COMPUTER VISION : CLOUDS SEGMENTATION

PERNA Luca, LANGLOIS Henri, FREDJ Zacharie

March 2025

Abstract

This project focuses on cloud segmentation in satellite imagery using classical computer vision techniques. The goal is to accurately identify and segment cloud regions from multi-spectral satellite images, which is crucial for applications in meteorology, climate research, and environmental monitoring. We explored various non-deep-learning methods, including thresholding and watershed segmentation. We also used supervised methods like Linear Discriminant Analysis (LDA), Gaussian Naive Bayes (GNB), and histogram-based approaches to compare the effectiveness of traditional computer vision techniques with supervised learning methods. The methods were evaluated using precision, recall, F1 score, Jaccard Index, and accuracy metrics. Our findings, restituted through plots and metrics, demonstrate the effectiveness of classical techniques in achieving reliable cloud segmentation, offering a transparent and computationally efficient alternative to deep learning methods.

Github repository : https://github.com/HenriLD/CloudDetection

1 Introduction

This project centers on the segmentation of clouds in satellite imagery using classical computer vision techniques. The primary problem we aim to solve is the accurate identification and segmentation of cloud regions from multi-spectral satellite images. This task is crucial for various applications in meteorology, climate research, and environmental monitoring. By accurately segmenting clouds, we can improve the analysis of land cover, vegetation health, and urban development, which are often obscured by cloud presence. The questions we seek to answer include: How can classical computer vision techniques effectively segment clouds portions from satellite images? How do these methods compare to supervised learning approaches in terms of accuracy and computational efficiency? The importance of this problem lies in its potential to enhance the precision of environmental analyses, which are vital for informed decision-making

in climate science and resource management.

The objective of this project is to generate a binary mask from a multi-spectral satellite image, where each pixel is classified as either cloud (1) or non-cloud (0). Formally, given an input image I with spectral bands $\{R, G, B, NIR\}$, we aim to produce a binary mask M such that $M(x, y) \in \{0, 1\}$ for each pixel coordinate (x, y). The goal is to maximize the accuracy of this mask, ensuring that cloud regions are correctly identified while minimizing false positives and negatives. Constraints include the use of classical computer vision techniques without relying on deep learning models, which often require extensive computational resources and large labeled datasets. The hardness of the problem arises from the variability in cloud appearances, including diverse shapes, textures, and brightness levels, as well as the potential confusion with similar surface features such as snow and ice. Additionally, changes in lighting conditions and atmospheric effects can further complicate the segmentation task, especially on the outlines of the clouds, necessitating robust and adaptable methods.

2 Related Work

1) Cloud detection methodologies : Variants and development- a review by S. Mahajan et. B. Fataniya
https://link.springer.com/article/10.1007/s40747-019-00128-0

2) A morphology-based approach for cloud detection by S. Danda, A. Challa and B.S. Daya Sagar
https://ieeexplore.ieee.org/document/7729011

3) Effective could Detection and segmentation using a gradient-based algorithm for satellite imagery : Application to improve PERSIANN-CCS by N. Hayatbini et. al.
https://arxiv.org/abs/1809.10801

4) A Cloud Detection Method for Ladsat 8 Images Based on PCANet by Y. Zi, F. Xie and Z. Jiang
https://www.mdpi.com/2072-4292/10/6/877

5) A Cloud Detection Algorithm for Remote Sensing Images Using Fully Convolutional Neural Networks by S. Mohajerani, T. Krammer and P. Saeedi
https://ieeexplore.ieee.org/document/8547095

3 Methodology

Our methodology for cloud segmentation in satellite imagery integrates classical computer vision techniques with supervised learning methods to achieve accurate and efficient segmentation. Below is a detailed explanation of each method used, including their functionality and mathematical background.

3.1 Data Collection

We used the "38-Cloud" dataset created by S. Mohajerani et al.[**4**], which consists of 38 Landsat 8 scenes decomposed into 17601 non-overlapping patches (each of shape 384*384). The training pack contains 8400 images, and the testing pack contains 9201 images. Each image contains four spectral bands: red, green, blue, and near-infrared (NIR), along with per-pixel ground truth masks for evaluation. They are in a Tiff format, and a pixel can represent between 15 and 100m on the Earth's surface. This dataset provides a rich source of multi-spectral information necessary for effective cloud segmentation.

3.2 Otsu's Method

Otsu's method is a widely used technique for automatic threshold selection in image segmentation, particularly effective for images with bimodal histograms. The method aims to find the optimal threshold that maximizes the inter-class variance between the background and foreground pixels. This is achieved by iterating over all possible threshold values and calculating the inter-class variance for each threshold. The threshold that yields the highest inter-class variance is selected as the optimal threshold.

3.2.1 Mathematical Formulation

1. Compute the histogram of the image to determine the frequency of each pixel intensity value.

2. Class Probabilities and Means: For a given threshold t, the pixels are divided into two classes: C_0 (background) and C_1 (foreground). To calculate the probabilities of each class:

$$\omega_0(t) = \sum_{i=0}^{t} p(i), \quad \omega_1(t) = \sum_{i=t+1}^{L-1} p(i)$$

where $p(i)$ is the probability of intensity i, and L is the total number of intensity levels. Calculate the means of each class:

$$\mu_0(t) = \sum_{i=0}^{t} \frac{i \cdot p(i)}{\omega_0(t)}, \quad \mu_1(t) = \sum_{i=t+1}^{L-1} \frac{i \cdot p(i)}{\omega_1(t)}$$

3. Inter-class Variance: The inter-class variance $\sigma_B^2(t)$ is calculated as:

$$\sigma_B^2(t) = \omega_0(t) \cdot \omega_1(t) \cdot (\mu_0(t) - \mu_1(t))^2$$

The optimal threshold t^* is the value of t that maximizes $\sigma_B^2(t)$.
Otsu's method is particularly useful in our project for detecting and segmenting

clouds from satellite images because it automatically determines the best threshold for separating cloud pixels from the background. This is crucial given the variability in cloud appearances and intensities across different spectral bands.

3.2.2 Implementation in our code

- **Normalization :** Normalize each spectral band to ensure consistent intensity ranges.
- **Histogram Computation:** Compute the histogram for each band to analyze the distribution of pixel intensities.
- **Threshold Calculation:** Apply Otsu's method to determine the optimal threshold for each band.
- **Segmentation:** Use the calculated thresholds to create binary masks that segment clouds from the background.
- **Visualization:** Plot histograms with the computed thresholds to visually assess the segmentation results.

3.3 Watershed Segmentation

Watershed segmentation is a morphological technique used for image segmentation, particularly effective for images where objects have distinct boundaries. The method treats the image as a topographic surface, identifying "catchment basins" and "watershed lines" to delineate regions based on gradient magnitudes. This approach is especially useful for segmenting clouds in satellite imagery due to their varying shapes and sizes.

3.3.1 Mathematical Formulation

1. Preprocessing: Apply a Gaussian blur to reduce noise and enhance gradient information. The Gaussian blur can be represented as:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

where σ is the standard deviation that determines the extent of smoothing.

2. Thresholding: Use a thresholding method (e.g., Otsu's method) to create a binary image that identifies potential foreground and background regions.

3. Distance Transform: Apply a distance transform to the binary image to create a "height map," where each pixel's value represents its distance to the nearest background pixel. The distance transform $D(x, y)$ for a binary image $B(x, y)$ is defined as:

$$D(x, y) = \min \sqrt{(x - x')^2 + (y - y')^2}$$

where (x', y') are the coordinates of the background pixels.

4. Marker Identification: Identify markers from the distance transform, which are local maxima representing potential cloud centers.

5. Watershed Transform: Apply the watershed algorithm using these markers to segment the image into distinct regions. The watershed transform assigns each pixel to a catchment basin based on the gradient flow. Mathematically, the watershed transform can be represented as:

$$W(x, y) = \arg \min_{m \in M} \text{dist}(P, m)$$

where $W(x, y)$ is the label assigned to pixel (x, y), M is the set of markers, and $\text{dist}(P, m)$ is the distance from pixel P to marker m.

6. Post-processing: Refine the segmentation by assigning labels to each region and creating a binary mask where clouds are highlighted.

The watershed method is beneficial for our project because it effectively delineates cloud boundaries, which can vary significantly in shape and size. This adaptability makes it robust for diverse satellite imagery.

3.3.2 Implementation in our code

- **Preprocessing:** Apply Gaussian blur to the image to smooth noise and enhance gradients.
- **Binary Image Creation:** Use Otsu's method to create a binary image that distinguishes potential cloud regions from the background.
- **Distance Transform:** Compute the distance transform to identify potential cloud centers.
- **Marker Definition:** Define markers based on the distance transform to guide the watershed algorithm.
- **Watershed Application:** Apply the watershed algorithm to segment the image into distinct regions.
- **Refinement:** Post-process the segmented regions to create a binary mask that accurately represents cloud coverage.
- **Evaluation:** Compare the resulting mask with the ground truth to evaluate the performance of the segmentation.

3.4 Gaussian Naive Bayes (GNB)

Gaussian Naive Bayes (GNB) is a probabilistic classifier based on applying Bayes' theorem with strong independence assumptions between features. It assumes that the continuous values associated with each class are distributed according to a Gaussian distribution.

3.4.1 Implementation in our code

- **Data Preparation:** Sample a subset of images and load multispectral bands. Normalize each band and flatten the features.
- **Statistics Calculation:** Compute the mean and variance for each feature in each class.
- **Log-Likelihood Calculation:** Calculate the log-likelihoods for each class using the Gaussian distribution.
- **Prediction:** Classify each pixel based on the maximum log-likelihood.
- **Error Analysis:** Compute error masks to evaluate the performance of the GNB segmentation.

3.5 Decision Tree

Decision Trees are non-parametric supervised learning models that recursively split the feature space based on feature thresholds to classify data. In our project, the decision tree minimizes the Gini impurity at each split to optimally distinguish cloud pixels from non-cloud pixels using multispectral features. The Gini impurity is defined as:

$$G = 1 - \sum_{i=1}^{K} p_i^2,$$

With p_i the proportion of samples in a given node that belong to class i. In our binary case, this simplifies to:

$$G = 1 - p^2 - (1-p)^2 = 2p(1-p)$$

3.5.1 Implementation in our code

- **Data Preparation:** Sample a subset of images and load multispectral bands, normalize each band, and flatten the pixel features.
- **Label Assignment:** Convert the ground truth masks into binary labels (cloud or non-cloud) for each pixel.
- **Training:** Train a decision tree classifier (using scikit-learn's `DecisionTreeClassifier`) with the default Gini impurity criterion to learn the optimal splits that best separate the classes.
- **Prediction:** Use the trained decision tree to classify each pixel in a test image, generating a binary segmentation mask.
- **Error Analysis:** Compute error masks by comparing the predicted mask against the ground truth to assess the performance of the decision tree segmentation.

4 Evaluation

For evaluating the efficiency of the different methods for segmentation described just before, we have decided to settle on 5 metrics :

- The **precision** is defined by the number of True Positives (TP) divided by the total number of Positive Predictions (True Positives and False Positives (FP)) : $P = \frac{TP}{(TP+FP)}$
- The **recall** is defined by the number of True Positives (TP) divided by the total number of actual positives in the dataset (TP and False Negatives (FN)) : $R = \frac{TP}{(TP+FN)}$
- The **F1-score** computes the harmonic average of precision and recall, and represents the global performance of a classifier : $F1 = \frac{2*P*R}{P+R}$
- The **Jaccard index** is defined by the number of TP divided by the sum of TP, False Positives (FP) and False Negatives (FN). This represents a measure of similarity between the set of predicted positives and the set of actual positives : $J = \frac{TP}{TP+FP+FN}$
- The **accuracy** is defined by the number of total positive predictions divided by the total number of samples. It describes how often a model correctly predicts the outcome :$A = \frac{TP+TN}{TP+TN+FP+FN}$

To begin with, we took 1000 samples from the test dataset. Each method will be evaluated on these 5 metrics and we take their averages over all samples to compare efficiency and understand whether the models are over or underconfident. Here are our findings :

```
Average Metrics over 1000 test images:
GNB:              Precision: 0.7222, Recall: 0.6452, F1: 0.6252, Jaccard: 0.5465, Accuracy: 0.7651
Histogram-based:  Precision: 0.6874, Recall: 0.7449, F1: 0.6597, Jaccard: 0.5970, Accuracy: 0.7686
Otsu's Method:    Precision: 0.7117, Recall: 0.5512, F1: 0.5549, Jaccard: 0.4532, Accuracy: 0.6815
Decision Tree:    Precision: 0.6812, Recall: 0.7765, F1: 0.6775, Jaccard: 0.6232, Accuracy: 0.7935
```

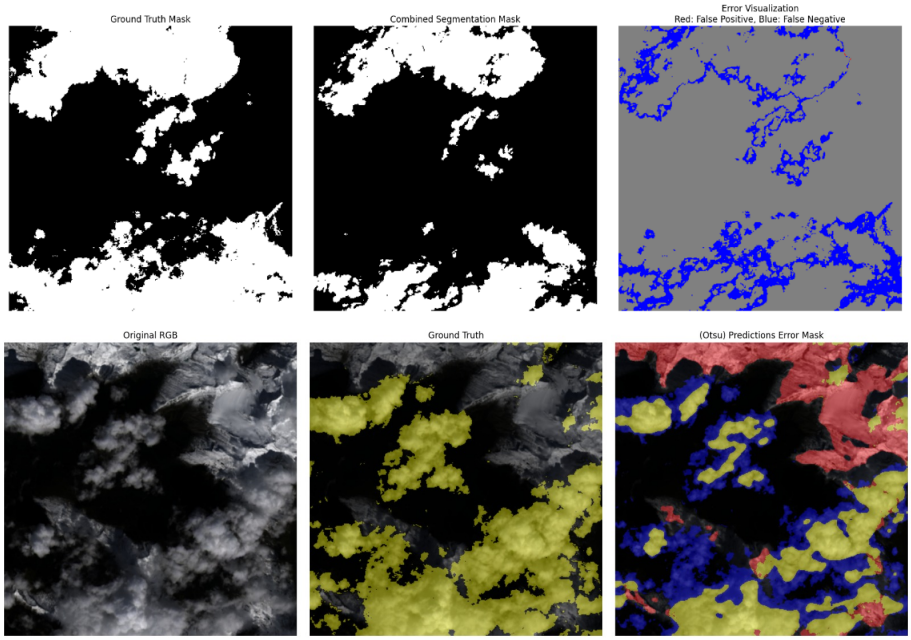
```
Average Metrics with Denoising over 1000 test images:
GNB (denoised):      Precision: 0.7258, Recall: 0.6374, F1: 0.6219, Jaccard: 0.5435, Accuracy: 0.7656
Histogram-based (denoised):  Precision: 0.6906, Recall: 0.7425, F1: 0.6587, Jaccard: 0.5962, Accuracy: 0.7682
Otsu's Method (denoised):    Precision: 0.7142, Recall: 0.5420, F1: 0.5477, Jaccard: 0.4458, Accuracy: 0.6796
Decision Tree (denoised):    Precision: 0.6909, Recall: 0.7812, F1: 0.6839, Jaccard: 0.6328, Accuracy: 0.7961
```

Among the classical methods, Otsu's method and GNB, while having good precision, have pretty average recall, demonstrating the model is very reluctant to predict presence of clouds (underconfidence). This can be seen in the mask visualizations. The histogram-based approach and the decision tree demonstrated strong performance, with a pretty good balance between precision and recall while slightly overconfident. They also have the best F1-score and Jaccard index. The most probable conjecture is that GNB and Otsu's underconfidence

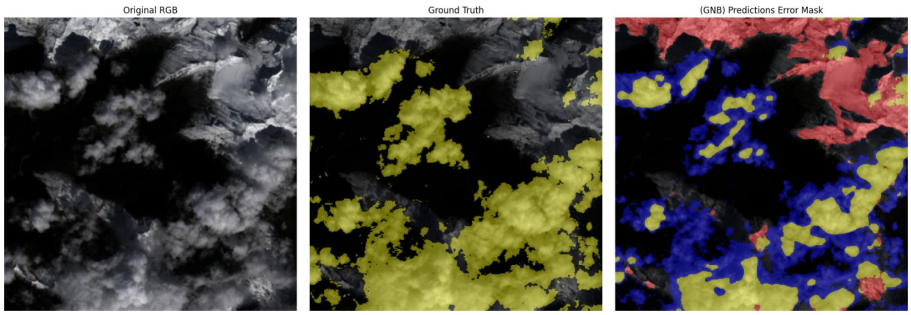
are very detrimential in the lens of our metrics, especially because of the clouds' contours.

But to better visualize what were the strengths and weaknesses of each method, especially the outlines regions and smaller clouds, and when there are snow regions underneath. Here are some examples :

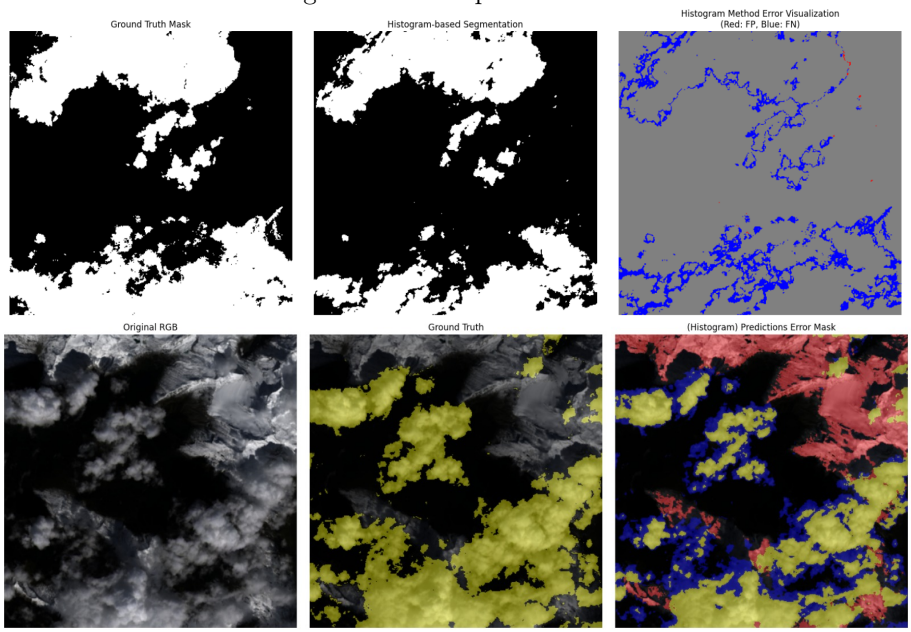
Otsu's Method



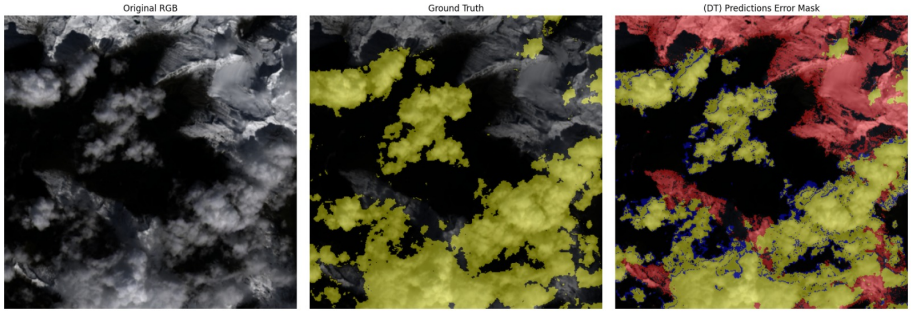
Gaussian Naive Bayes



Histogram-based Supervised Method



Decision Tree



5 Conclusion

In this project, we explored cloud segmentation in satellite imagery using classical computer vision techniques and supervised learning approaches. Through our evaluation, we observed notable differences in performance across methods, with some excelling in precision while others performed better in recall.

While no single method emerged as universally superior, our findings highlight that classical techniques can still provide reliable cloud segmentation without the computational overhead of deep learning models. Future work could explore hybrid techniques that mesh classical segmentation and lightweight learning-based refinements to further improve robustness and adaptability. These approaches could be particularly useful for environments with less resources.

References

[1] Otsu, N. Automatic threshold selection based on discriminant and least squares criteria. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

[2] Beucher, S., & Lantuéjoul, C. Use of watersheds in contour detection. In *International Workshop on Image Processing: Real-Time Edge and Motion Detection/Estimation*, pages 17–21, 1992.

[3] Duda, R. O., Hart, P. E., & Stork, D. G. *Pattern Classification*. John Wiley & Sons, 2001.

[4] Sorour Mohajerani. 38-Cloud - Cloud Segmentation in Satellite Images. *Kaggle*, 2021. Available at: `https://www.kaggle.com/sorour/38cloud-cloud-segmentation-in-satellite-images`

[5] Danda, S., Challa, A., & Daya Sagar, B. S. (2016). A morphology-based approach for cloud detection. 2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS),