MILESTONE REPORT

COVER PAGE

Project Title:
Stellar Mapping

Prepared by:
Rahul Prasanna

Sai Nandini Tata

sivakumar.ramakrishnan@colorado.edu

Course/Subject:
Data Mining

CSCI 5502-872

Instructor:
Alfonso G. Bastias, Ph.D.

Submission Date:
10/18/2024

Website URL : stellar_mapping

GitHub: link

ABSTRACT

Constellations, which are patterns of stars that form recognizable shapes in the night sky, have played an important role throughout human history. They were used as guides for navigation and have been deeply connected to myths, legends, and cultural heritage across different civilizations. However, in recent years, people's knowledge of constellations has faded, largely due to factors such as light pollution, which makes it harder to see the stars, and the fact that fewer people spend time observing the night sky.

This project aims to apply modern data science techniques to develop a model that can automatically identify constellations in astronomical images. By leveraging these technologies, we hope to reignite interest in astronomy and make it easier for people to learn about the stars and use them for navigation. The project also has strong educational potential, as it could be useful for planetariums, science centers, and even professional astronomers.

Additionally, this model aims to address common challenges in constellation recognition, such as dealing with partial visibility of star patterns, differences in image quality, and changing observation conditions. By solving these problems, the project seeks to improve existing constellation recognition systems, making astronomy more accessible and engaging for a wide audience.

DATA COLLECTION

Data for our project has been collected from various reliable online platforms in the fields of astronomy and computer vision. This section shall describe how the data was collected and prepared and also mentions the datasets used.

1. Roboflow Universe Dataset: We utilized the **Constellation Dataset** from Roboflow Universe, which contains approximately **2,000 labeled images** of various constellations. These images include annotations for the stars that comprise each constellation.

The images are of uneven quality and resolution, mimicking how real-world conditions could affect visibility and recognition.

Here's an updated section on **Data Collection/Preparation** that includes more details about the datasets:

- **Link**: [Roboflow Universe Constellation Dataset](https://universe.roboflow.com/ws-qwbuh/constellation-dsphi)

2. **NOAA Milky Way Panorama:**

The Milky Way Panorama dataset from NOAA features high-resolution images of the Milky Way, with **outlines of 88 major constellations**. This dataset provides a detailed view of the constellations' spatial distribution and serves as a reference for understanding how stars are grouped.

- **Link**: [NOAA Milky Way Panorama](https://sos.noaa.gov/catalog/datasets/milky-way-panorama-constellation-outlines/)

3. **Astronomy Online:**
Information from Astronomy Online was used to gather insights into the characteristics and definitions of various constellations. This resource provides detailed descriptions for all 88 recognized constellations, aiding in the labeling process and enhancing our understanding of their significance.

- Link: [Astronomy Online - Constellations](https://astronomyonline.org/Observation/Constellations.asp?Cate=Observation&SubCate=MP07&SubCate2=MP0801)

4. **GitHub Repository:**
The Constellation Detection and Classification repository on GitHub served as a foundational resource, showcasing various algorithms and methodologies for constellation detection. This repository contains over 1,000 images used for training and testing, along with detailed documentation that informs our approach to data preparation.

- Link: [Constellation Detection and Classification GitHub](https://github.com/lowgunnn/Constellation-Detection-and-Classification/tree/master/Constellation%20Classification)

5. **Instructables Star Recognition:**
The Instructables article on Star Recognition Using Computer Vision provided practical insights into using OpenCV for star recognition. While this resource doesn't offer a specific dataset, it provides guidelines and methodologies relevant to image processing, which were crucial for our data preparation steps.

- Link: [Instructables Star Recognition](https://www.instructables.com/Star-Recognition-Using-Computer-Vision-OpenCV/)

Data Preparation information

Data Preprocessing Steps

In this project, we performed several data preprocessing steps to prepare our dataset for training a model to identify constellations in astronomical images. The following outlines the key steps involved in our preprocessing pipeline, along with code snippets and explanations.

Here's a breakdown of each step:

1. Resize: transforms.Resize((128, 128))

Resizes the input images to a fixed size of 128x128 pixels. This ensures uniformity in input dimensions, which is crucial for batch processing in neural networks.

2. Random Horizontal Flip: transforms.RandomHorizontalFlip()

Randomly flips the image horizontally with a probability of 0.5. This augmentation helps the model become invariant to horizontal orientation, which can improve generalization.

3. Random Rotation: transforms.RandomRotation(15)

Randomly rotates the image by up to 15 degrees in either direction. This adds rotational variability to the dataset, helping the model learn features that are less sensitive to orientation.

4. Convert to Tensor: transforms.ToTensor()

Converts the image (likely in PIL format or a NumPy array) to a PyTorch tensor. This transformation also scales pixel values to the range [0, 1].

5. Cutout: Cutout(n_holes=1, length=16)

This custom transformation randomly selects a region (16x16 pixels) in the image and sets its values to zero. This simulates occlusion, encouraging the model to focus on the remaining visible features and improving robustness.

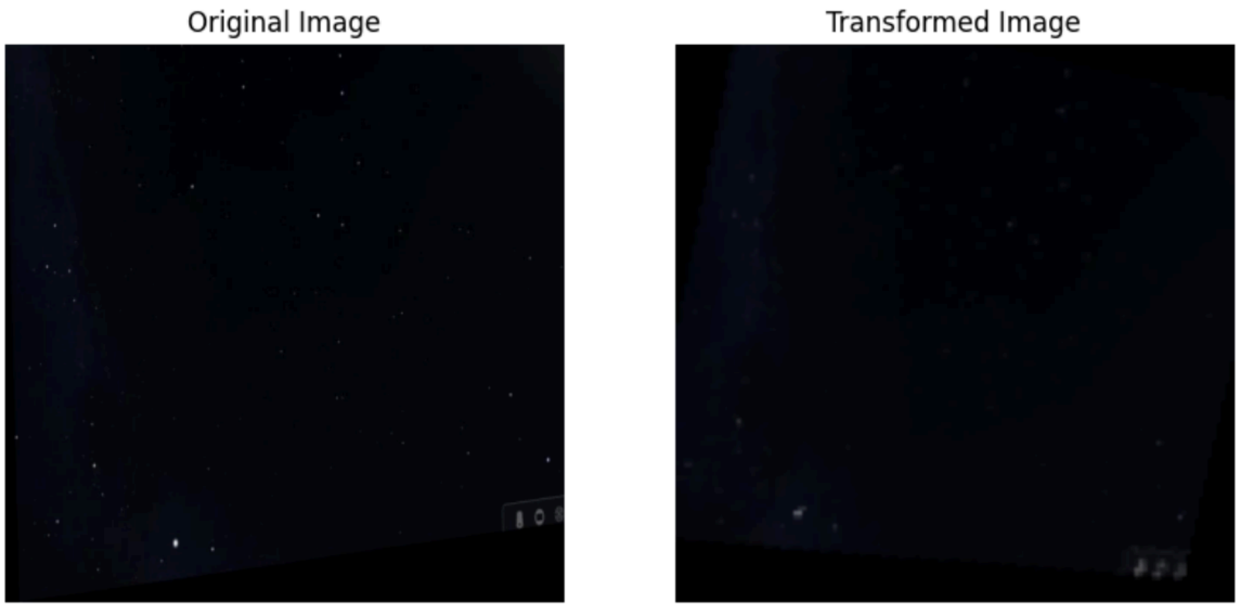
6. Normalize: transforms.Normalize((0.5,), (0.5,))

Normalizes the tensor image to have a mean of 0.5 and a standard deviation of 0.5 for each channel. This is important for speeding up convergence during training and can help the model perform better by ensuring that inputs are centered around zero.

Together, these preprocessing steps enhance the dataset through augmentation (increasing diversity), ensure consistent input sizes, convert images to the required format, and normalize pixel values for effective model training. This pipeline aims to improve model robustness and generalization by exposing it to varied versions of the input images.

We load an example image and apply the defined transformations. The original and transformed images are displayed side by side for visual comparison. This helps to illustrate the effect of our preprocessing pipeline.

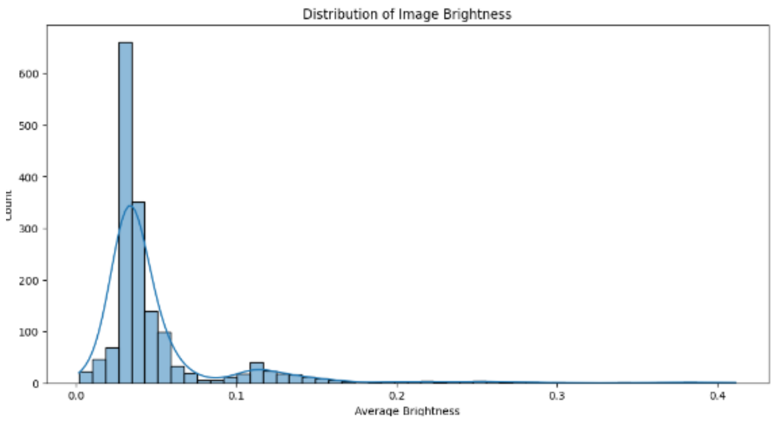
The following contains a sample from the dataset that shows how images are, before and after preprocessing:



Visualization

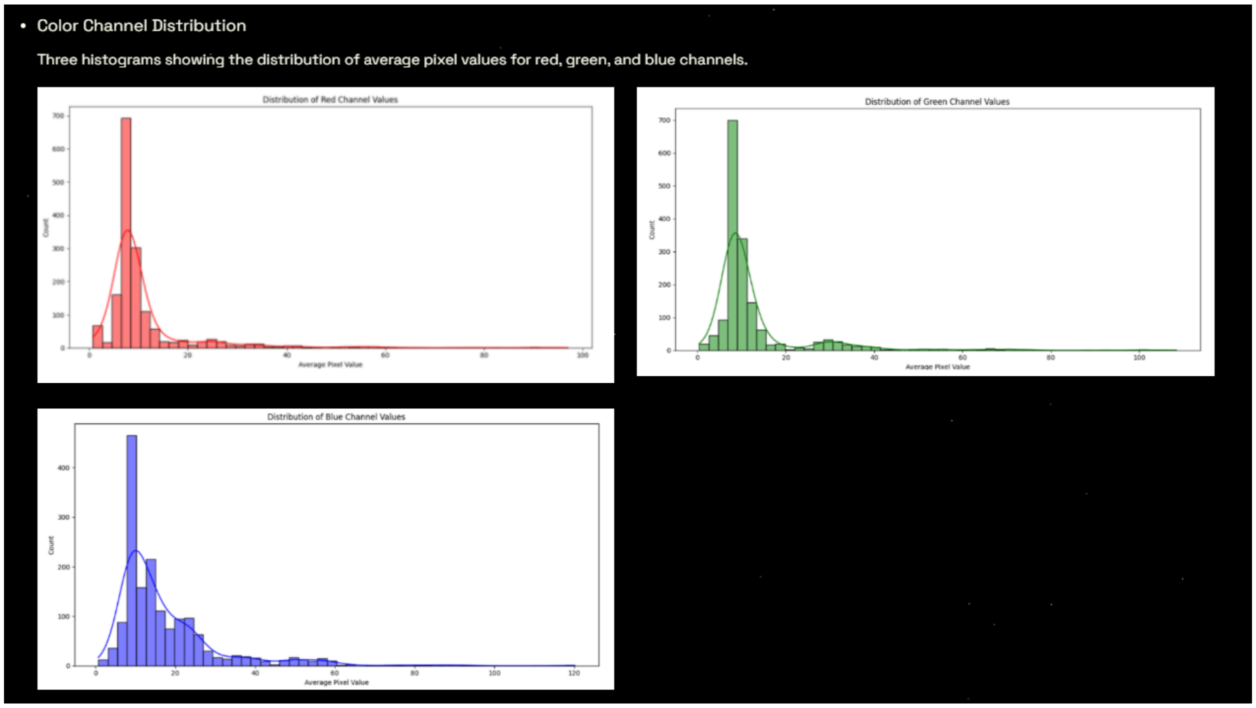
Each plot explanations:

1. Distribution of Image Brightness
    - Description: A histogram of average pixel brightness for each image.
    - Uses:
        - Reveals if images are generally dark, bright, or have a good spread of brightness levels.
        - Helps decide if brightness normalization is needed in preprocessing.
        - Identifies potential issues with over or underexposed images.



2. Color Channel Distributions

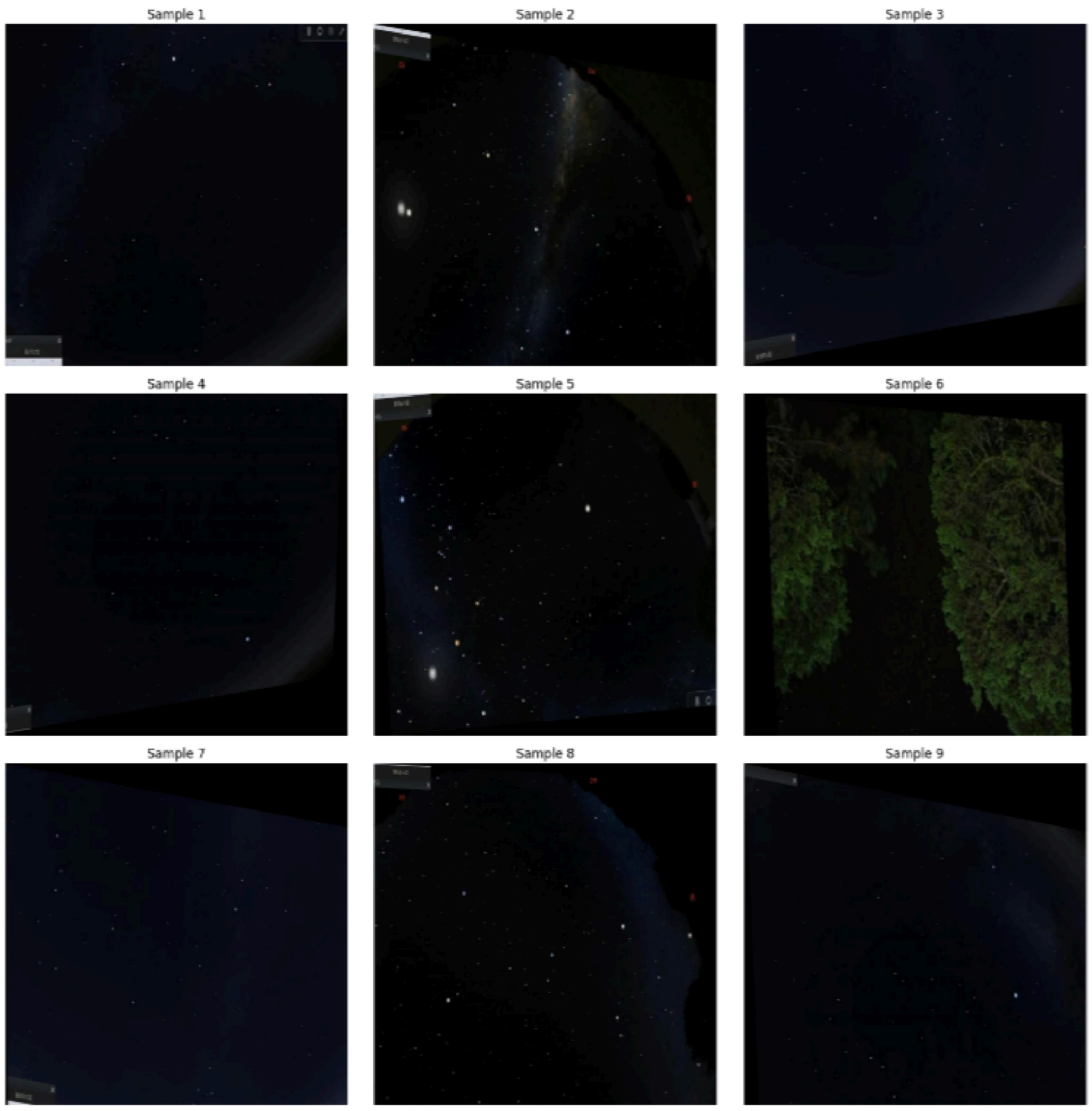
- Description: Three histograms showing the distribution of average pixel values for red, green, and blue channels.
- Uses:
    - Reveals color biases in the dataset (e.g., if images tend to be more red or blue).
    - Helps in deciding if color normalization or balancing is needed.
    - Can indicate issues with color consistency across the dataset.



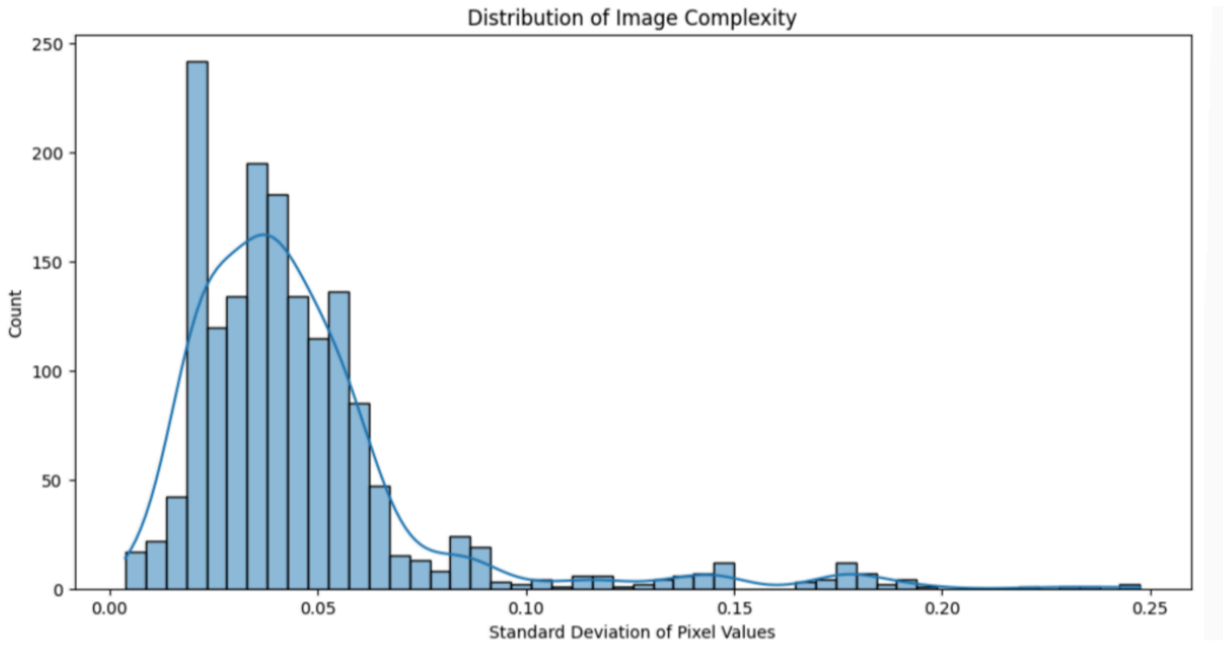
- Color Channel Distribution

  Three histograms showing the distribution of average pixel values for red, green, and blue channels.

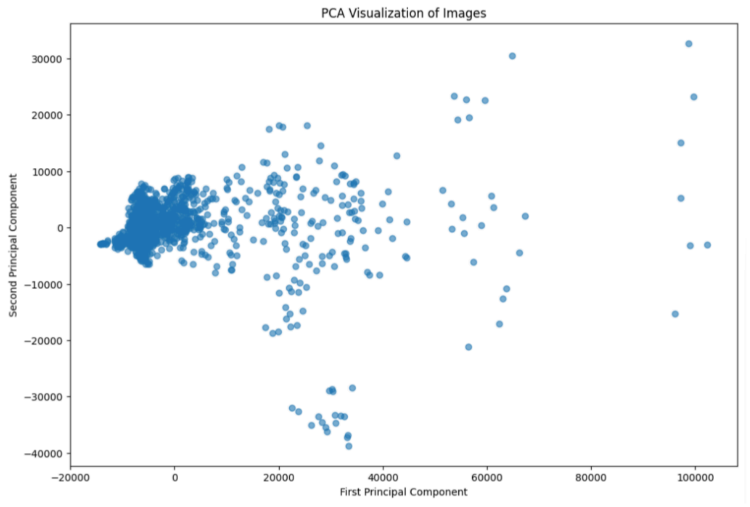
3. Sample Images
    - Description: A grid of 9 randomly selected images from the dataset.
    - Uses:
        - Provides a visual inspection of the types of images in the dataset.
        - Helps in identifying any obvious issues with image quality or content.
        - Gives a sense of the variety in the dataset.



4. Distribution of Image Complexity
    - Description: A histogram of image complexity, measured by the standard deviation of pixel values.
    - Uses:
        - Indicates the range of image complexities in the dataset.
        - Helps identify if there's a good mix of simple and complex images.
        - Can guide decisions on data augmentation or model complexity.



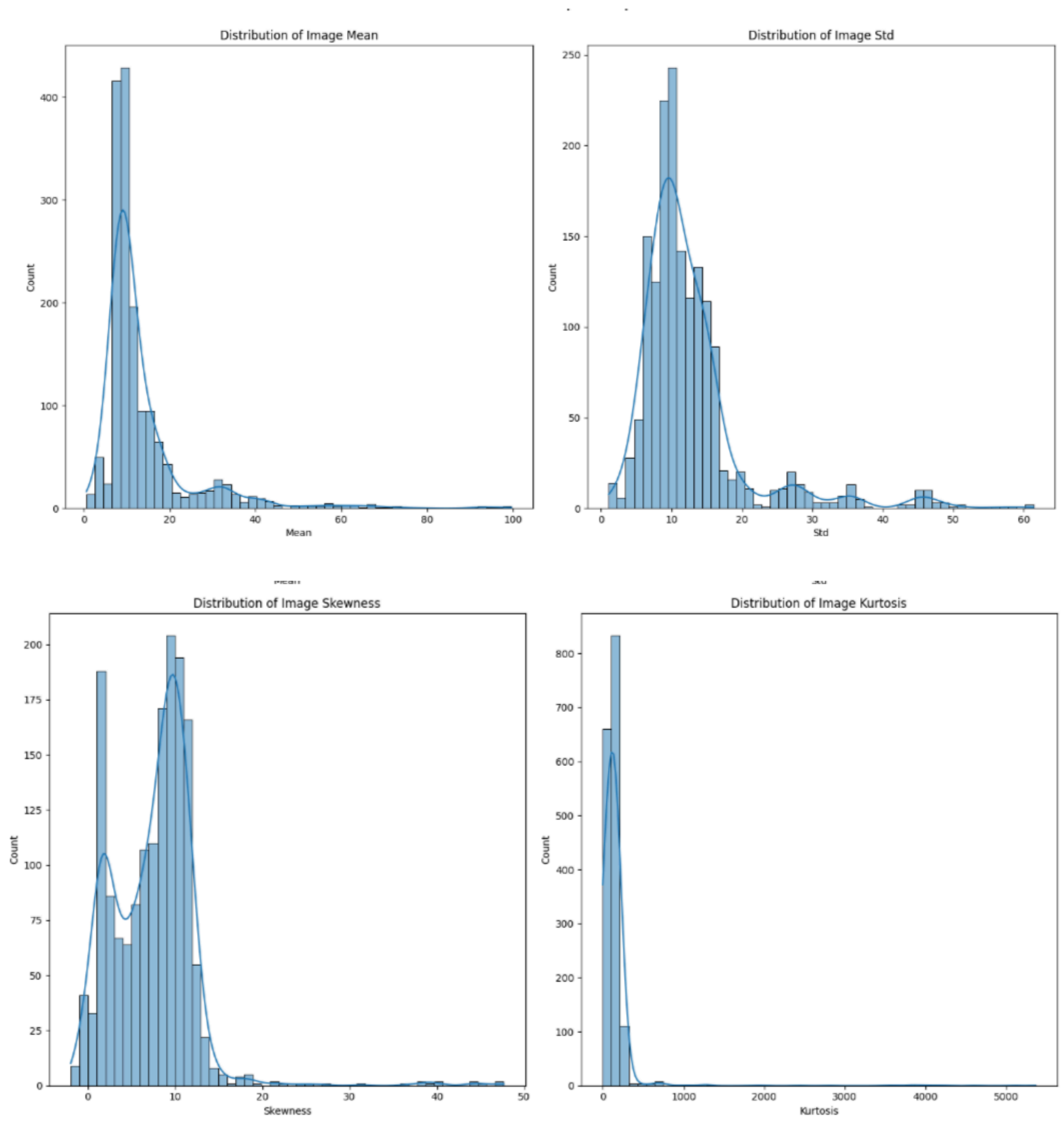
5. PCA Visualization of Images
    - Description: A scatter plot of images projected onto their first two principal components.
    - Uses:
        - Reveals clusters or patterns in the dataset that might not be apparent from other plots.
        - Can indicate if there are distinct groups of similar images.
        - Helps in understanding the overall structure and variability in the dataset.



6. Image Statistics Distributions
    - Description: Four histograms showing the distribution of mean, standard deviation, skewness, and kurtosis of pixel values across all images.
    - Uses:

- ■ Mean: Indicates overall brightness and color trends.
- ■ Standard Deviation: Reflects contrast and complexity.
- ■ Skewness: Shows if pixel values tend to be above or below the mean.
- ■ Kurtosis: Indicates the presence of extreme pixel values.

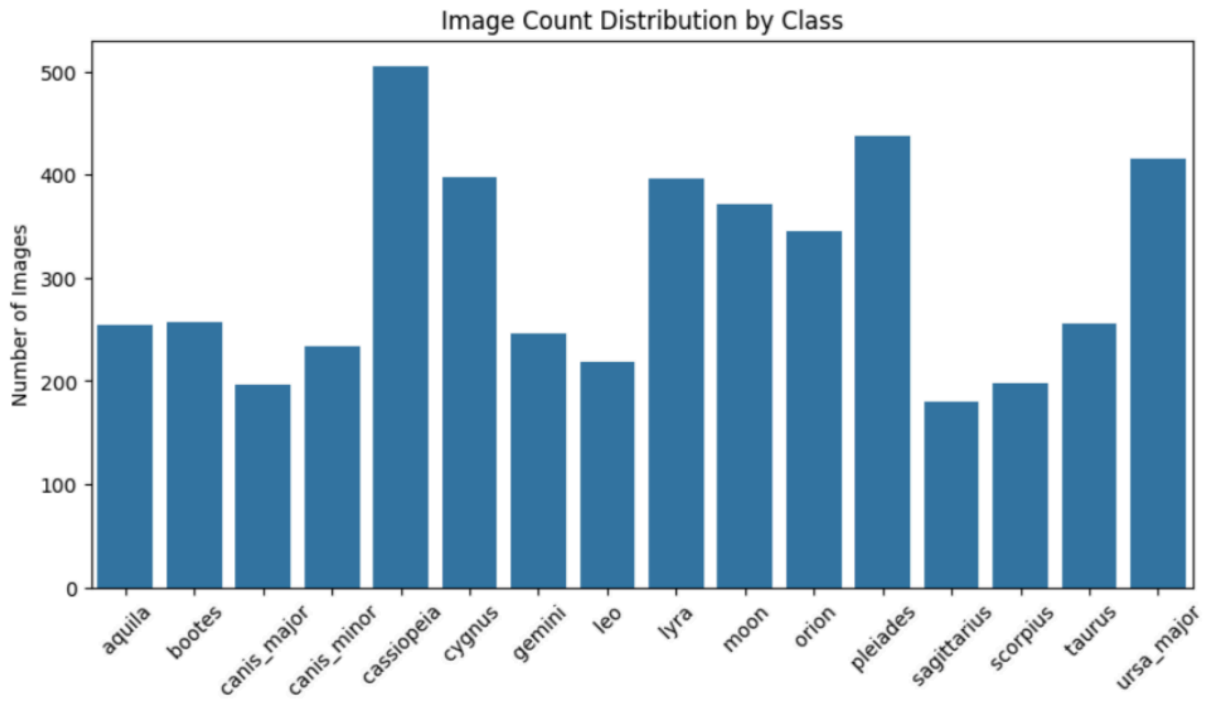
○ These statistics can help in identifying outliers, understanding the overall characteristics of the images, and guiding preprocessing decisions.



7. Image Count Distribution by Class

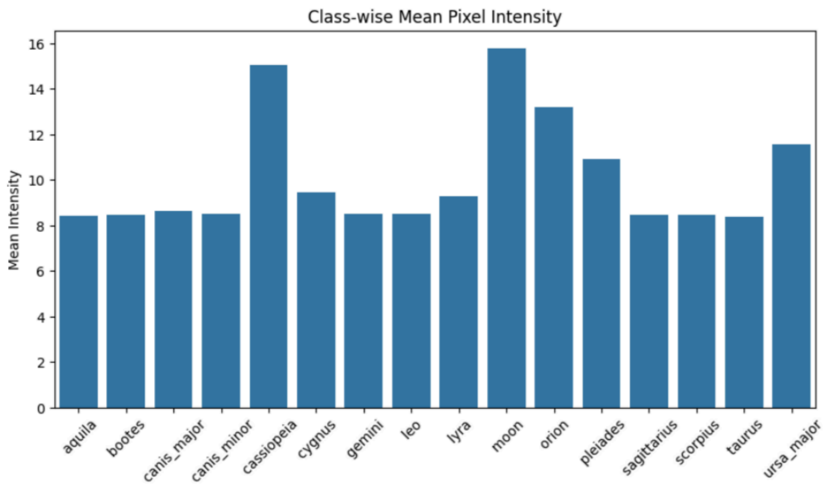
- **Purpose**: To understand the distribution of images across different classes.

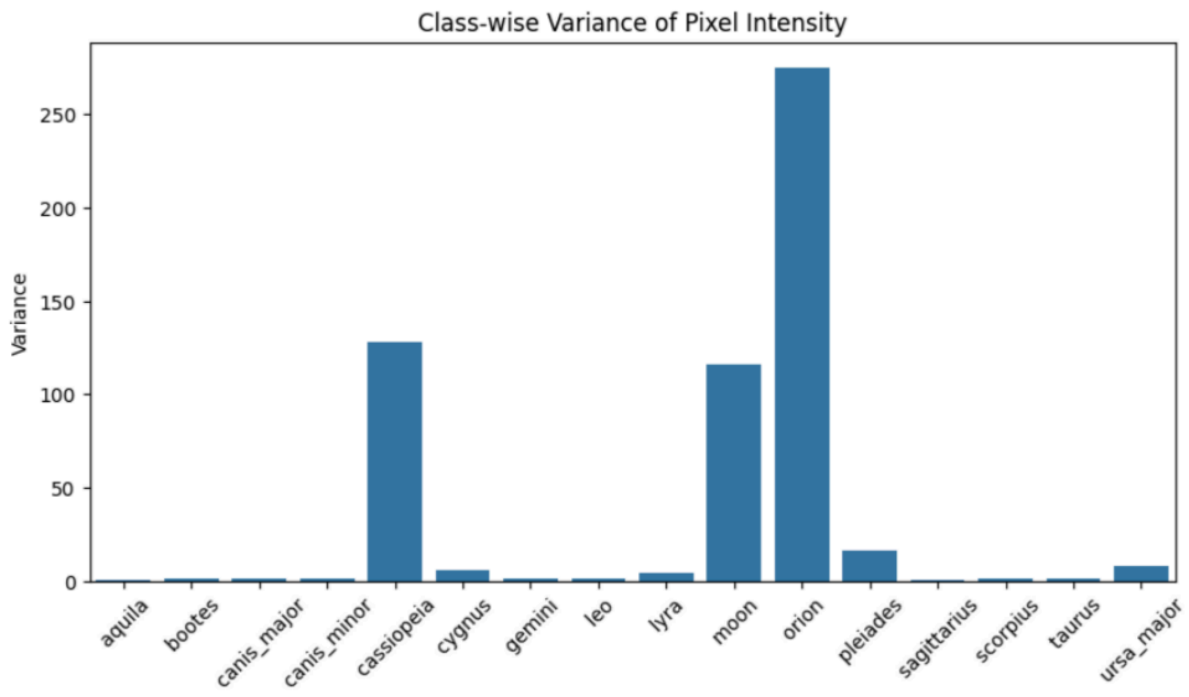
- **Why**: Identifying class imbalances is crucial for model training. If one class has significantly more images than another, it may lead to bias in the model's performance, favoring the more abundant class.



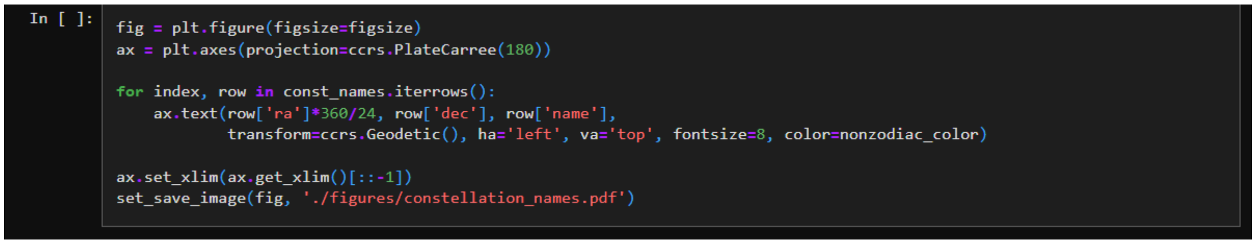
8. Class-wise Mean and Variance of Pixel Intensity

- **Purpose**: To evaluate the average and variability of pixel intensities for each class.
- **Why**: Understanding mean and variance can reveal differences in brightness or contrast between classes, which could inform preprocessing strategies and help optimize model training.





9. Constellation mapping:

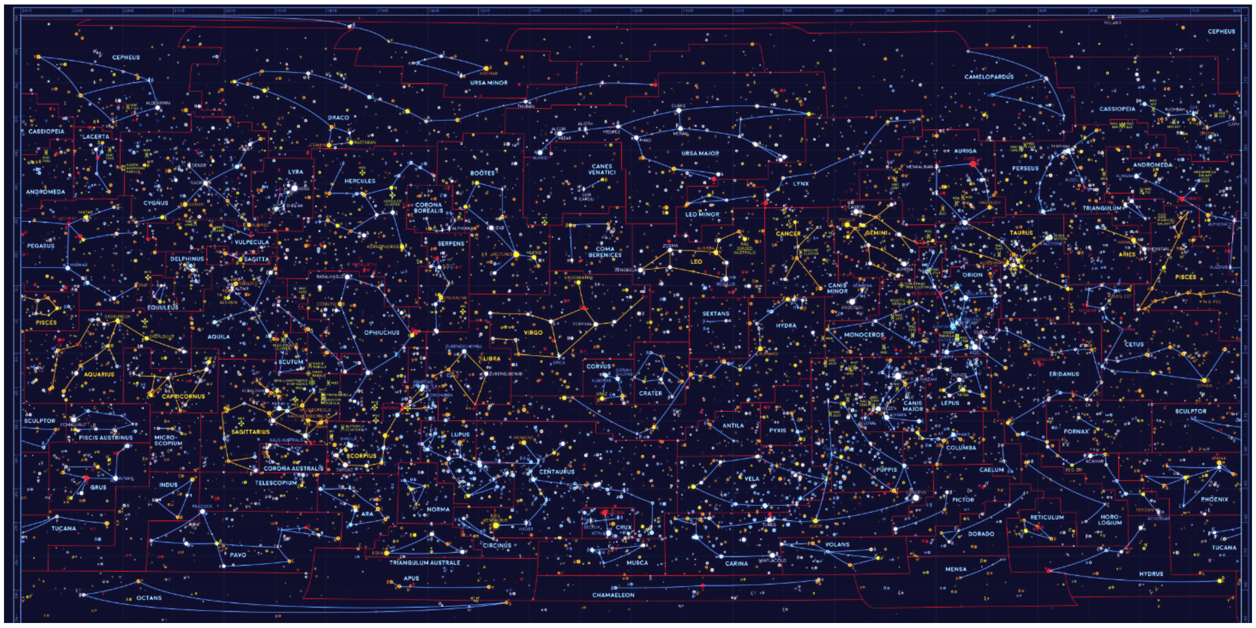

```
In [ ]:
fig = plt.figure(figsize=figsize)
ax = plt.axes(projection=ccrs.PlateCarree(180))

for index, row in const_names.iterrows():
    ax.text(row['ra']*360/24, row['dec'], row['name'],
            transform=ccrs.Geodetic(), ha='left', va='top', fontsize=8, color=nonzodiac_color)

ax.set_xlim(ax.get_xlim()[::-1])
set_save_image(fig, './figures/constellation_names.pdf')
```

Camelopardus
Ursa Minor
Capheus
Draco
Cassiopeia
Ursa Major
Perseus
Lynx
Lacerta
Andromeda
Auriga
Canes Venatici
Lyra
Cygnus
Leo Minor
Triangulum
Boötes
Corona Borealis
Hercules
Vulpecula
Aries
Gemini
Cancer
Coma Berenices
Pegasus
Pisces
Taurus
Leo
Sagitta
Delphinus
Orion
Canis Minor
Aquila
Sextans
Virgo
Ophiuchus
Monoceros
Cetus
Scutum
Crater
Libra
Aquarius
Eridanus
Lepus
Canis Major
Hydra
Corvus
Capricornus
Sagittarius
Sculptor
Fornax
Pyxis
Piscis Austrinus
Columba
Antila
Microscopium
Caelum
Puppis
Scorpius
Corona Australis
Lupus
Grus
Phoenix
Vela
Centaurus
Norma
Telescopium
Pictor
Ara
Indus
Horologium
Reticulum
Carina
Crux
Circinus
Tucana
Dorado
Triangulum Australe
Pavo
Volans
Musca
Hydrus
Apus
Mensa
Chamaeleon
Octans

10. Constellation with boundaries:



By analyzing these plots, we will be doing the following:

- Determine necessary preprocessing steps (resizing, normalization, color balancing, etc.).
- Identify potential issues or biases in your dataset.
- Understand the variability and complexity of your images, which can inform model architecture choices.
- Detecting outliers or unusual images that might need special handling or removal.