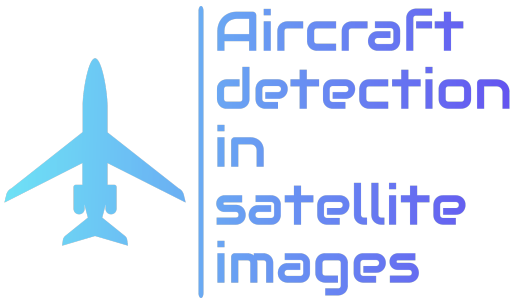
Aircraft detection in satellite images

Giuseppe Alfieri, Riccardo D'Aguanno, Gianmarco Luongo, Aurora Pisa, Paolo Simeone

Università degli studi di Cassino e del Lazio Meridionale



Abstract

This report provides an in-depth overview of a project dedicated to the recognition of aircraft from satellite images. It elaborates on the key design choices and methodologies employed throughout the project's development.
The workflow of the project includes several crucial stages: template extraction (initial extraction of template features from satellite images), image clustering (use of the K-Means clustering algorithm to group similar image segments), calculation of average airplane (derivation of the average airplane template from clustered image data to enhance detection accuracy) and airplanes detection (application of the calculated average airplane template to identify and classify aircraft within new satellite images).

1 Introduction

In this paper we introduce our project whose aim is to implement a system capable of detecting the presence of aircraft in satellite images.
Nowadays, aircraft detection from satellite images is used for air traffic monitoring, military surveillance, search and rescue of missing aircraft, border control and much more.
Satellite images represent a rich and increasingly accessible source of data, providing a global overview of vast terrestrial areas. However, accurate aircraft detection depends on various factors such as atmospheric conditions variations, aircraft orientation and the presence of noise.
This report explains the approach adopted in our project to address these issues through the use of advanced image processing and analysis techniques.
For this project, we used the **OpenCV** library and the **C++** programming language.

1.1 Workflow

3.1 Template extraction
3.2 K-Means algorithm for image clustering
3.3 Calculation of average airplane
3.4 Airplanes detection
3.5 Performance evaluation

2 Dataset description

We used the **HRPlanesv2** dataset, a set designed specifically for aircraft detection and recognition in high-resolution satellite images.
The HRPlanesv2 dataset contains 2120 Google Earth VHR images. To compose the dataset, im-

ages of airports from different geographical locations around the world with various uses (such as civil, military and joint) were selected and labeled. A total of 14.335 aircraft were tagged. Each image is stored as a ".jpg" file measuring 4800 x 2703 pixels, and each label is stored as a YOLO ".txt" format (each aircraft in the images is annotated with bounding boxes, specifying the precise location and dimensions of the aircraft in the image).
The dataset includes images captured over different periods and a variety of challenging scenarios such as partially occluded aircraft, different orientations, and varying lighting conditions to ensure robust model training.

2.1 Data distribution

The dataset is divided into three parts: 70% for training, 20% for validation, and 10% for testing. The aircraft in the images within the training and validation datasets are 80% or larger in size.
In particular, the dataset is made up of three sets:

- **Train set**, it consists of 1490 images and 9734 airplanes;
- **Validation set**, it consists of 420 images and 2555 airplanes;
- **Test set**, it consists of 210 images and 2046 airplanes.

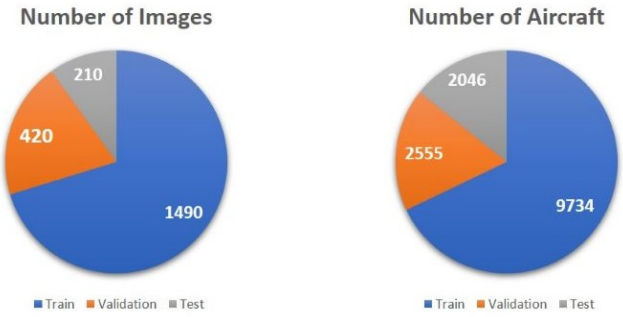


Figure 1. Illustration of data distribution.

3 Methodology

3.1 Template extraction

In this section, the template extraction will be described in detail because templates are necessary for the subsequent processes.
For template extraction in our project, we used 1700 images containing 11780 aircraft (210 images from the test set and 1490 images from the training set). For each image, the corresponding ".txt" file is read line by line because each line contains bounding box information for aircraft in YOLO format. The normalized coordinates of each bounding box are converted to pixel coordinates, creating rectangles to define the regions of interest in the images. Each object contained within the rectangles (every airplane in each image) is displayed to the user, who can decide whether to process the plane or not.

3.1.1 Binarization step

For each ROI selected by the user, the V channel is extracted from the HSV representation of the image (because this channel contains useful intensity information for binarization), and an adaptive threshold using the Gaussian method is applied to it.
Then, a morphological dilation is performed using an elliptical kernel: this helps to fill any gaps and smooth out the contours of the object, improving the resulting binary image.

3.1.2 Calculation of geometric moments

During the calculation of geometric moments, the contours of objects in the binarized images are identified. Using these moments the centroid of the object, which represents the central point of its mass, and the orientation of the object with respect to the image's reference axis are determined. Then, a ROI (from which the template will be extracted) is delimited around the object based on the calculated centroid and correct orientation.

Through manual selection by the user, the aircraft templates are chosen and saved in a specific folder.
At the end of these processes, **7925 aircraft templates** with different shapes and sizes with

the tip pointing upwards were selected.

Figure 2. Some of the extracted templates.

3.2 K-Means algorithm for image clustering

Following the template extraction, we implement two variants of k-means for image clustering: one based on image dimensions and the other based on pixel intensity.

3.2.1 K-Means by images size

The k-means by size accepts a vector of extracted templates and a desired number of clusters (in our project k = 5). It computes the dimensions (width and height) of each image in the vector and uses these dimensions as features to perform k-means clustering. The resulting five clusters are organized based on the sizes of the images, with each cluster containing images of similar dimensions.

Figure 3. Some of the templates clustered by size.

3.2.2 K-Means by images intensities

This step takes a vector of images already grouped based on their dimensions and a desired number of clusters (in our project k = 6). It calculates the average pixel intensity of each image and uses these intensities as features to perform k-means clustering. The result is six clusters group images with similar pixel intensity characteristics.
The grayscale was used because eigenfaces works better in grayscale: this is because PCA reduces the dimensionality of the data and converting the images to grayscale simplifies the data, reducing the computational complexity.

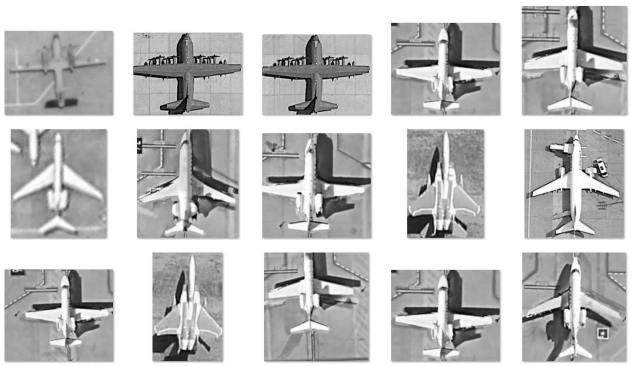


Figure 4. Some of the templates clustered by intensity.

At the end of these processes we obtained thirty clusters.

3.2.3 Uniformity of clusters

To ensure uniformity in dimensions within each cluster, we resized every image. We found the average dimensions among the images in each cluster and resized all images to these dimensions. This operation is essential for the subsequent steps.

3.3 Calculation of average airplane

In this work, we propose our approach based on Principal Component Analysis (PCA) to obtain average representations of airplanes from clusters of images for its application in template matching.
The first phase of our method involves converting a set of images into a data matrix suitable for PCA. Next, we applied PCA to the data matrix to extract

the principal components and calculate the clusters average image.
Our 'eigenPlanes function' uses OpenCV's PCA, retaining 95% of the total variance. Eigenvectors play a crucial role in this process, identifying the directions in which the data varies the most and contributing to the transformation of the original data into the principal component space. At the end of this step we have achieved **30 average airplanes**.

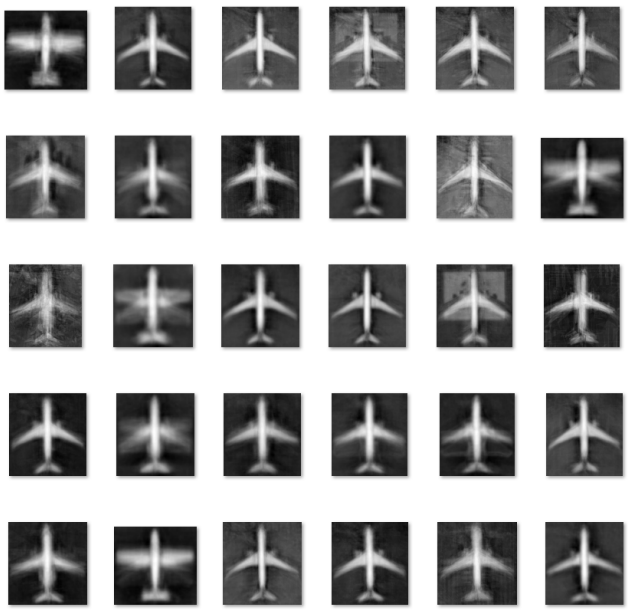


Figure 5. The 30 average airplanes.

3.4 Airplanes detection

3.4.1 Template matching

We used template matching to find matches between the images of average airplanes and the airplanes in a source image.
First of all, we rotated the souce image at various angles in accord to the images of average airplanes to ensure we can find matches no matter how the source image is oriented. The rotation was centered on the image and included adjustments to the bounding rectangle to encompass the entire rotated image. After this step, template matching identified in the rotate image the best match position and the matching point was transformed into the original coordinates. This process was repeated for each specified angle.
To make the process faster, we implemented template matching using multiple threads: each thread operated on different angles and models simultaneously. The results from all threads were consolidated into a single list of match points.
In the main function, we loaded the average models, performed multithreaded template matching and filtered match points to remove those that were too close to each other.
At the end of the template matching, we obtained the points with the highest correlation.

3.4.2 Hog features extraction

For the SVM training we used hog features extraction.
Our code implements a process for extracting and writing HOG (Histogram of Oriented Gradients) features from an image.
First, an object is defined and configured to work with a detection window of 64x64 pixels (to keep feature vectors at a reduced size), cells of 8x8 pixels, blocks of 8x8 pixels with a stride of 8x8 pixels, and a histogram bin count of 9. Next, for each ROI defined by the input rectangles, the corresponding image is extracted, resized to 64x64 pixels, and HOG features are calculated and added to a vector of vectors of floats.
Additionally, the code writes the extracted HOG features to a specified CSV file.
After opening the file, the HOG features are written in CSV format, ensuring that each value is separated by a comma and formatted to a fixed precision of six decimal places. This process ensures that the features are written in a consistent and readable format, regardless of locale settings.

3.4.3 Classification and labeling of ROI for SVM training

This step begins with loading the grayscale images from the dataset and reading the corresponding YOLO labels. Next, **HRPlatemplate matching** is performed to identify points of interest in the

images, which are then classified based on their position relative to the YOLO bounding boxes. Regions of interest are generated for true positives by selecting those with the highest Intersection over Union (IoU) with the YOLO bounding boxes, while for false positives, random sizes are chosen from pre-calculated average dimensions. HOG features are extracted for all identified ROIs.
For true positives, the ROIs are extracted from areas with the highest IoU relative to the YOLO bounding boxes, while for false positives, random areas are selected that do not overlap with the true positives or the YOLO bounding boxes.

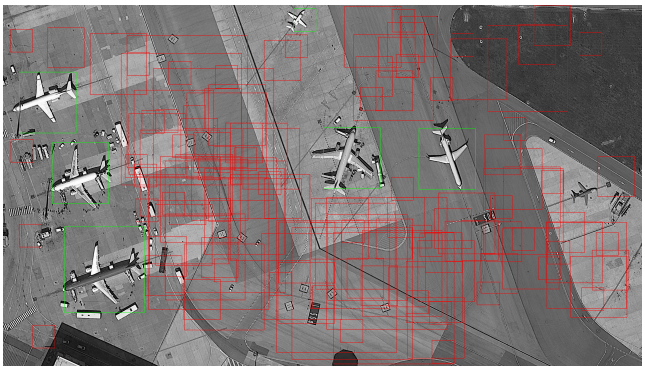
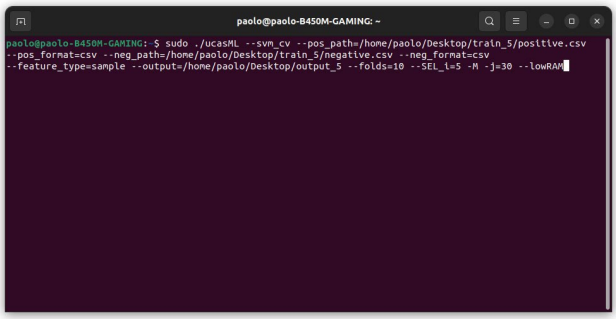


Figure 6. ROIs extracted from true positives (green) and ROIs extracted from false positives (red).

Finally, the HOG features are saved in separate CSV files for true positives and false positives, ensuring a balanced and structured representation of the data for effective machine learning model training. This process guarantees that the HOG features are ready for further analysis or applications, providing a well-organized dataset useful for SVM model training.

3.4.4 SVM Cross-Validation

We used cross-validation for the SVM model, providing input files in CSV format derived from the processing of **50 images**. This allows us to evaluate the model's performance using different subsets of the data, helping to prevent overfitting.



3.5 Performance evaluation

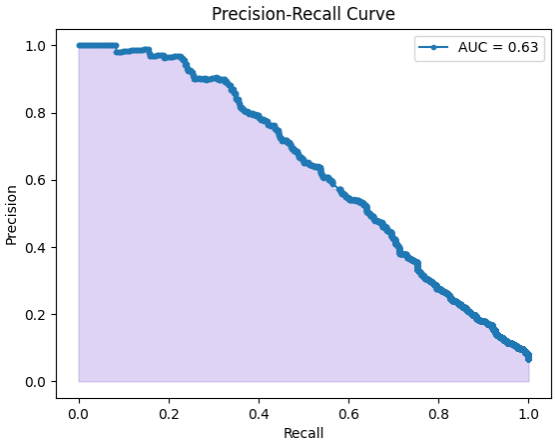


Figure 7. Precision recall curve.