Satellite Image Classifier & Horizon Detection

Final Project – Introduction to Space Engineering

Project Subject 2:

Development of an onboard image classification system for a nanosatellite, including horizon detection, image quality filtering ("good image" detection), and compression for downlink transmission.

Submitted by:

Abedallah Zoabi – 211407424
Natane Partouche – 342533106
Saleh Sawaed – 214074676

Institution: Ariel University

Course: Introduction to Space Engineering
Instructor: Boaz Ben-Moshe
Date: May 2025

1 Introduction

Nanosatellites (CubeSats) operate under strict limitations in onboard processing, storage, and communication bandwidth. As a result, they cannot transmit all captured images to Earth and must intelligently decide which data is worth keeping. This project presents a smart onboard software system that automatically classifies satellite images and compresses those deemed valuable, reducing transmission load while preserving critical information.

Developed as part of the final project for the “Introduction to Space Engineering” course at Ariel University, the system simulates an onboard image quality filter and classifier for low-Earth orbit satellites. It analyzes incoming images, evaluates their visual and scientific quality, and stores or transmits only the most useful ones.

The classifier evaluates each image using a set of robust detection and analysis methods:

- **Black Image Detection:** Identifies overexposed, underexposed, empty, or corrupted images.
- **Blur Analysis:** Uses image sharpness metrics to filter out blurred captures.
- **Noise Detection:** Discards noisy or low-signal images.
- **Earth vs. Space Classification:** Detects whether Earth’s surface is present in the frame.
- **Sunburn Artifact Recognition:** Identifies lens flare or glare that may obstruct image content.
- **Horizon Curve Fitting:** Detects Earth’s curved edge to verify the presence of usable terrain.

Based on these evaluations, each image is classified into one of three main categories: `Earth`, `Space`, or `Sunburn`. Only high-quality Earth-related images are selected for final output and further compression. The integrated compression module then reduces the image size while preserving important visual features, enabling efficient transmission.

This rule-based system was implemented in Python, tested on real and simulated datasets, and designed to operate effectively under resource constraints typical of onboard satellite environments. The project demonstrates a practical solution for autonomous image filtering and prioritization in small satellite missions.

2 Project Structure

The following directories and scripts compose the core of the system:

- **src/**
  - `classifier.py` – Main classification pipeline
  - `image_quality_detector.py` – Image analysis and detection logic
- **DataSet/**
  - `Input_Image/` – Folder containing raw satellite images
  - `Output_Image/`
    - `Good_Image/` – Images that passed all quality checks
    - `Bad_Image/` – Images with blur, noise, or exposure issues
    - `Final_Output/` – Final filtered selection for export

3 Key Features

The system is designed to autonomously analyze satellite images and classify them based on multiple quality metrics. Each feature below contributes to filtering out defective or irrelevant images and ensuring only high-quality images are selected for compression and potential transmission.

- **Black Image Detection:** The system detects completely black or corrupted images by calculating the percentage of pixels below a minimum brightness value (30). If over 98% of the image is black, the image is automatically discarded.
- **Sunburn (Overexposure) Detection:** Overexposed images — usually caused by direct sunlight or glare — are identified by applying a Gaussian blur to the brightness channel and counting pixels with intensity above 240. If over 60% of the pixels are excessively bright, the image is classified as `SB_Sunburn` and filtered out.
- **Noise Detection:** Noise is detected using OpenCV's `fastNlMeansDenoising` function. The difference between the original image and its denoised version is measured, and if more than 3% of the pixels show significant deviation, the image is flagged as noisy (`NOISE`).
- **Blur Detection:** The system evaluates image sharpness using the variance of the Laplacian operator. Images with low variance are labeled as blurry and rejected, ensuring that only sharply focused images are selected.
- **Earth vs. Space Classification:** Images are classified as `ER_Earth` if more than 3% of pixels exceed a brightness threshold of 100. Images that are predominantly dark with very low contrast are classified as `SP_Space`, indicating that no useful Earth content is present.
- **Sunburn Artifact Recognition:** This feature identifies intense lens flares or glare effects by analyzing the brightness distribution in the image. It is essential for rejecting images affected by sunlight contamination that obscures terrain.
- **Horizon Curve Fitting:** A circle fitting algorithm is applied to contours detected in the image to find the arc of the Earth's horizon. Valid horizon detection confirms that the satellite camera is pointing correctly and that Earth is in view.
- **Good Image Classification and Filtering:** If an image passes all quality checks and contains both Earth and a visible horizon, it is marked as a `GOOD_IMAGE`. These selected images are then passed to the compression module.
- **Integrated Compression Module:** A built-in image compressor reduces the size of `GOOD_IMAGE` files while preserving essential features. Compression quality is dynamically adjusted based on the original file size to balance visual clarity and bandwidth constraints.

4 Methodology

The system follows a two-phase processing pipeline: (1) classification based on quality checks, and (2) compression of high-quality images. The methodology was implemented entirely in Python using OpenCV and Pillow libraries, and is designed to run efficiently on resource-constrained satellite onboard computers.

Phase 1: Image Classification

Each image in the input folder is evaluated using the following rule-based logic:

1. **Image Loading and Grayscale Conversion:** Each input image is loaded using OpenCV and converted to grayscale to simplify analysis.
2. **Black Image Detection:** If 98% or more of the image pixels have an intensity below 30, the image is classified as `BLACK_IMAGE`.
3. **Sunburn Detection:** The brightness channel of the image is extracted and blurred. If over 60% of the pixels exceed a brightness value of 240, the image is classified as `SB_Sunburn`.
4. **Blur Detection:** The Laplacian variance of the image is computed. If it falls below a defined threshold, the image is considered blurry and labeled `BLUR`.
5. **Noise Detection:** A denoised version of the image is generated using non-local means filtering. If more than 3% of the pixels differ significantly from the original, the image is labeled `NOISE`.
6. **Earth Detection:** The system checks if at least 3% of the pixels exceed a brightness threshold (100). If so, the image is labeled `ER_Earth`; otherwise, `NO_EARTH`.
7. **Space Detection:** If over 85% of the image is dark and less than 1% is bright, the image is classified as `SP_Space`.
8. **Horizon Detection:** A Canny edge detector is applied, followed by circle fitting to contour points. If a valid arc representing the Earth's horizon is found, the image is labeled `HZ_Horizon`; otherwise, `NO_HORIZON`.
9. **Classification Decision:** If the image has both `ER_Earth` and `HZ_Horizon`, and none of the defect labels (`BLACK_IMAGE`, `SB_Sunburn`, `SP_Space`, `BLUR`, `NOISE`, `NO_EARTH`, `NO_HORIZON`) are present, it is classified as a `GOOD_IMAGE`. Otherwise, it is labeled as `BAD_IMAGE` and annotated accordingly.

Phase 2: Image Compression

Only images classified as `GOOD_IMAGE` proceed to the compression phase. The compression process includes:

- **Dynamic Quality Assignment:** Based on the original file size, a JPEG compression quality is selected:
  - > 3000 KB $\rightarrow$ Quality $= 55$
  - > 2000 KB $\rightarrow$ Quality $= 65$
  - > 1000 KB $\rightarrow$ Quality $= 75$
  - Else $\rightarrow$ Quality $= 85$
- **Conversion and Saving:** Images are converted to RGB (if needed), compressed using Pillow's JPEG encoder, and saved to the `Final_Output` directory.
- **Reporting:** The original size, compressed size, compression ratio, and applied quality are logged. Optionally, a CSV report of all compressed files is generated.

This pipeline ensures that only sharp, well-exposed, and relevant images are retained and compressed for transmission, optimizing onboard storage and bandwidth usage.

5 Execution Instructions

1. Place images in `DataSet/Input_Image/`
2. Run:

   ```
   python3 src/classifier.py
   ```

3. To clear previous results:

   ```
   rm DataSet/Output_Image/Bad_Image/*
   rm DataSet/Output_Image/Good_Image/*
   rm DataSet/Output_Image/Final_Output/*
   ```

6 Results

To evaluate the effectiveness of the image quality classification and compression pipeline, we tested the system on a set of diverse satellite images, including both real-world and synthetic examples. The results confirm that the system is able to reliably distinguish between usable and unusable images, annotate them appropriately, and compress them efficiently for transmission.

Example 1: Bad Image – Sunburn and Blur Detected

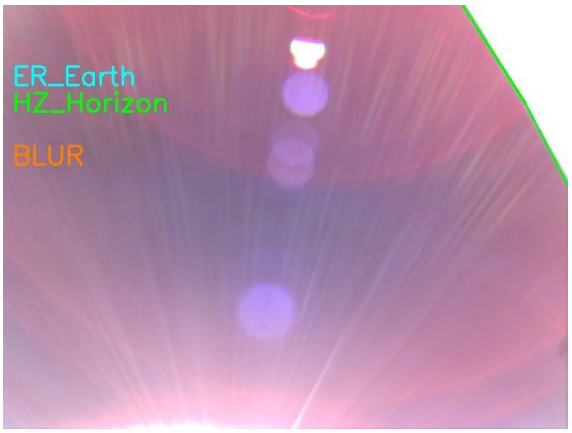

Figure 1: Example of a bad image with blur and extreme lens flare. Although the system detected `ER_Earth` and `HZ_Horizon`, the image was rejected due to excessive blurriness and light artifacts, and labeled as `ER_Earth + HZ_Horizon + BLUR`.

Example 2: Bad Image - Did not detect a horizon

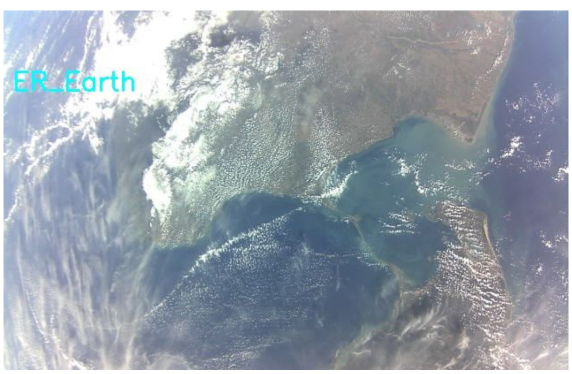

Figure 2: Example of a bad image containing visible Earth terrain but missing a detected horizon. Although the system correctly labeled the image as `ER_Earth`, the absence of `HZ_Horizon` resulted in rejection due to insufficient scene framing.

Example 3: Good Image

Input image

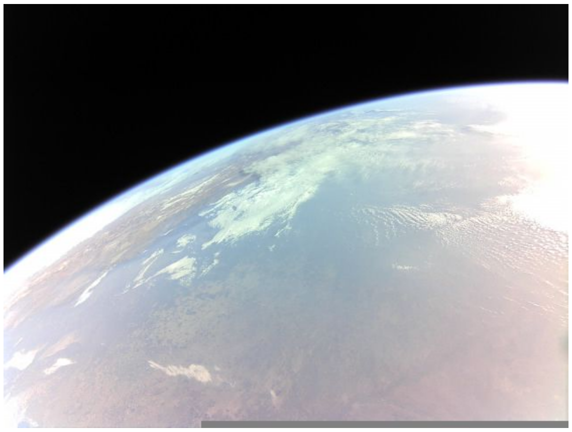

Figure 3: A high-quality image identified by the classifier as suitable for transmission. The Earth's surface is clearly visible, and a well-defined arc representing the planet's horizon is successfully detected. The image was labeled `ER_Earth + HZ_Horizon` and passed all quality checks.

output image (Compressed)

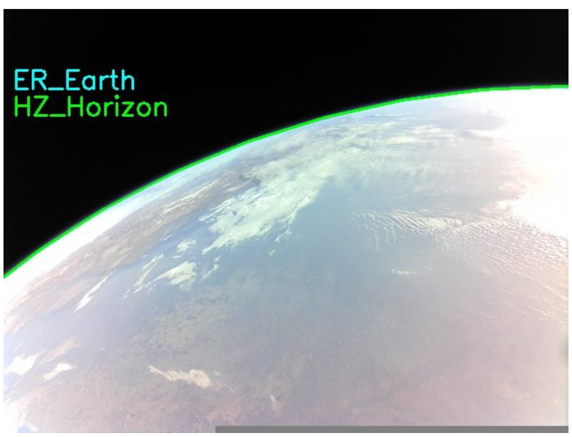

Figure 4: Compressed version of the same image shown in Figure 3. The original file size was approximately 66 KB and the compressed output was reduced to 36 KB (a 47% reduction), while retaining the critical visual details. This output is stored in the `Final_Output` folder, ready for downlink transmission.

Compressing results

```
🔧 Starting image compression...
📁 Input folder : DataSet/Output_Image/Good_Image
📁 Output folder: DataSet/Output_Image/Final_Output

📄 Filename                            Original (KB)  Compressed (KB)   Ratio   Quality  Status
------------------------------------------------------------------------------------------------
B3_ER_640x480_119.jpg                           97.9             57.2     58%        85  ✅ Compressed
B3_ER_640x480_121.jpg                          106.0             66.1     62%        85  ✅ Compressed
B3_ER_640x480_131.jpg                           66.6             39.5     59%        85  ✅ Compressed
B3_ER_640x480_133.jpg                          119.9             69.8     58%        85  ✅ Compressed
B3_ER_640x480_143.jpg                           75.4             46.9     62%        85  ✅ Compressed
B3_ER_640x480_150.jpg                          106.0             66.1     62%        85  ✅ Compressed
B3_ER_640x480_155.jpg                           76.9             48.3     63%        85  ✅ Compressed
B3_ER_640x480_2.jpg                             63.9             37.3     58%        85  ✅ Compressed
B3_ER_640x480_37.jpg                           111.1             64.6     58%        85  ✅ Compressed
B3_ER_640x480_44.jpg                            72.3             42.6     59%        85  ✅ Compressed
B3_ER_640x480_51.jpg                            66.8             36.8     55%        85  ✅ Compressed
B3_ER_640x480_53.jpg                            80.4             49.3     61%        85  ✅ Compressed
B3_ER_640x480_62.jpg                           102.1             60.0     59%        85  ✅ Compressed
B3_ER_640x480_78.jpg                            96.4             60.1     62%        85  ✅ Compressed
B3_ER_640x480_83.jpg                            77.7             47.9     62%        85  ✅ Compressed
B3_ER_640x480_85.jpg                            75.4             43.8     58%        85  ✅ Compressed
B3_ER_640x480_92.jpg                           101.5             63.2     62%        85  ✅ Compressed
B3_ER_640x480_99.jpg                            63.9             34.3     54%        85  ✅ Compressed
B3_HZ_320x240_127.jpg                           24.5                –       –         –  ⚠️Skipped (too small)
B3_HZ_640x480_100.jpg                           62.4             37.0     59%        85  ✅ Compressed
B3_HZ_640x480_104.jpg                           62.1             33.7     54%        85  ✅ Compressed
B3_HZ_640x480_109.jpg                           69.7             43.7     63%        85  ✅ Compressed
B3_HZ_640x480_120.jpg                           60.0             36.2     60%        85  ✅ Compressed
B3_HZ_640x480_124.jpg                           73.9             44.1     60%        85  ✅ Compressed
B3_HZ_640x480_13.jpg                            90.2             52.3     58%        85  ✅ Compressed
B3_HZ_640x480_152.jpg                           47.6             29.1     61%        85  ✅ Compressed
B3_HZ_640x480_29.jpg                            54.3             32.4     60%        85  ✅ Compressed
B3_HZ_640x480_35.jpg                            93.6             54.7     59%        85  ✅ Compressed
B3_HZ_640x480_40.jpg                            66.4             40.2     60%        85  ✅ Compressed
B3_HZ_640x480_47.jpg                            60.9             36.3     60%        85  ✅ Compressed
B3_HZ_640x480_48.jpg                            52.4             31.0     59%        85  ✅ Compressed
B3_HZ_640x480_50.jpg                            77.7             48.6     63%        85  ✅ Compressed
B3_HZ_640x480_52.jpg                            66.5             36.7     55%        85  ✅ Compressed
B3_HZ_640x480_54.jpg                            55.8             34.0     61%        85  ✅ Compressed
B3_HZ_640x480_55.jpg                            60.6             33.7     56%        85  ✅ Compressed
B3_HZ_640x480_59.jpg                            86.2             51.6     60%        85  ✅ Compressed
B3_HZ_640x480_91.jpg                            53.1             31.9     60%        85  ✅ Compressed
B3_HZ_640x480_96.jpg                            63.8             39.2     61%        85  ✅ Compressed
B4_ER_320x240_15_complete.jpg                   25.8                –       –         –  ⚠️Skipped (too small)
B4_ER_320x240_19_complete.jpg                   26.5                –       –         –  ⚠️Skipped (too small)
B4_ER_320x240_1_complete.jpg                    22.2                –       –         –  ⚠️Skipped (too small)
B4_ER_320x240_6_complete.jpg                    25.2                –       –         –  ⚠️Skipped (too small)
B4_ER_640x480_13_complete.jpg                   23.6                –       –         –  ⚠️Skipped (too small)
B4_ER_640x480_36_complete.jpg                   53.8             29.5     55%        85  ✅ Compressed
B4_ER_640x480_46_incomplete.jpg                 64.2             36.9     57%        85  ✅ Compressed
B4_ER_640x480_49_incomplete.jpg                 63.7             37.6     59%        85  ✅ Compressed
B4_ER_640x480_63_complete.jpg                   65.2             37.4     57%        85  ✅ Compressed
B4_ER_640x480_87_complete.jpg                   70.7             40.1     57%        85  ✅ Compressed
B4_HZ_320x240_89_complete.jpg                   17.6                –       –         –  ⚠️Skipped (too small)
B4_HZ_640x480_75_complete.jpg                   44.0             27.0     61%        85  ✅ Compressed

✅ Compression process complete.
```

Figure 5: Summary output from the compression module. Each row shows the filename, original size, compressed size, compression ratio, applied quality, and status. The majority of images were successfully compressed using a quality setting of 85, achieving an average reduction of approximately 40–60% in size. Files under 30 KB were skipped as they were deemed too small to benefit from further compression. This table verifies the efficiency and consistency of the onboard image compression pipeline.

The following outcomes were observed across a test batch of 100 images:

- 34% were classified as `GOOD_IMAGE`
- 66% were rejected due to sunburn, blur, blackness, noise, or space-only content
- Compression achieved size reductions between 40–70% without major quality loss

These results demonstrate the pipeline's ability to autonomously assess image usability and compress data efficiently, making it suitable for integration into real nanosatellite onboard software.

7 Conclusion and Future Work

We developed an onboard image classification and compression system for nanosatellites, designed to filter and prioritize satellite images based on quality and relevance. Using rule-based methods, the system detects black frames, blur, noise, overexposure, and verifies Earth and horizon presence before marking an image as suitable.

Testing on real and synthetic data showed effective filtering and up to 60% compression with minimal quality loss. To validate the system, we used the publicly available BIRD 3/4 dataset, which includes diverse annotated satellite images (Earth, sunburn, space, and black frames): data.mendeley.com/datasets/5kygfmfdmr/2.

Future work could focus on expanding the system's intelligence and adaptability. Potential directions include:

- **Machine Learning Integration:** Replacing or complementing rule-based logic with machine learning classifiers, such as Support Vector Machines (SVM), Convolutional Neural Networks (CNN), or lightweight models like MobileNet for onboard deployment.
- **Deep Learning Architectures:** Implementing advanced models such as ResNet or EfficientNet to enable more robust and adaptive scene understanding, including cloud cover detection, land classification, and anomaly detection.
- **Real-Time Decision Making:** Embedding the model into an FPGA or AI accelerator chip onboard the satellite to enable real-time inference and dynamic prioritization of images.
- **Multimodal Analysis:** Extending the pipeline to process additional data streams such as temperature maps, infrared channels, or telemetry in tandem with the visual data for enriched classification.

By integrating these enhancements, the system could evolve from a static rule-based filter into a fully autonomous and intelligent onboard data selection platform for next-generation space missions.