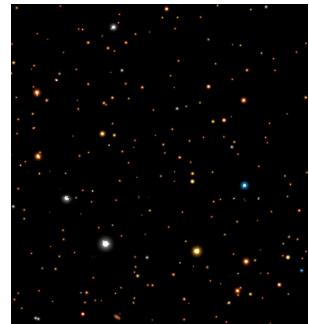


Figure 3: CoronaBorealis Cleaned [12]



Figure 4: CoronaBorealis Brightness & Contrast Augmentation



Figure 5: CoronaBorealis Noise Augmentation

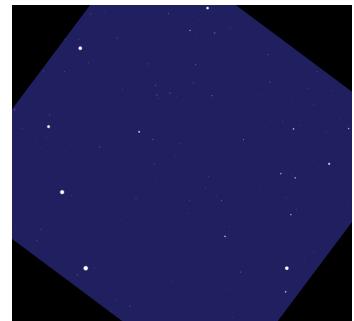


Figure 6: CoronaBorealis Rotation Augmentation



Figure 7: Virgo with Constellation Lines [9]



Figure 8: Virgo Cleaned [11]



Figure 9: Virgo with Constellation Lines & Names [12]



Figure 10: Virgo Blur Augmentation



Figure 11: Virgo Vertical Flip Augmentation

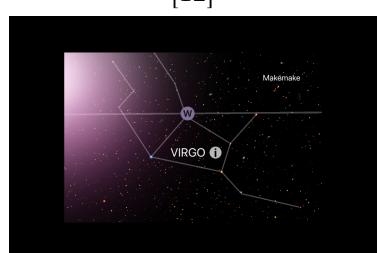


Figure 12: Virgo Perspective Transformation Augmentation

4.3 Implementation of Template Matching System

TBM was used as the core method for detecting constellations in astronomical images. This technique leverages keypoint detection, descriptor matching, and geometric alignment to identify patterns in the sky image that correspond to predefined constellation templates. The process involved several stages and utilized state-of-the-art computer vision techniques implemented within the *Template-BasedMatching.ipynb* notebook.

The first step in the TBM pipeline was to preprocess the sky image and the constellation templates. The sky image was converted to greyscale, and histogram equalization was applied when necessary to enhance contrast. Similarly, all templates were resized across multiple scales to account for variations in constellation sizes due to image capture conditions. Scales ranged from 0.6x to 1.4x, ensuring that the system could effectively detect constellations of varying dimensions.

Feature detection and description were performed using ORB. ORB is a computationally efficient method that identifies key-points and generates binary descriptors for matching. The system detected up to 2,000 key-points in both the input image and the templates, providing a dense set of features for comparison.

Key-point matching was carried out using a FLANN-based matcher optimized for binary descriptors. Matches were filtered using Lowe's Ratio Test, which retained only those pairs of matches where the ratio of the best match distance to the second-best match distance was below a threshold of 0.7. This step significantly reduced the number of false positives, improving the reliability of the matching process.

To refine the matches further, RANSAC was employed to compute a robust homography matrix. This matrix mapped the template coordinates onto the sky image, accounting for transformations such as rotation, scaling, and perspective distortions. RANSAC iteratively removed outliers, ensuring that only consistent matches contributed to the homography calculation. The resulting homography was then used to warp the template's bounding box onto the input image.

The validity of the homography was assessed using geometric constraints. The Shoelace Formula was employed to calculate the area of the warped bounding box, which was compared against pre-defined thresholds. Bounding boxes with areas below 100 pixels or above 10 million pixels were discarded as unrealistic. This step ensured that only plausible matches were considered for further analysis.

For each template, a matching score was computed based on the number of good matches and their average distance. The score was defined as:

$$\text{Score} = \frac{\text{Number of Good Matches}}{\text{Mean Distance} + 1}$$

This formula rewarded matches with a higher count and lower distances, ensuring that the most reliable matches were ranked higher. The system processed each template at multiple scales, retaining the scale that yielded the highest score.

The results were summarized by identifying the top matches across all constellations. The system displayed a visual overlay of the matched key-points and the warped template on the sky image, providing a clear indication of the detection. This method provided a reliable framework for identifying structured patterns in astronomical images, showcasing the strengths of classical computer vision techniques in this domain.

5 Evaluation of Results

In the following section, the performance of the TBM system for constellation identification is evaluated. The evaluation focuses on assessing the accuracy, robustness, and practical effectiveness of the system. Additionally, comparisons with state-of-the-art DL (CNN) approaches will be discussed to provide a comprehensive understanding of the results.

5.1 Evaluation Methodology

The evaluation methodology was structured to thoroughly assess the performance of the TBM system. The process began with the TBM pipeline analysing all templates from the *8CD Dataset*. Matching scores were computed for each template by considering the number of good matches normalized by the mean distance of matched features. These scores quantified the system's confidence in the predictions and enabled the ranking of the top three most likely constellations for each test image. The top-3 prediction framework was chosen as it reflects real-world scenarios where multiple hypotheses are often considered.

To further evaluate the system, ground-truth labels were used to validate the predictions. Specifically, the system was checked to determine if the correct constellation appeared within the top three predictions. The top-3 accuracy provided a practical metric for assessing the system's reliability in identifying constellations under varying conditions.

A confusion matrix was constructed to gain deeper insights into the system's classification performance. Each row of the matrix represented the true constellation label, while each column indicated the predicted label. Correct classifications appeared along the diagonal of the matrix, while misclassifications were shown in the off-diagonal elements. The overall accuracy was calculated using the formula:

$$\text{Accuracy} = \frac{\text{Sum of Diagonal Elements}}{\text{Sum of All Elements in the Matrix}}$$

The accuracy obtained for the TBM system was 50%, indicating that half of the constellations were correctly identified within the top-3 predictions. This accuracy, combined with the confusion matrix analysis, provided a detailed view of the system's strengths and areas for improvement.

5.2 Results and Analysis of the Confusion Matrix

The confusion matrix provided a detailed analysis of the TBM system's classification performance. It illustrated the relationship between true labels and predicted labels, revealing both correct classifications and areas where the system struggled. The diagonal elements of the matrix indicated the number of correctly classified images for each constellation, while the off-diagonal elements represented instances of misclassification.

Using the accuracy formula, the system achieved a 50% accuracy rate, demonstrating that the correct constellation was identified in half of the test cases. This result reflects the effectiveness of the TBM system under ideal conditions while also highlighting challenges in handling more ambiguous scenarios.

The confusion matrix analysis revealed several important observations. Constellations such as *Corona Borealis* and *Leo Minor* were consistently identified correctly, indicating that their templates were well-designed or their star patterns were sufficiently distinct. In contrast, constellations like *Norma* and *Pisces* showed higher misclassification rates, often being confused with similar constellations

such as *Leo Minor* or *Corona Borealis*. These errors suggest limitations in the current template database and the need for additional refinement in template creation or preprocessing techniques.

The system also struggled with constellations that exhibited overlapping or subtle star arrangements. For example, *Pisces* was often misidentified as *Norma* or *Leo Minor*, highlighting areas where the TBM approach faced challenges in distinguishing between similar patterns. These patterns of confusion underline the importance of improving template quality and introducing more sophisticated matching techniques to enhance the system's discriminatory capabilities.

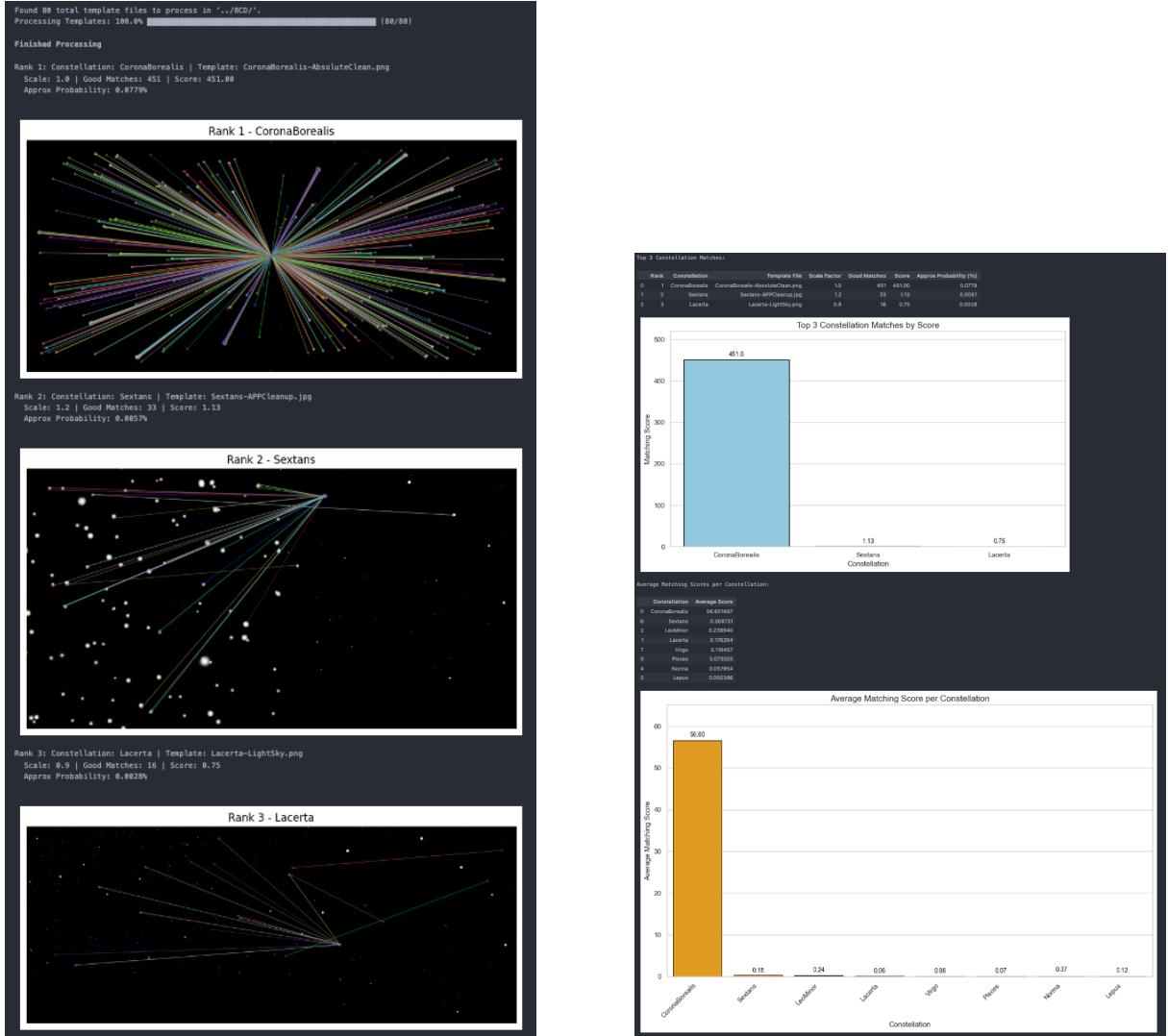


Figure 13: Top-3 results and visualizations for constellation matching.

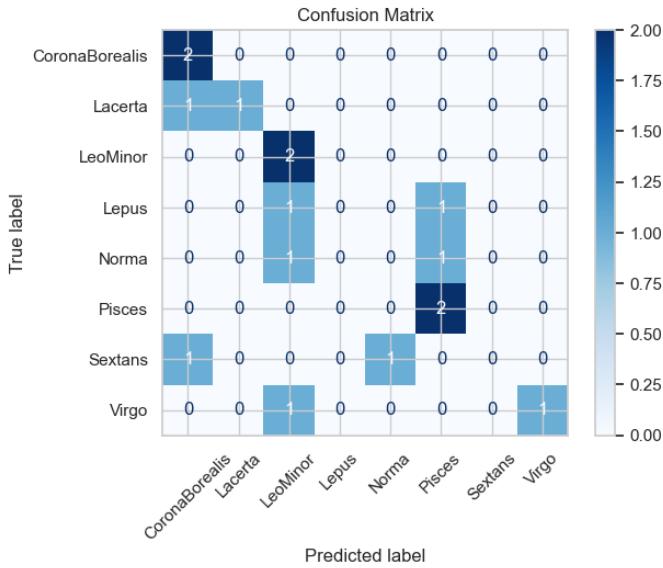


Figure 14: Confusion Matrix of Predicted vs True Constellation Labels.

5.3 Evaluation of Deep Learning Approaches

From the research conducted, the best results using deep learning methods for constellation classification were achieved by [4]. Their work employed a ResNet-like CNN architecture specifically designed for the task, yielding a remarkable F1-score of 0.927. This metric demonstrates the high quality and reliability of the model in identifying constellations.

The neural network was trained on a dataset comprising 1,284,780 images of the starry sky, each with a field of view of 20 x 20 degrees. For verification, a separate test dataset of 48,772 images was utilized. This extensive dataset allowed the CNN to generalize effectively across a wide range of input conditions, a crucial factor for achieving robust performance in real-world applications.

To optimize the training process, the researchers used a combination of binary cross-entropy loss and the Stochastic Gradient Descent (SGD) optimizer. This approach proved more effective than relying solely on binary cross-entropy, as early experiments demonstrated poor performance when the optimizer was not adapted. Through iterative refinement, the model's training strategy was improved, resulting in a significant increase in the median F1-score to 0.927. This improvement highlights the neural network's ability to map the input sky images to the corresponding constellations accurately.

Further evaluation involved testing the adaptability of the CNN under conditions of uncertainty and distortion. By applying random distortions to the constellation patterns, the network achieved a median F1-score of 0.841, demonstrating its capacity to handle stochastic disturbances and incomplete observability. This adaptability suggests that the CNN formed robust representations of star clusters rather than merely memorizing pixel-level patterns. Such generalization capabilities are critical for practical applications where variability in input data is unavoidable.

The project's results illustrate the potential of deep learning for constellation identification. The ResNet-like CNN demonstrated exceptional accuracy and robustness, outperforming traditional TBM methods in terms of both precision and adaptability. By leveraging a well-designed architecture and a comprehensive training dataset, the researchers effectively addressed challenges associated with noise, distortions, and incomplete observations.

6 Conclusion

This report evaluated two approaches for constellation identification: TBM and DL. TBM, achieved moderate success with a top-3 accuracy of 50%, demonstrating efficiency for structured patterns but struggling with noise and overlapping features. In contrast, DL approaches, such as the ResNet-like CNN by [4], achieved an impressive F1 score of 0.927, highlighting their superior adaptability, accuracy, and robustness.

While TBM offers simplicity and computational efficiency, DL clearly outperforms it in handling complex and variable conditions, albeit at the cost of higher resource demands. This comparison underscores the potential of DL as the future of constellation identification, while TBM remains a viable option in resource-constrained settings. Future work should explore hybrid models that combine the strengths of both approaches to achieve greater efficiency and accuracy.

7 References

- [1] S. Ji, J. Wang, X. Liu, and International Astronomical Union, "Constellation Detection," 2021.
 - [2] D. Zuehlke, M. Tiwari, and T. Henderson, "Autonomous Template Generation and Matching for Satellite Constellation Tracking," AIAA SCITECH 2022 Forum, Jan. 2022, sdoi: 10.2514/6.2022-0528.
 - [3] T. Hossler, "Where's Waldo? A Deep Learning Approach to Template Matching," Stanford University, Department of Geological Sciences, journal-article.
 - [4] V. A. Galkin and A. V. Makarenko, "Neural network approach to recognition of visible constellations by sky photo image," Journal of Physics Conference Series, 2021.
 - [5] : M. A. Costa E Silva e, "Machine learning applied to deep space images," 2022.
 - [6] P. Jia et al., "A deep learning based astronomical target detection framework," Astronomy and Computing, 2023.
 - [7] M. J. Lopez-Morales et al., "A Survey about Deep Learning for Constellation Design in Communications," CSNDSP, 2020.
 - [8] D. Zuehlke et al., "Autonomous template generation and matching for satellite constellation tracking," AIAA SCITECH, 2022.
 - [9] *In-The-Sky Planetarium*, available: <https://in-the-sky.org/skymap.php>, accessed: January 14, 2025.
 - [10] *The Sky Live Planetarium*, <https://theskylive.com/planetarium>, accessed: January 14, 2025.
 - [11] *Stellarium Web*, available: <https://stellarium-web.org/>, accessed: January 14, 2025.
 - [12] *Night Sky App*, available: <https://apps.apple.com/us/app/night-sky/id475772902>, accessed: January 14, 2025.
 - [13] *Cleanup.pictures*, available: <https://cleanup.pictures/>, accessed: January 14, 2025.
 - [14] *Shottr*, available: <https://shottr.cc/>, accessed: January 14, 2025.
-

8 Appendix Generative AI - Journal

This Generative AI Journal outlines the integration of AI tools like GPT-4o, GPT o1, Claude 3.5 Sonnet, Quillbot, and NotebookLM into the project. It discusses their contributions, methodologies, ethical considerations, and personal reflections, highlighting their impact on productivity and challenges like bias and hallucinations. This journal provides insights into using AI responsibly to achieve project goals.

8.1 Introduction

In this project, I leveraged multiple generative AI models, each chosen to enhance specific aspects of my workflow. These models included GPT-4o, GPT-o1, GPT-o1-mini, Claude 3.5 Sonnet, Quillbot (Premium), and NotebookLM. The rationale behind using these tools was to streamline processes such as code generation and optimization, in-line code comments and documentation, while also enhancing the accuracy and productivity of the project's outputs.

8.1.1 Reflection on Generative AI Contributions

The primary model utilized was GPT-4o, selected for its superior natural language understanding and ability to generate highly contextualized code. Its capability to suggest optimizations for complex algorithms proved invaluable, particularly in improving model architectures and debugging errors. Additionally, GPT-4o's nuanced approach to explaining theoretical concepts helped in refining the project's methodology and ensuring clarity in technical discussions.

GPT-o1 and its mini variant were particularly helpful when tackling specific coding problems, as their ability to identify and resolve bugs was outstanding. These models not only aided in overcoming technical obstacles but also ensured consistent progress by resolving bottlenecks efficiently.

Claude 3.5-Sonnet was integrated as a secondary review system. Its strengths in summarization and contextual analysis provided an additional layer of validation for generated content, particularly for documentation. Its insights helped to identify gaps in explanations and ensured that all aspects of the project were articulated comprehensively.

Quillbot (Premium) was employed for refining written content and enhancing the quality of documentation. Additionally, it was instrumental in generating references in IEEE 2018 style by processing uploaded PDFs. This dual functionality ensured that the written components of the project were both polished and correctly formatted.

NotebookLM played a key role in generating podcasts and questioning literature from uploaded sources. This tool ensured that concepts were well-understood by synthesizing insights and providing a deeper understanding of theoretical frameworks. It also facilitated the seamless integration of source materials and ensured that the project's deliverables adhered to academic standards.

8.2 Ethical Considerations

Incorporating generative AI into projects requires addressing significant ethical considerations, particularly those concerning data bias and privacy. These factors were carefully evaluated in this project to ensure responsible and equitable use of technology.

A critical ethical concern is the risk of data bias, as generative AI models are trained on vast datasets that may unintentionally embed stereotypes or discriminatory patterns. Such biases can manifest in the outputs, potentially reinforcing inequality. To mitigate this, outputs were closely monitored for signs of bias, and corrective measures were implemented promptly to maintain fairness and inclusivity.

Privacy was another key consideration. Generative AI models often process large datasets, which may include sensitive information. To address this, steps were taken to avoid using identifiable or sensitive data in prompts or uploads, minimizing the risk of exposure.

Additionally, I acknowledged the potential unintended consequences of generative AI, such as the propagation of misinformation or misuse of generated content. To address this, rigorous checking, understanding and testing were prioritized to ensure the accuracy and reliability of the results. Outputs were thoroughly reviewed and refined to reduce errors and prevent unintended negative impacts.

8.3 Methodology for Integrating Generative AI Models

The integration of generative AI into this project followed a structured and iterative approach, utilizing multiple tools at different stages to enhance efficiency and accuracy.

8.3.1 Research Phase

During the initial research phase, NotebookLM played a pivotal role. It was used to question and explore insights from academic papers, generating podcasts for an auditory understanding of complex topics. This approach helped refine the scope of the project and prioritize relevant literature. Once suitable papers were identified, they were uploaded to Quillbot (Premium) to generate IEEE 2018 style references, which were then added to the Overleaf document for consistent formatting.

8.3.2 Code Development

The foundation of the code was developed independently, allowing for a deeper understanding of the implementation. Once a functional prototype was achieved, GPT-4o was employed to optimize the code. This involved asking for improvements, suggesting refinements, adopting better user interfacing (where possible) and incorporating in-line comments for better clarity and maintainability. When errors were encountered, Claude 3.5-Sonnet and GPT-o1 were consulted to identify and resolve issues efficiently.

8.3.3 Results Validation

To ensure the validity of results, output images were uploaded to GPT-4o for secondary verification. This step confirmed that the generated outputs aligned with the expected outcomes and adhered to the project's objectives.

8.3.4 Documentation & Finalization

For documentation, particularly the markdown in Jupyter notebooks, generative AI tools were leveraged to refine and polish the content. The final project report underwent a peer-review process using GPT-4o, which provided critical feedback to ensure all requirements were met and highlighted areas for improvement.

8.4 Prompts and Responses

In this subsection, specific prompts used with generative AI tools are listed, along with the responses and their contributions to the project. Each response demonstrates how these tools facilitated problem-solving, debugging, and idea generation.

- **Prompt 1:** *"Write the following as IEEE 2018 style references:"* (Fig. 15)

Reflection: The AI provided well-formatted references in IEEE 2018 style, reducing manual effort in formatting. This helped maintain consistency across all citations in the report.

Model: ChatGPT-4o

- **Prompt 2:** *"Generate IEEE 2018 style references:"* (Fig. 16)

Reflection: The AI provided well-formatted references in IEEE 2018 style, just by uploading the PDF, reducing manual effort in formatting.

Model: Quillbot (Premium)

- **Prompt 3:** *"Add proper in-line comments to the code snippet below:"* (Fig. 17)

Reflection: The AI generated detailed comments explaining each line of the code. This improved code clarity and ensured better understanding for team members and reviewers.

Model: ChatGPT-4o

- **Prompt 4:** *"Generate a podcast summary of this literature paper."* (Fig. 18)

Reflection: The AI summarized the paper into an audio-friendly format, enhancing comprehension and enabling multitasking during the literature review phase.

Model: NotebookLM

- **Prompt 5:** *"The following code snippet keeps throwing an error; how can I fix it?"* (Fig. 19)

Reflection: The AI identified the issue in the code and provided a clear explanation alongside a corrected version. This reduced debugging time and ensured accurate results.

Model: Claude 3.5-Sonnet

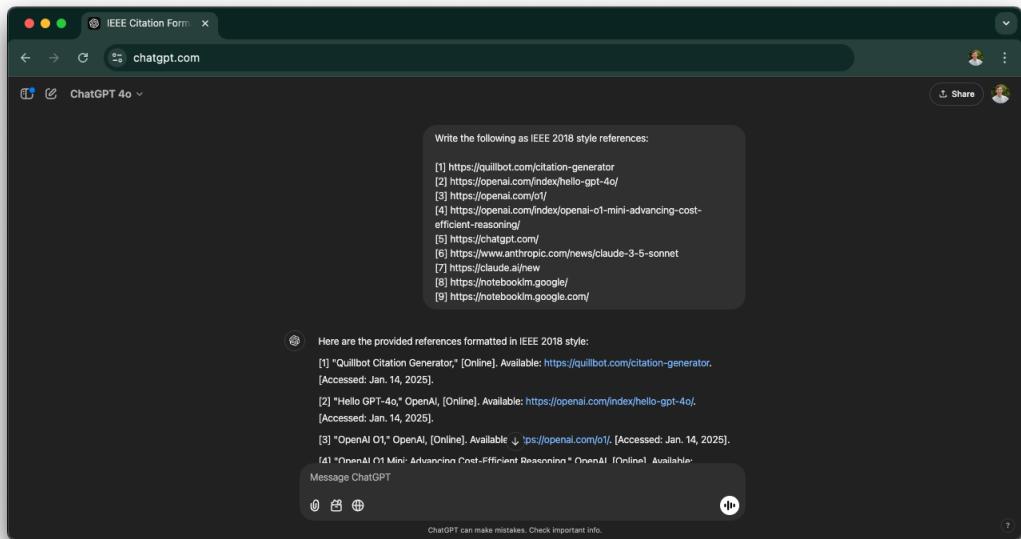


Figure 15: Response to Prompt 1: Formatting references in IEEE 2018 style.

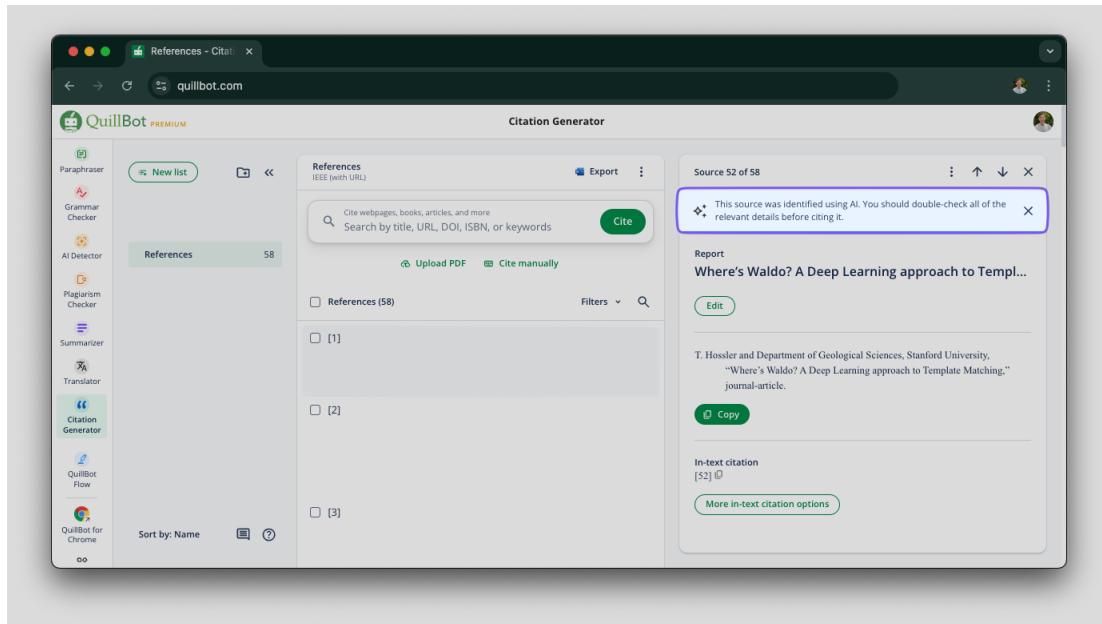


Figure 16: Response to Prompt 2: Formatting references in IEEE 2018 style.

```

Add Proper in-line comments to the following:

...
The predictions were done manually as when loops were
implemented the kernel was crashing due to high memory usage
...

predictions = [
    "1-CB.png": "CoronaBreaks",
    "1-LAC.png": "CoronaBreaks",
    "1-LM.png": "LeoMinor",
    "1-LPS.png": "LeoMinor",
    "1-P.png": "Pieces",
    "1-V.png": "LeoMinor",
    "1-S.png": "Normal",
    "1-N.png": "Pieces",
    "2-CB.png": "CoronaBreaks",
    "2-LAC.png": "CoronaBreaks",
    "2-LM.png": "LeoMinor",
    "2-LPS.png": "Pieces",
    "2-P.png": "Pieces",
    "2-V.png": "Vingo",
    "2-S.png": "CoronaBreaks",
    "2-N.png": "LeoMinor"
]

true_labels = [ground_truth[filename] for filename in ground_truth.keys()]
predicted_labels = [predictions.get(filename, "Unknown") for filename in ground_truth.keys()]

unique_labels = sorted(set(true_labels + predicted_labels))

cm = confusion_matrix(true_labels, predicted_labels,
labels=unique_labels)

disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=unique_labels)
disp.plot(cmap=plt.cm.Blues, ticks_rotation=45)
plt.title("Confusion Matrix")
plt.show()

Here's the updated code with proper in-line comments for clarity:
python
Copy code
# Predictions were generated manually because implementing loops caused the kernel
predictions = {
    "1-CB.png": "CoronaBreaks", # Prediction for the image "1-CB.png"
    "1-LAC.png": "CoronaBreaks", # Prediction for the image "1-LAC.png"
    "1-LM.png": "LeoMinor", # Prediction for the image "1-LM.png"
    "1-LPS.png": "LeoMinor", # Prediction for the image "1-LPS.png"
    "1-P.png": "Pieces", # Prediction for the image "1-P.png"
    "1-V.png": "Pieces", # Prediction for the image "1-V.png"
    "1-S.png": "Normal", # Prediction for the image "1-S.png"
    "1-N.png": "Pieces", # Prediction for the image "1-N.png"
    "2-CB.png": "CoronaBreaks", # Prediction for the image "2-CB.png"
    "2-LAC.png": "CoronaBreaks", # Prediction for the image "2-LAC.png"
    "2-LM.png": "LeoMinor", # Prediction for the image "2-LM.png"
    "2-LPS.png": "Pieces", # Prediction for the image "2-LPS.png"
    "2-P.png": "Pieces", # Prediction for the image "2-P.png"
    "2-V.png": "Vingo", # Prediction for the image "2-V.png"
    "2-S.png": "CoronaBreaks", # Prediction for the image "2-S.png"
    "2-N.png": "LeoMinor", # Prediction for the image "2-N.png"
}

```

Message ChatGPT

ChatGPT can make mistakes. Check important info.

Figure 17: Response to Prompt 3: Adding detailed in-line comments to code.

Deep Learning for Constellation-Based Modulation Classification

Sources

- + Add source
- Select all sources
- Autonomous_Template_Generation_for_-
- Constellation Detection.pdf
- Constellation map with Deep CNN.pdf
- DL2.pdf
- ML for DeepSpace.pdf
- Neural_network_approach_to_recogniti...
- Where is Waldo.pdf
- oe-25-15-17150.pdf
- survey CSDSP 2020 os.pdf

Chat

Deep Learning for Constellation-Based Modulation Classification

Start typing... 1 source

How do deep learning methods improve constellation detection accuracy? >

Studio

Audio Overview

Deep Learning for Constellation-Ba... 00:00 / 16:01

Interactive mode (BETA)

Notes

+ Add note

Study guide Briefing doc FAQ Timeline

Saved notes will appear here

Figure 18: Response to Prompt 4: Generating a podcast summary.

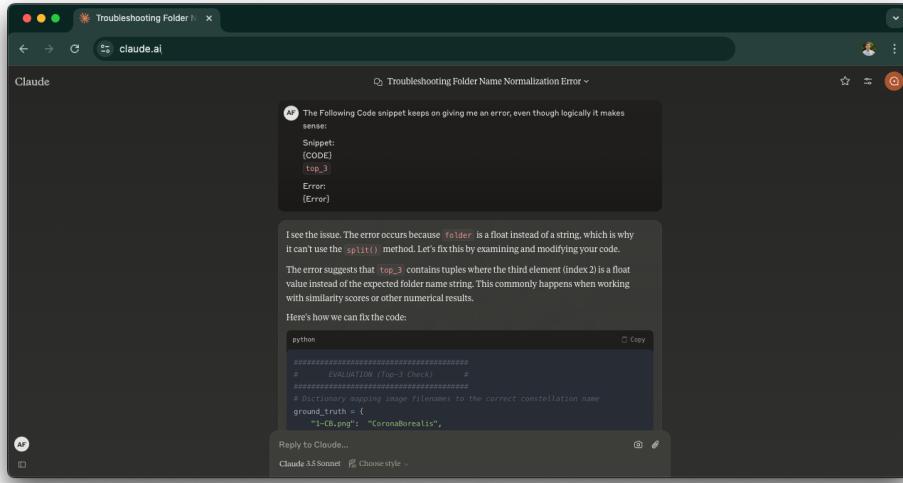


Figure 19: Response to Prompt 5: Debugging a code snippet.

8.5 Improvements, Errors, and Contributions of Generative AI

Generative AI significantly enhanced various aspects of this project, particularly in refining the quality of code, providing clarity through inline comments, and assisting with research. These contributions were instrumental in improving both the technical and written components of the project.

Generative AI played a crucial role in research and literature review. Tools like NotebookLM and Quillbot simplified the process of extracting and formatting relevant references, which streamlined the integration of research papers into the project. Additionally, GPT-4o provided valuable insights for code optimization and offered suggestions that enhanced functionality and efficiency. It also served as a peer reviewer for documentation, ensuring that ideas were effectively communicated and aligned with the project's objectives.

In tasks involving data analysis and results validation, generative AI acted as a secondary verification mechanism. For instance, output images were reviewed using GPT-4o to ensure the results were accurate and met the project's standards. These validations added confidence to the final outputs and helped identify areas requiring adjustment.

However, while generative AI was undeniably helpful, I often preferred to work manually rather than fully depend on it. This was particularly evident during coding tasks, as lower-tier models like GPT-4o and Claude 3.5-Sonnet frequently hallucinated or produced erroneous outputs. Such issues led to a reliance on GPT-01 for error comprehension, as it was better at diagnosing specific problems. Despite these setbacks, the iterative use of generative AI tools alongside manual efforts allowed for the resolution of errors and refinement of results.

8.6 Personal Reflection on Using GenAI

Using generative AI in this project was both an enlightening and mixed experience. On one hand, the tools excelled at tasks like rewording and restructuring text, acting as a peer reviewer, and providing innovative ideas for improving the user interface. These contributions often resulted in solutions or suggestions that would have taken me significant time to devise or might not have occurred to me at all. The efficiency and creativity AI brought to these aspects were surprising and highly valuable.

However, the experience was not without its challenges. Lower-tier models frequently produced errors or hallucinated results, particularly during debugging. This proved frustrating and often led me to rely on manual efforts to resolve issues. In these instances, the time spent attempting to derive useful outputs from generative AI felt wasted. Debugging, therefore, became a domain where personal intervention was more reliable and efficient.

What I learned from this experience is that generative AI is a powerful tool that can significantly augment a student's capabilities, allowing for more comprehensive work within a shorter time frame. However, it is crucial to use AI as a tool rather than a crutch. Over-reliance can hinder learning outcomes and understanding, especially when errors or hallucinations persist. Using AI in conjunction with manual work ensures a deeper grasp of the project's intricacies while still reaping the benefits of efficiency and idea generation.

8.7 Conclusion

This experience has reshaped my perspective on the role of AI in academic projects. While it is undeniably helpful for tasks like literature review, text refinement, and ideation, its limitations highlight the importance of maintaining critical oversight. Generative AI should be leveraged thoughtfully to maximize its potential while ensuring meaningful learning and understanding. In practice, it serves best as a supplementary tool—enhancing productivity without compromising the foundational skills required for academic and professional growth.

8.8 References & List of resources used

- [1] "Quillbot Citation Generator," [Online]. Available: <https://quillbot.com/citation-generator>. [Accessed: Jan. 14, 2025].
 - [2] "Hello GPT-4o," OpenAI, [Online]. Available: <https://openai.com/index/hello-gpt-4o/>. [Accessed: Jan. 14, 2025].
 - [3] "OpenAI O1," OpenAI, [Online]. Available: <https://openai.com/o1/>. [Accessed: Jan. 14, 2025].
 - [4] "OpenAI O1 Mini: Advancing Cost-Efficient Reasoning," OpenAI, [Online]. Available: <https://openai.com/index/openai-o1-mini-advancing-cost-efficient-reasoning/>. [Accessed: Jan. 14, 2025].
 - [5] "Claude 3.5 Sonnet," Anthropic, [Online]. Available: <https://www.anthropic.com/news/clause-3-5-sonnet>. [Accessed: Jan. 14, 2025].
 - [6] "Claude AI," [Online]. Available: <https://claude.ai/new>. [Accessed: Jan. 14, 2025].
 - [7] "Notebook LM," Google, [Online]. Available: <https://notebooklm.google/>. [Accessed: Jan. 14, 2025].
-