Abstract

This project presents the **design** and **implementation** of a complete system for acquiring and enhancing real-time Earth imagery transmitted by **NOAA** (**National Oceanic and Atmospheric Administration**) weather satellites—specifically **NOAA 15**, **NOAA 18**, and **NOAA 19**—via their Automatic Picture Transmission **(APT)** signals. Utilizing publicly available **WebSDR** platforms and open-source decoding tools, analog APT signals are captured, processed, and converted into visible satellite images. The decoded images often suffer from **noise** and **signal degradation**, which are mitigated using **image enhancement techniques** such as **Gaussian** and **median filtering**. To further improve visual clarity, advanced methods like the **Hough Transform** are employed to detect and reconstruct corrupted line segments. In addition, **K-means clustering**, is applied to **segment** and **map** key geographic features such as **land**, **water bodies**, and **cloud formations**. The project not only provides hands-on insight into **satellite communication** and **remote sensing** but also demonstrates a **cost-effective approach** for **Earth observation** using publicly accessible technologies. The resulting system lays a foundation for **educational** and **environmental applications**, including **weather monitoring** and **geospatial analysis**.

Contents

List of Figures

Abbreviations

NOAA	National Oceanic and Atmospheric Administration
APT	Automatic Picture Transmission
SDR	Software-Defined Radio
WebSDR	Web Software-Defined Radio
SNR	Signal-to-Noise Ratio
RF	Radio Frequency
FM	Frequency Modulation
AM	Amplitude Modulation
KNN	K-Nearest Neighbors

Chapter 1

INTRODUCTION

1.1 Objective

This project aims to acquire real-time Earth images by capturing analog radio signals transmitted by weather satellites NOAA 15, NOAA 18, and NOAA 19. These signals contain Automatic Picture Transmission (APT) data, which can be decoded into satellite images and to understand how satellites transmits data and to develop a complete system for enhancing NOAA APT satellite images by:

- Receiving and decoding analog APT signals transmitted by NOAA weather satellites using WebSDR and open-source decoding software.
- Enhancing the visual quality of decoded satellite images through **Gaussian and median filtering** to reduce noise and improve detail.
- Identifying regions of noisy lines and using surrounding pixels to reconstruct the image using techniques like **Hough transform**
- Applying **K-means clustering** for landmark segmentation mapping to visually differentiate regions of interest such as land, water and cloud formations.

1.2 Overview of NOAA Satellites

1.2.1 NOAA-15

- **Launch Date:** May 13, 1998
- **Status:** Operational
- **Orbit Type:** Sun-synchronous, near-polar orbit
- **Altitude:** ∼870 km
- **Orbital Period:** ∼102 minutes
- **Transmission Frequency:** 137.620 MHz
- **Instruments Onboard:**
  - AVHRR (Advanced Very High-Resolution Radiometer)

  - AMSU (Advanced Microwave Sounding Unit)

- **Data Type:**
  Transmits analog weather images (visible and infrared). Allows for Earth coverage and weather pattern observation.

1.2.2 NOAA-18

- **Launch Date:** May 20, 2005
- **Status:** Operational
- **Orbit Type:** Sun-synchronous, near-polar orbit
- **Altitude:** ∼854 km
- **Orbital Period:** ∼102 minutes
- **Transmission Frequency:** 137.9125 MHz
- **Instruments Onboard:**
  - AVHRR/3
  - MHS (Microwave Humidity Sounder)
  - HIRS/4 (High Resolution Infrared Radiation Sounder)
- **Data Type:**
  Provides analog weather images (visible and IR). Reliable for real-time imaging with SDR.

1.2.3 NOAA-19

- **Launch Date:** February 6, 2009
- **Status:** Operational
- **Orbit Type:** Sun-synchronous, near-polar orbit
- **Altitude:** ∼870 km
- **Orbital Period:** ∼102 minutes
- **Transmission Frequency:** 137.100 MHz
- **Instruments Onboard:**
  - AVHRR/3
  - AMSU-A, MHS, HIRS/4
  - SEM-2 (Space Environment Monitor)
- **Data Type:**
  Offers high-quality APT imagery, especially suitable for nighttime and IR-based imaging.

1.3 Why We Choose These Satellites for Our Project?

- **Public Access:** No encryption or proprietary access restrictions.
- **Real-Time Broadcast:** Broadcast data in near real-time via APT, suitable for SDR-based reception.
- **Global Coverage:** Due to polar orbits, each satellite provides comprehensive Earth coverage multiple times per day.
- **Historical Utility:** NOAA satellites have been foundational in Earth observation and have consistent documentation, tools, and community support for processing data.

Chapter 2

Signal Reception Setup

2.1 Satellite Tracking and Scheduling

We began the project by tracking NOAA satellites using predictive satellite tracking tools provided by n2yo.com. By utilizing data, we were able to accurately predict satellite passes over our geographic location. This allowed us to schedule audio recording sessions precisely during the time intervals when a satellite was directly overhead, thereby maximizing the quality and reliability of the received data.

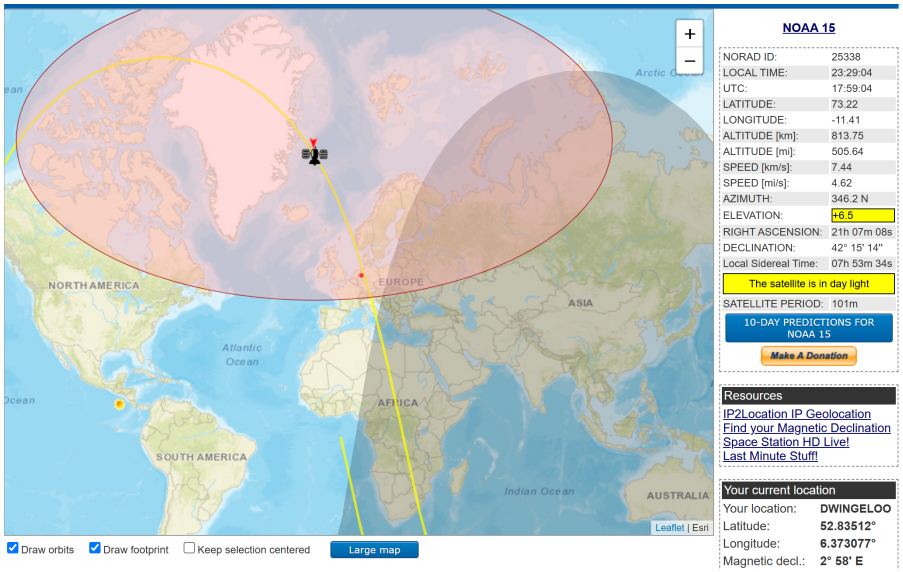


Figure 2.1: Tracking NOAA 15 satellite

2.2 Signal Reception Setup

To receive the analog signal from NOAA satellites, we utilized the WebSDR at websdr.camras.nl:8901. This web-based software-defined radio interface allowed us to remotely tune into the 137 MHz frequency range, which is used by NOAA satellites for APT transmission. The use of WebSDR eliminated the need for a physical SDR setup, enabling reliable signal reception and real-time audio monitoring directly through a web browser.

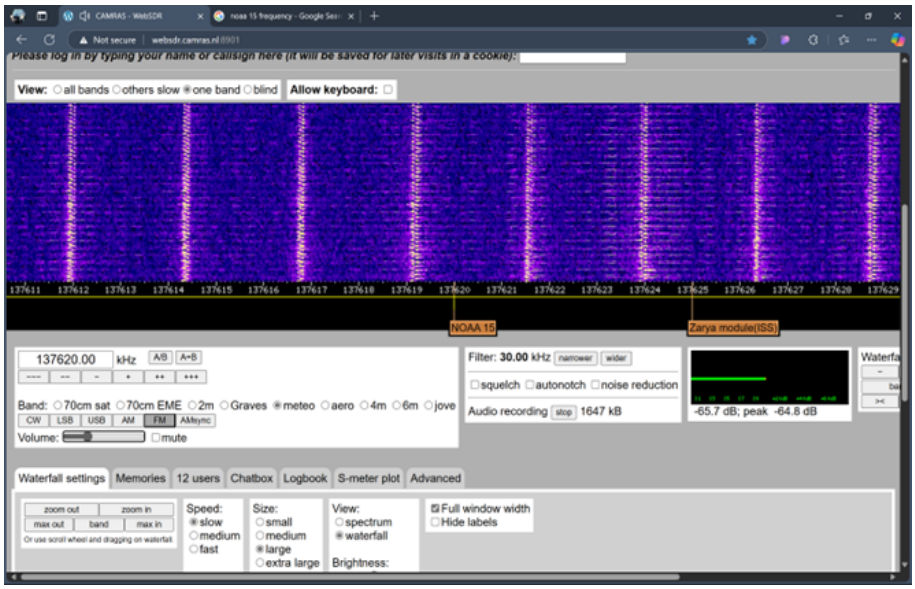


Figure 2.2: Set-up for recording the audio signal of satellite

2.3 Audio Recording and Conversion

The received APT signals were recorded in WAV audio format during the satellite pass. These audio files were later processed using the *Satdump* application, which decoded the analog signals into visible satellite imagery. The decoded images successfully revealed basic meteorological features such as cloud formations, weather patterns, and clear distinctions between land and sea areas, demonstrating the effectiveness of the APT reception and processing workflow.

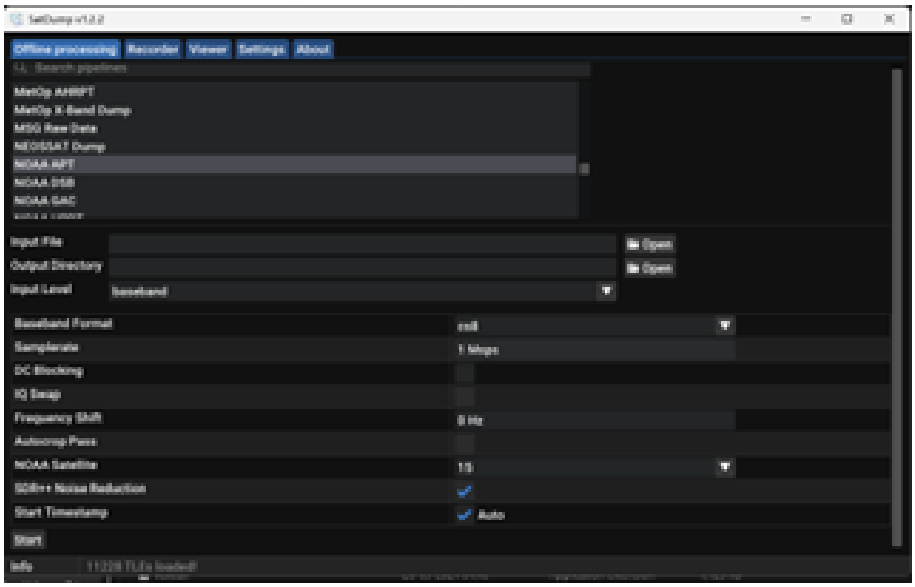
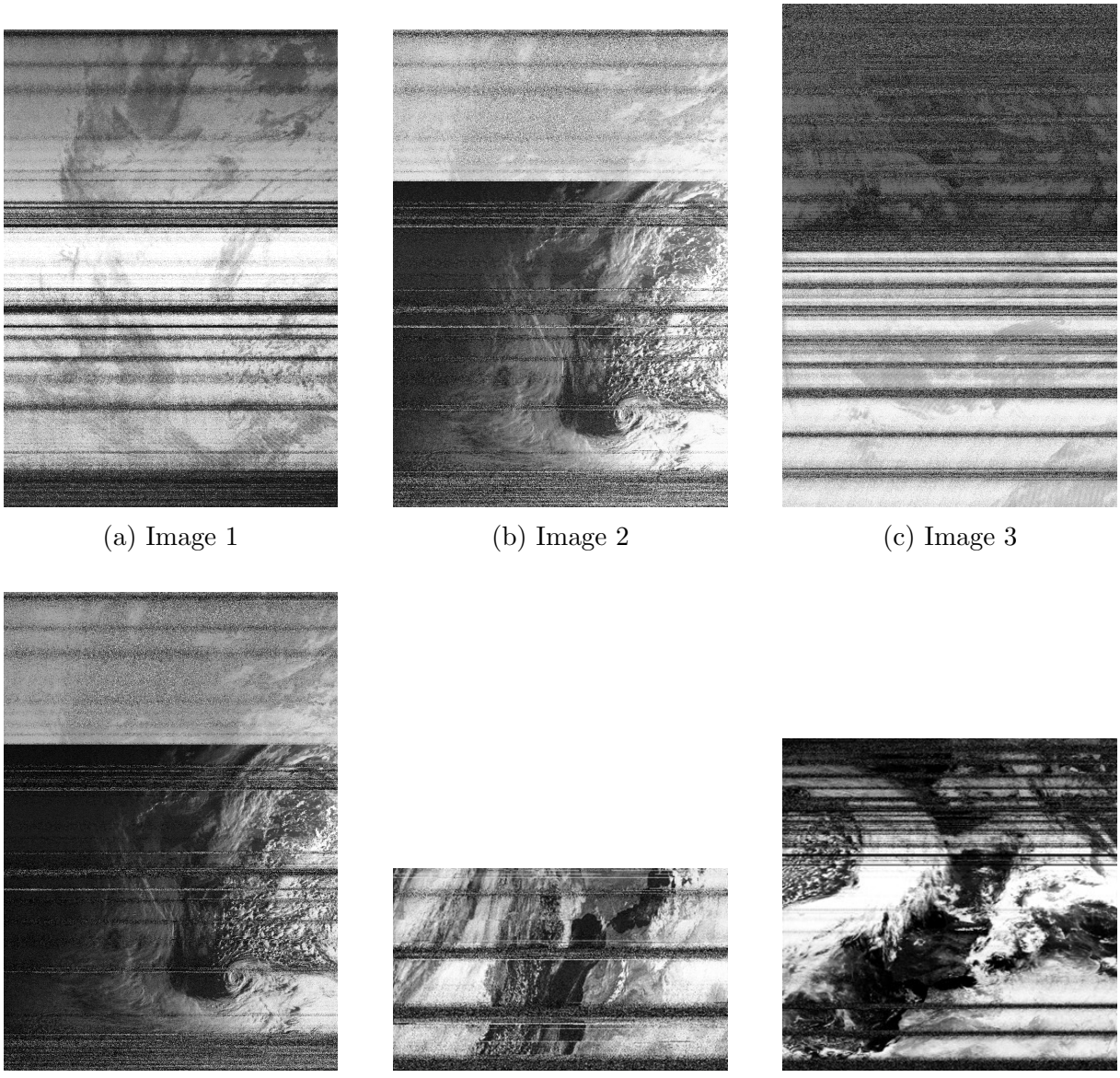


Figure 2.3: Satdump application

2.4 Observations

The clarity of the decoded image heavily depended on signal strength and noise level. Several distortions and noise lines were present in the unprocessed image, highlighting the need for signal and image filtering techniques in the upcoming weeks. The satellite pass had to be precisely timed for effective reception; even a few minutes of error resulted in poor image quality.



(a) Image 1 (b) Image 2 (c) Image 3

(d) Image 4 (e) Image 5 (f) Image 6

Figure 2.4: Decrypted captured satellite images

Chapter 3

Image Enhancement Using Gaussian and Median Filtering

3.1 Image Enhancement Using Gaussian and Median Filtering

Image enhancement is a vital step in image processing aimed at improving the visual quality or analytical clarity of an image. This chapter explores two widely used filtering techniques—Gaussian and Median filters—implemented in MATLAB for denoising and interference removal prior to image reconstruction.

3.2 Introduction

In the real world, signals and images often pick up noise when they are captured or sent. This noise makes it tough to understand or work with them properly. Cleaning up the signal helps a lot with things like finding edges, splitting up image parts, or recognizing objects.

3.3 Gaussian Filtering

The Gaussian filter is a smoothing method used to reduce normal (Gaussian) noise and make an image look softer. It does this by applying a Gaussian pattern to the image, which gently blurs it in a controlled way:

- Reduces fast-changing (high-frequency) noise.
- Keeps the main structure of the image intact.
- Works well for noise that follows a normal (Gaussian) pattern.

Techniques Used

- Gaussian kernel (e.g., 3x3 or 5x5) assigns higher weights to center pixels.
- Convolution is applied between the Gaussian kernel and the image.
- Standard deviation (σ) controls the spread of the filter.

In MATLAB, the function `imgaussfilt()` is used for applying Gaussian filtering:

- `imgaussfilt(I, sigma)` — applies Gaussian filter with specified standard deviation.
- Example: `imgaussfilt(I, 1)` smooths the image with $\sigma = 1$.

3.4 Median Filtering

The median filter is a method used to clean up images by replacing each pixel with the middle value of its neighboring pixels. This helps remove small, unwanted dots or specks, especially the kind known as salt-and-pepper noise. Unlike some filters, it doesn't blur the edges much, so important details in the image are better preserved.

- Keeps edges sharper than a Gaussian filter.
- Effectively removes sudden, sharp noise (like salt-and-pepper noise).
- Helps keep small details in the image clear.

Techniques Used

- A fixed-size neighborhood (usually 3×3 or 5×5) is selected around each pixel.
- All pixel values in this neighborhood are sorted.
- The center pixel is replaced with the **median** of these values.
- This process is repeated for the entire image, resulting in a smoothed image with reduced noise.

In MATLAB, `medfilt2()` is used to apply 2D median filtering:

- Syntax: `medfilt2(I)` — applies a 3×3 median filter to image I.
- You can also specify the neighborhood size, e.g., `medfilt2(I, [5 5])`.

3.5 MATLAB Implementation

Listing 3.1: MATLAB Code for Image Filtering

```
clc;
clear;
close all;

% Reading the image
I = imread("D:\Downloads\Satelite_raw_image - Copy.png");
figure;
imshow(I);
title('Original Image');

% Converting the Image to gray
Igray = rgb2gray(I);
```

```
13
14 % Applied Gaussian Filter
15 J = imgaussfilt(Igray, 3);
16
17 figure;
18 imshow(J);
19 title('Gaussian Filter Image');
20
21 % Applied Median Filter
22 K = medfilt2(Igray, [9, 9]);
23
24 figure;
25 imshow(K);
26 title('Median Filter Image');
```

The filters were applied using MATLAB:

- Synthetic noise was added to test images.
- Both filters were applied independently and jointly.
- Resulting images were visually and quantitatively evaluated.

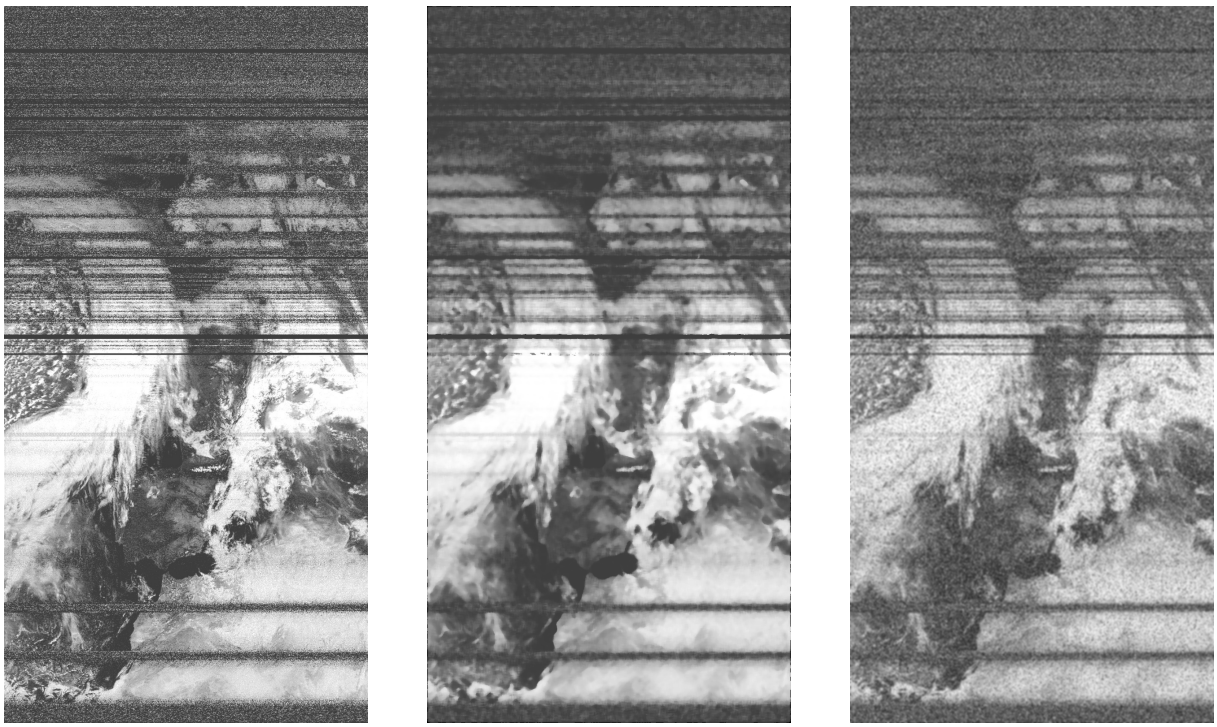
The combined use of filters demonstrated enhanced signal clarity and denoising effectiveness.

3.6 Results and Observations

- Gaussian filter reduced fine-grain noise but slightly blurred edges.
- Median filter preserved edges and removed impulsive noise effectively.
- Combining both filters yielded the best results in terms of visual clarity and noise suppression.

3.7 Conclusion

Gaussian and median filtering are two common techniques used to improve image quality by removing noise. Gaussian filtering smooths the image and reduces noise, while median filtering removes small, sharp noise like speckles. MATLAB makes it easy to apply these filters with powerful functions. After filtering, the image becomes clearer, which helps in accurately reconstructing and processing images for further tasks like object recognition or segmentation.



(a) Original Image (b) Median Filtered (c) Gaussian Filtered

Figure 3.1: Comparison of Filtering Techniques

Chapter 4

Image Reconstruction Using Hough Transform

4.1 Hough Trasnform

Another important part of our image project involved using a technique called the **Hough Transformation.** The Hough transform in its simplest form is a method to detect straight lines, even if those straight lines aren't perfectly drawn or if parts of them are missing or hidden by noise.

4.2 How the Hough Transform Works

4.2.1 Edge Detection

Before detecting lines, the filtered image must be simplified to highlight just the edges.

- This is done using the **Canny edge detector**, which finds strong gradients or rapid intensity changes, potential edges.
- The **Canny edge detector** is a multi-stage algorithm used to detect sharp intensity changes (edges) in an image. It provides precise and clean edge maps with minimal noise.

4.2.2 Transforming to Hough Space (`hough`):

Each edge pixel (x, y) is converted into a set of possible lines that could pass through it. A line in image space can be represented by:

$$\rho = x \cos\theta + y \sin\theta$$

Where:

- ρ is the distance from the origin to the line.
- θ is the angle the line makes with the x-axis.
- (x, y) are the coordinates of a point on the edge.

4.2.3 Creating an Accumulator Matrix

- The Hough Transform uses a 2D matrix (for parameters ρ and θ). Each edge pixel in the image votes for all possible lines it could lie on. These votes are recorded in the accumulator matrix.
- Peaks (local maxima) in this matrix represent lines strongly supported by many edge points.

4.2.4 Detecting Prominent Lines (`houghpeaks`):

- After the voting process, the algorithm looks for peaks (local maxima) in the accumulator matrix. Each peak indicates that many edge pixels lie along a certain line — meaning a strong, straight line exists there.

4.2.5 Recovering Lines (`houghlines`):

- These peaks are converted back into visible line segments in the image using their angle and distance parameters (θ, ρ).

4.2.6 Suppressing or Removing Lines:

- The detected lines represent noise. So instead of highlighting them, we overwrite (draw over) them to remove them using (`insertShape`).

4.2.7 Filling in the Gaps (`regionfill`):

- Removing lines creates empty gaps. These are filled in using nearby pixel values with `regionfill`, which helps in smoothly reconstructing those parts.

4.3 MATLAB Code for Image Reconstruction

Listing 4.1: MATLAB code for Image Reconstruction using Hough Transform

```
clc;
clear all;
close all;

% Read the image
Img = imread('Final_image_after_filters.jpg');
if size(Img,3) == 3
    I = rgb2gray(Img);     % Convert to grayscale
else
    I = Img;               % alredy in grayscale
end

% Edge Detection
edges = edge(I, 'Canny', [0.1 0.25]);

%Morphological operations to connect lines
se = strel('line', 7, 0); % Line-shaped structuring element (horizontal)
```

```
edges = imdilate(edges, se);
edges = imerode(edges, se);

% Hough Transform
[H, theta, rho] = hough(edges);

% Find Hough Peaks
numPeaks = 150;
peaks = houghpeaks(H, numPeaks, 'Threshold', 0.20 * max(H(:)));

% Find lines from Hough peaks
lines = houghlines(edges, theta, rho, peaks,'FillGap', 40, 'MinLength',20);

% Select only near-horizontal lines
horizontalLines = [];
angleThreshold = 10;  % Accept lines within  10  degrees of horizontal (0
    degrees)

for k = 1:length(lines)
    angle = atan2d((lines(k).point2(2) - lines(k).point1(2)), (lines(k).
        point2(1) - lines(k).point1(1)));
    if abs(angle) <= angleThreshold
        horizontalLines = [horizontalLines; lines(k)];
    end
end

% Reconstruct only the horizontal lines
reconstructedImage = zeros(size(I), 'logical');
for k = 1:length(horizontalLines)
    xy = [horizontalLines(k).point1; horizontalLines(k).point2];
    reconstructedImage = insertShape(uint8(reconstructedImage)*255, 'Line',
        [xy(1, :) xy(2, :)], 'Color', 'white', 'LineWidth', 2);
    reconstructedImage = im2bw(reconstructedImage);
end

% Fill broken parts of horizontal lines
reconstructedImage = imdilate(reconstructedImage, strel('line', 40, 0)); %
    Strengthen horizontally
reconstructedImage = imerode(reconstructedImage, strel('line', 20, 0));  %
    Restore thickness

% Blend reconstructed lines with original image
finalImage = max(I, uint8(reconstructedImage)*255);

% Display results
figure;
imshow(I), title('Final image after filters');
figure;
imshow(edges), title('Edges Detected (after preprocessing)');
figure;
imshow(reconstructedImage), title('Reconstructed Only Horizontal Lines');
figure;
imshow(finalImage), title('Final Cleaned Image');
```

4.4 Results

- **Final image after filters:** This figure displayed input image , which you loaded from 'Final image after filters'. This image is already the result of previous filtering.
- **Edges Detected:** This figure showed the output of edge detection (Canny) and morphological operations. We can see that the process successfully detected many edges, particularly highlighting the horizontal features which were likely enhanced by the horizontal structuring element used in the morphology. These edges are what the Hough Transform will use to find lines.
- **Reconstructed Horizontal Lines:** This was the reconstructedImage. As analyzed before, this image showed only the horizontal lines that were successfully detected by the Hough Transform and then redrawn as solid white lines onto a black background. The morphology that we applied made these lines appear thicker and more solid. We could see several distinct horizontal segments. The number and location of these lines depend directly on the Hough parameters (‘`numPeaks`‘, ‘`Threshold`‘, ‘`FillGap`‘, ‘`MinLength`‘) and threshold angle (‘`angleThreshold`‘).
- **Final Cleaned Image:** This was the ‘`finalImage`‘, calculated using ‘`finalImage = max(I, uint8(reconstructedImage)*255);`‘. As expected from the ‘`max`‘ operation, this image showed the original image with the solid white lines from image (c) overlaid on top. Wherever a pixel was white (1) in ‘`reconstructedImage`‘, the corresponding pixel in ‘`finalImage`‘ was 255 (white), replacing the original pixel data in those locations.

4.5 Observation

- **Successful Detection:** The code successfully detected a set of near-horizontal lines in the image, as evidenced by the clear lines shown in image (c).
- **Final Output:** The result in Final Cleaned Image is the original image with these solid white lines drawn over it. This effectively highlights the detected lines by obscuring the original image data in those areas with white. This is not a 'cleaned' image in the sense of noise removal that preserves the underlying image features.
- **Parameter Influence:** The specific number, length, and position of the white lines in Reconstructed Horizontal Lines and Reconstructed Vertical Lines are a direct result of the parameters chosen for edge detection, Hough Transform peak finding, line extraction, and the filtering for horizontal lines. Adjusting these would change which lines are detected and subsequently drawn.

(a) Final image after filters

(b) Edges Detected

(c) Reconstructed Horizontal Lines

(d) Final Cleaned Image

Figure 4.1: Result Of Hough Transform

4.6 Conclusion

Our code effectively uses Hough Transform to identify horizontal lines, but the final step is drawing these lines onto the original image rather than cleaning the original noisy lines. The output clearly shows the original image with the detected horizontal lines overlaid in white.

Chapter 5

Landmark Mapping Using K-Means Clustering

5.1 Theory

After filtering and cleaning, the satellite image is enhanced by mapping key landmarks and adding false colours, turning a plain black-and-white NOAA image into a vibrant, informative map.

5.1.1 LandMark Mapping:

Raw NOAA images display land, water, mountains, and cities in similar shades of gray, making it difficult to distinguish features like coastlines, urban areas, or forests. To improve clarity, landmark mapping is used to automatically identify and label these regions.

5.1.2 K-Means Clustering:

K-means Clustering is an unsupervised machine learning algorithm used to partition data into K distinct non-overlapping clusters. It minimises the variance within each cluster by assigning data points to the nearest centroid.

5.1.3 Working:

- Choose the number of clusters (K).
- Initialize centroids: Randomly select K points as initial centroids.
- Assign data points to the nearest centroid using a distance metric (typically Euclidean distance).
- Update centroids by calculating the mean of all points assigned to each cluster.
- Repeat steps 3–4 times until centroids no longer change significantly.
- The false colouring is then applied to label the geographical features, avoiding doing it manually:
  - Cluster 0: Deep ocean
  - Cluster 1: Land
  - Cluster 2: Cloud cover
  - Cluster 3: Vegetation regions

5.2 MATLAB Code for K-Means Clustering

5.2.1 K-Means Clustering

Listing 5.1: K-Means Clustering

```
% Clear workspace/command window
clc;
clear;
close all;

% Import Image %
I=imread("I.jpeg");
figure;
imshow(I);
title("Reconstructed Image");
hold on;
% K Means Clustering %
[nrows,ncols]=size(I);
pixels=I(:);

k=4;

[idx,~]=kmeans(pixels,k);
kcluster_I=reshape(idx,nrows,ncols);

kcluster_norm=(kcluster_I-1)/(k-1);

figure;
imshow(kcluster_norm);
title('K-Means Clustered Image');
```

5.2.2 False Coloring

Listing 5.2: False Coloring

```
%False Coloring %
false_colors=[0 0 1; 0.71, 0.40, 0.11; 1 1 1; 0 0.6 0.1];
final_I=ind2rgb(kcluster_I, false_colors);

labels={'Ocean','Land','Clouds','Vegetation'};

legend_height=3;
legend_width=30;
legend_img=ones(legend_height,legend_width,3);

num_clusters=size(false_colors,1);
block_width=floor(legend_width/num_clusters);

for i=1:num_clusters
    x_start=(i-1)*block_width+1;
    x_end=i*block_width;
    for c=1:3
```

```
        legend_img(:, x_start:x_end, c)=false_colors(i,c);
    end
end

figure;
subplot(2,1,1);
imshow(final_I);
title('K-Means Clustered Image');

subplot(2,1,2);
imshow(legend_img);
title('Cluster Legend');

Add text labels under the color blocks
hold on;
for i=1:num_clusters
    x_pos=(i-1)*block_width + block_width/2;
    text(x_pos, legend_height + 5, labels{i}, 'HorizontalAlignment','center
        ', ...
         'FontSize', 10, 'FontWeight', 'bold', 'Color', 'k');
end
hold off;
```

5.2.3 Results

- The grayscale clustered image is displayed, showing four distinct regions.
- Each region corresponds to a class (e.g., land, water, clouds, vegetation) based on pixel intensity groupings.
- Without false coloring, segmentation is visible but lacks interpretability (colors improve clarity).
- The following legend has been successfully shown by the Output Image.
  - Ocean - Blue
  - Land - Light Brown
  - Clouds - White
  - Vegetation - Green

Figure 5.1: Reconstructed Image

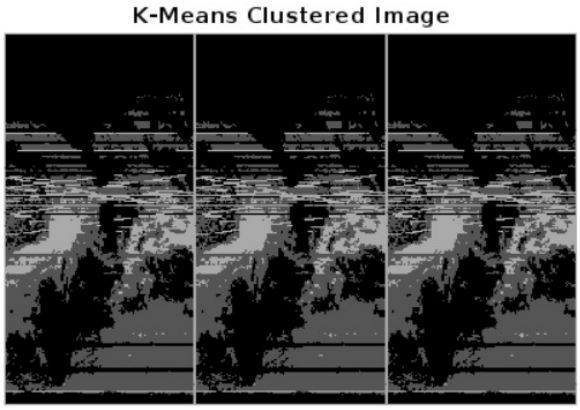
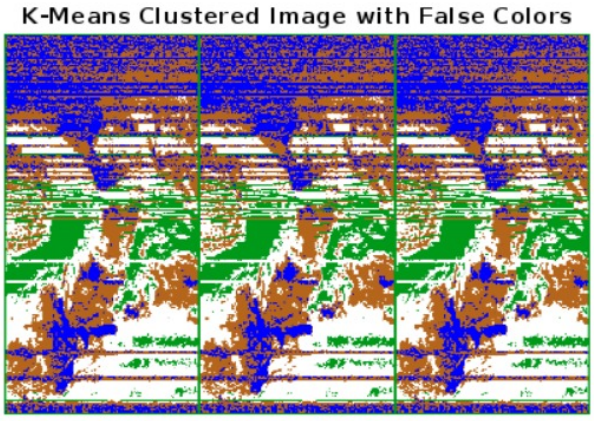
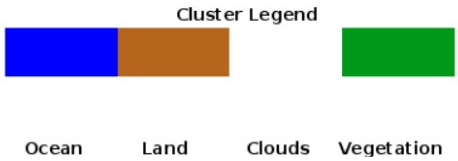


Figure 5.2: K-Means Clustered Image



(a) K-Means Clustered Image With False Colors



(b) Color Legend

5.3 Conclusion

- The application of K-means clustering with k = 4 shows effective separation of distinct regions in satellite imagery, enabling segmentation into meaningful geographical categories. However, when using grayscale images, visually similar classes, such as clouds and snow, or vegetation and land, may appear too close in tone, making interpretation less intuitive.
- As K-means relies solely on pixel intensity, it does not account for textural or spatial context, which limits its ability to distinguish complex patterns. Still, the method demonstrates a simple yet powerful way to analyze satellite images without the need for supervised learning or labeled data.
- In conclusion, this project successfully highlights how unsupervised K-means clustering can be leveraged for basic satellite image analysis. While effective, the approach can be further improved by:
- In future iterations, the system could benefit from:
  - Utilising colour or multispectral data for richer, more distinct segmentation,
  - Incorporating texture or contextual features for better accuracy,
  - And automating cluster labelling using external geographic references.

Chapter 6

Conclusion

6.1 Summary of Findings

- This project successfully demonstrated the end-to-end workflow of receiving, decoding, and visualizing NOAA satellite APT signals using WebSDR. Through precise satellite tracking, WebSDR tuning, and Satdump-based decoding, real-time analog weather imagery was extracted and analyzed. Key meteorological features such as cloud patterns and land-water boundaries were clearly visible in the decoded images, validating the effectiveness of the reception pipeline.
- Furthermore, post-processing techniques such as image filtering and noise reduction were applied to enhance the visual clarity of the decoded satellite images. Reconstruction algorithms were used to restore partially corrupted image segments, thereby improving the continuity and usability of the data.
- Finally, K-means clustering was employed to segment the images based on pixel intensity distributions. This allowed for a simplified classification of regions such as land, water, and cloud cover, contributing to a more structured understanding of meteorological patterns captured in the satellite passes.

Innovations and Uniqueness

- **Hardware-Free Signal Acquisition:** Most APT projects rely on SDR hardware; this project demonstrates a hardware-free method using WebSDR streams, making satellite imaging accessible without physical investment.
- **Advanced Post-Processing:** Standard NOAA APT processing typically ends at basic decoding. This project extends the workflow by applying Gaussian and median filters to denoise and enhance image features such as coastlines and cloud edges.
- **Unsupervised Image Segmentation:** Clustering algorithms are rarely applied to APT images. The use of K-means clustering for unsupervised segmentation introduces analytical depth and enables geographic feature isolation beyond mere visual inspection.
- **Interdisciplinary Integration:** Unlike projects that focus solely on signal reception, this work integrates signal processing, image analysis, and unsupervised machine learning, making it valuable for both academic demonstration and future research applications.

Chapter 7

Future Scope

While the current implementation offers a functional prototype for analog signal reception and image generation, several enhancements can significantly improve the robustness and capability of the system:

- **Automation:** The satellite pass prediction, signal recording, and image processing stages can be automated using scripting tools and scheduled jobs, enabling a continuous and unattended ground station setup.
- **Machine Learning Integration:** ML models can be used to enhance image quality, denoise satellite images, and classify weather phenomena, providing deeper analytical insights.
- **High-Resolution Reception:** Transitioning from APT to HRPT (High-Resolution Picture Transmission) reception using more advanced SDRs and antennas can dramatically increase image detail, allowing for more refined meteorological assessments.
- **Georeferencing and Landmark Mapping:** Future work can include aligning satellite images with real-world coordinates and overlaying political boundaries and city names, transforming raw data into actionable geospatial maps.

This foundational work provides a scalable path forward toward building an intelligent, real-time satellite image analysis system with research and operational applications.

Bibliography

[1] Dennis Roddy, *Satellite Communications*, 4th Edition, McGraw-Hill, 2006.

[2] WEBSDR Project, "Web Software Defined Radio," http://www.websdr.org, Accessed April 2025.

[3] NOAA Satellite and Information Service, https://www.nesdis.noaa.gov/, Accessed April 2025.

[4] SatDump, "NOAA APT Signal Decoding Tool," https://github.com/SatDump/SatDump, Accessed April 2025.

[5] Andrew Barron, *An Introduction to Software Defined Radio*, ARRL Publications, 2018.

[6] M. HemaLatha; S. Varadarajan; Y. Murali Mohan Babu, "Comparison of DWT, DWT-SWT, and DT-CWT for low resolution satellite images enhancement" in *2017 International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET) , Chennai India*10.1109/ICAMMAET.2017.8186662

[7] Vineet Vilas Naik , Saylee Gharge, "Satellite image resolution enhancement using DTCWT and DTCWT based fusion" in *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Jaipur, India 10.1109/ICACCI.2016.7732338

[8] Vaishnavi Kharat, Sanyukta Khatdeo, Harshada Kothe, Rutuja Kshirsagar, Mrudul Dixit, and M. Selva Balan, "Land Cover Clustering and Classification of Satellite Images," in *Proceedings of [conference name or journal if known]*, Cummins College of Engineering for Women and CWPRS, Pune, India.