CMPEN 454 Project 1 Group 3 Report

Team Members: Brian Nguyen, Kyle Bradley, Jordan Reed, Drice Bahajak

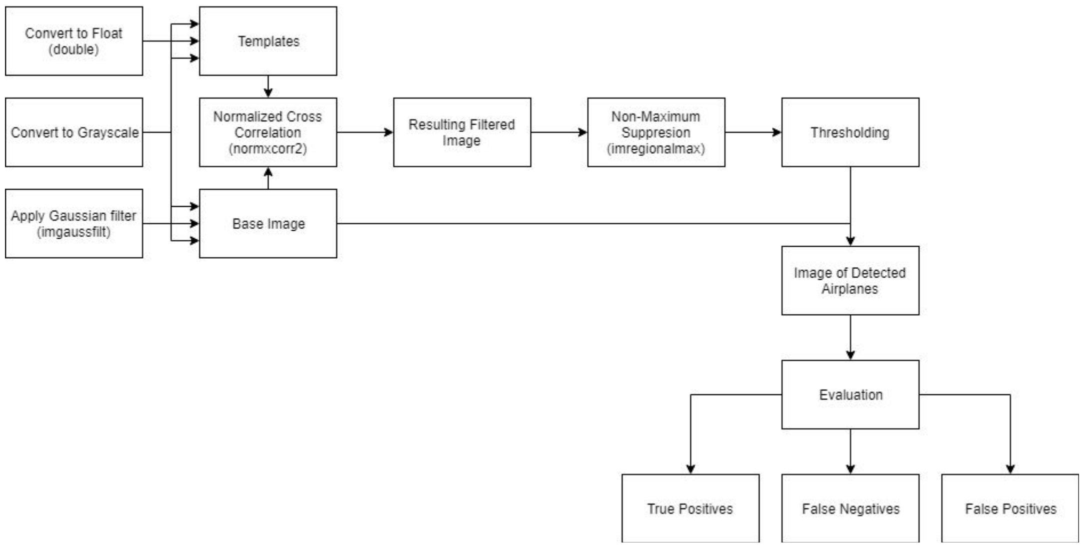
Date: 09/21/2018

Summary

The purpose of this project was for us to collaboratively work as a team to gain hands-on experience with computer vision tasks. The main goal of this particular project was to be able to detect objects (airplanes) from the use of processes we learned in class. To get to that goal, we created suitable templates for the different types of objects we had to handle, such as planes of varying size, style and direction of planes. Before using these templates with the base image, we used pre-processing techniques such as converting to grayscale, converting to float, and applying a Gaussian filter to reduce noise. We then cross-correlated the templates with the base image to get a filtered image. From the resulting image, we found the regional maximums and detected areas of pixels that exceeded our threshold to identify those points as an airplane. From using these processes, we wished to achieve reasonably high true positives and low false positives and negatives.

Approach

Below is a flowchart and explanation of our implementation of the airplane detector model.


Convert to Float (double)
Convert to Grayscale
Apply Gaussian filter (imgaussfilt)
Templates
Normalized Cross Correlation (normxcorr2)
Base Image
Resulting Filtered Image
Non-Maximum Suppresion (imregionalmax)
Thresholding
Image of Detected Airplanes
Evaluation
True Positives
False Negatives
False Positives

- **Creating Templates**
  - Each of our templates were taken as cropped images of the base image. We chose templates based on the different sizes, styles, and directions of planes, which basically aligned with the different areas of planes. In total, we ended up with only two plane templates, which makes our solution pretty general.
- **Convert to Float** (double())
  - By converting to float, we avoid computation errors such as when converting to grayscale.
- **Convert to Grayscale** ((im(:,:,1)+im(:,:,2)+im(:,:,3))/3)
  - By converting to grayscale, we are dealing with single pixel values instead of RGB values. This means less calculations for any filtering we perform on the image.
- **Gaussian Filter** (imgausfilt())
  - As we learned in class, a Gaussian filter is a center-weighted filter. This is important to use in our context because we want the center of the planes to be weighted so that we have high pixel values in the center for easy detection. The result is a blurred image, which heavily helped with plane detection. We experimented with many values of sigma, but since our templates were small (and decreased in size using imresize()), we found it best to use small values of sigma, between 1 and 3 depending on the template
- **Cross Correlation** (normxcorr2())
  - Since we are using a Gaussian filter, we chose to use a normalized cross correlation. By doing this, we will have our filtered image ready for non-maximum suppression since pixel areas with planes in them should have high pixel values.
- **Non-maximum Suppression** (imregionalmax())
  - By using this function, we are checking to see if areas of pixels are maxes in their regions (at its lowest level, determining local extrema). If they are, their pixel values stay the same, if not they become zero. This is so we can reduce our scope for the high pixel values for easier thresholding.
- **Thresholding**

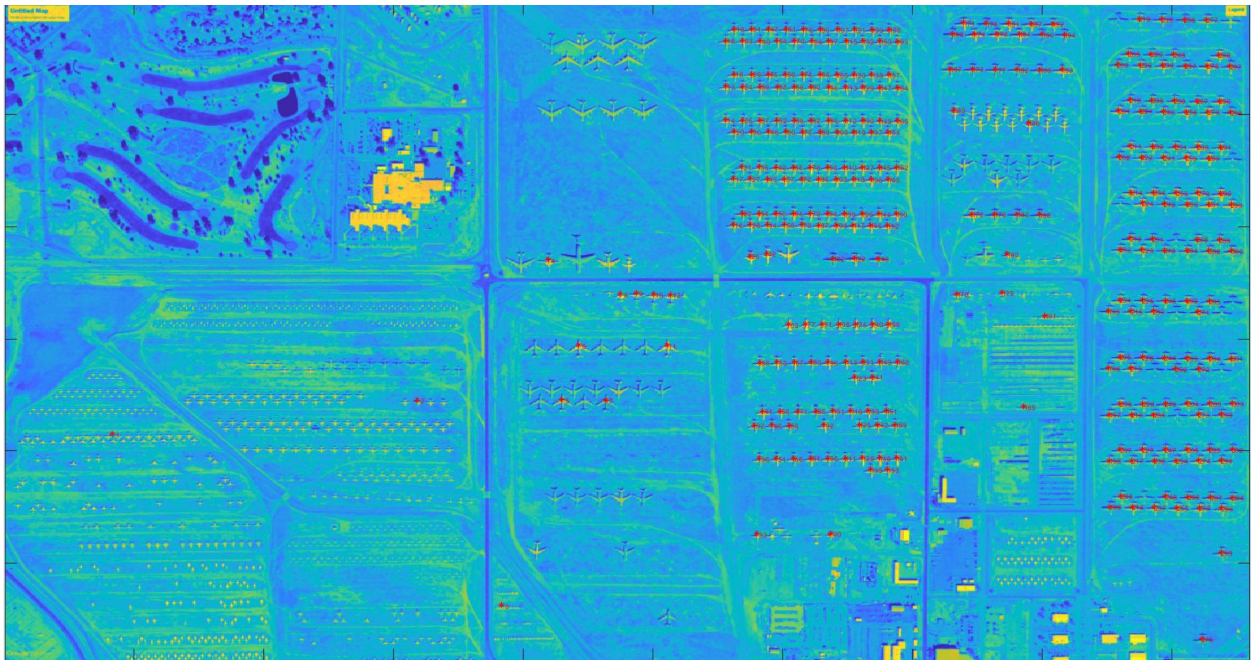
    - By iterating through the pixel values of the image, we check to see when the pixel values exceed our threshold, so that we label those points as a detected plane. Through trial and error, we chose threshold values of 0.4 or 0.5, since those values gave us the best results for each template respectively.

Observations

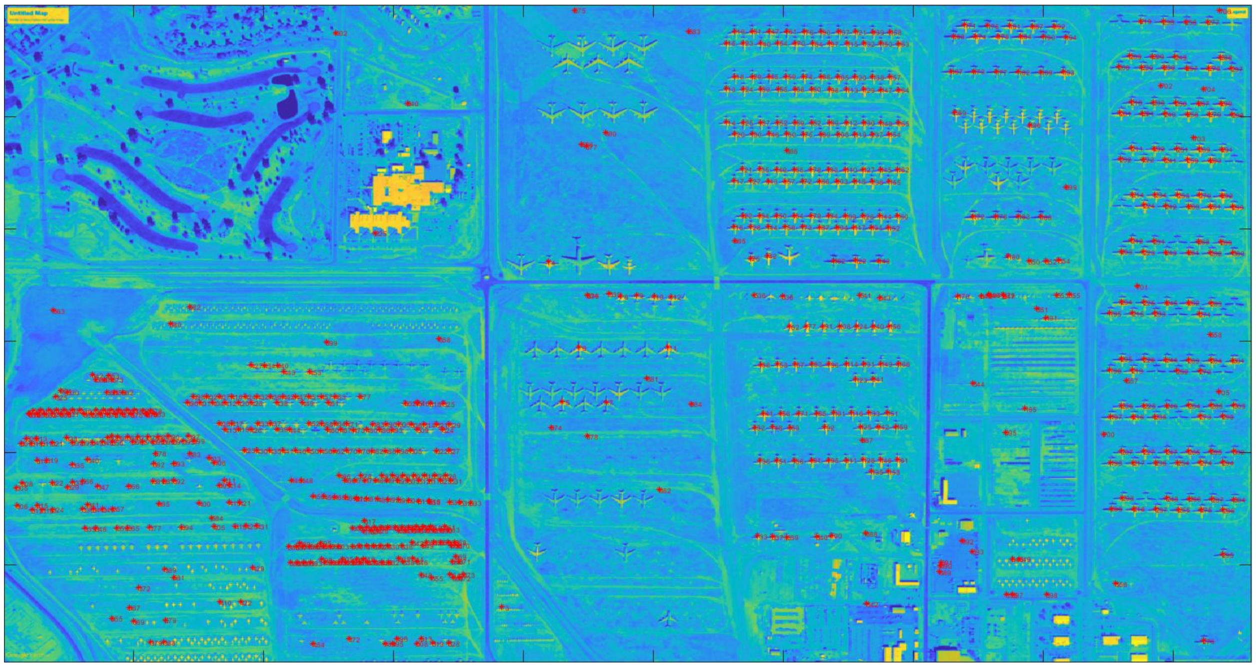
Throughout the duration of the project, there were many different results that came about due to trial and error. One of the pieces in the project we heavily experimented with was changing the Gaussian sigma values for the different types of templates. We noticed the higher the sigma value, the higher the number of true and false positives of the airplanes there were. If we were to lower the sigma values, the ratio of true positives to false positives would increase, but the total number of true positives would decrease. With a lower sigma value also comes a greater number of false negatives. Another part of the project we experimented with was changing the correlation threshold. Similar to changing the sigma values, if we were to lower the correlation threshold, there would be more false positives because there would be more “matches” which were usually just in incorrect areas such as on turf. Raising the threshold would decrease the number of false positives, but would also create more false negatives.

Intermediate Results

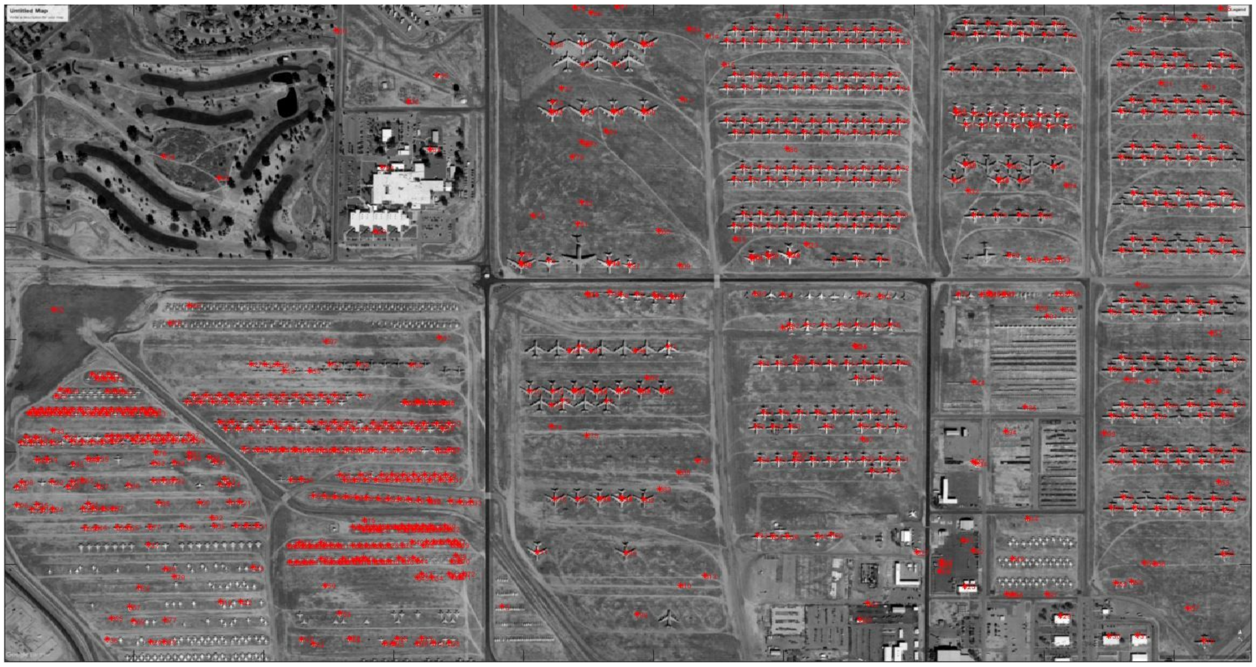
We experimented with a vast variety of templates, but only one worked well, the straight wing plane template. Through experimenting with the sigma values of the smoother and the threshold, we ended up using just this one template to identify most of the planes on the right hand side of the image, as follows:



From this point, we tried grabbing more templates from the smaller planes on the right hand side, but that felt like a roundabout way of approaching the problem. Instead, we decided to resize the straight wing template using the Matlab function imresize(), and the results were spectacular:



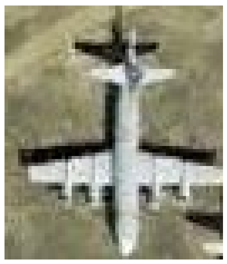
Our precision on the left hand side is incredible. We still miss some small planes, but we tried resizing the image even more with no luck. All that was left to do was add a new template that included the downward bent planes in the middle of the picture, and we obtained our final result (for this image):



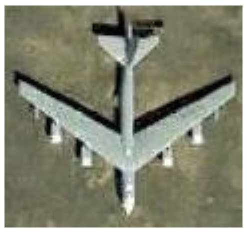
Explorations

- Started with a slice of one plane. Works effectively in identifying same type of plane, but can't generalize to other types of planes. Most of our experimentation was performed on the dma image because there is more variety to the types of planes seen.
- Next tried smoothing template and image. This gave us really good accuracy as picking out planes again, but it also picks up a lot of land.
- We tried to generalize our templates to accommodate for planes of different rotations, but it didn't work out due to the shadows of the planes. We tried filtering out the shadow on these rotations but it was not providing good results applied to the rotations so we abandoned the rotation method and tried to create a more generalized template.
- We then picked another template of 6 different planes and applied the same method as the first plane. With all of the different templates, we were able to get a good amount of the planes to match in the picture, but we wanted to see if we could further generalize our use of templates.

- We ended up adapting to reduce amount of templates to 2. For these 2 templates, we scaled them to different sizes to accomodate for the different sizes of planes. We applied a scaling of 0.5 and 0.25 to the “straight wing plane” template, and just a scaling of 0.5 to the “downward-bent wing plane” template.
- This method ultimately provided the best results on the dma image. The vcv image provides a different challenge due to orientation which we solved by taking similar templates at different orientation rather than scale. We did however apply the same process to that image once the templates were determined.
- Also we found a larger blurring factor had a positive effect in the vcv image possibly due to the base images being larger.



The “straight wing plane” template



“The downward-bent” plane template.

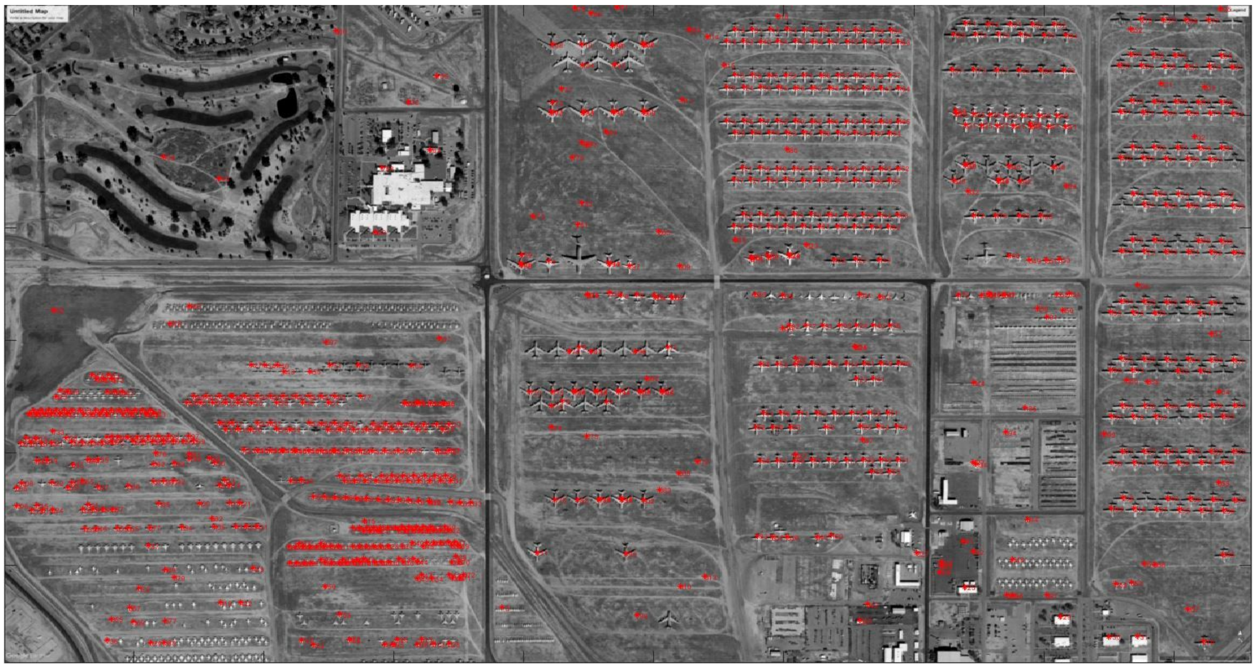
Evaluation

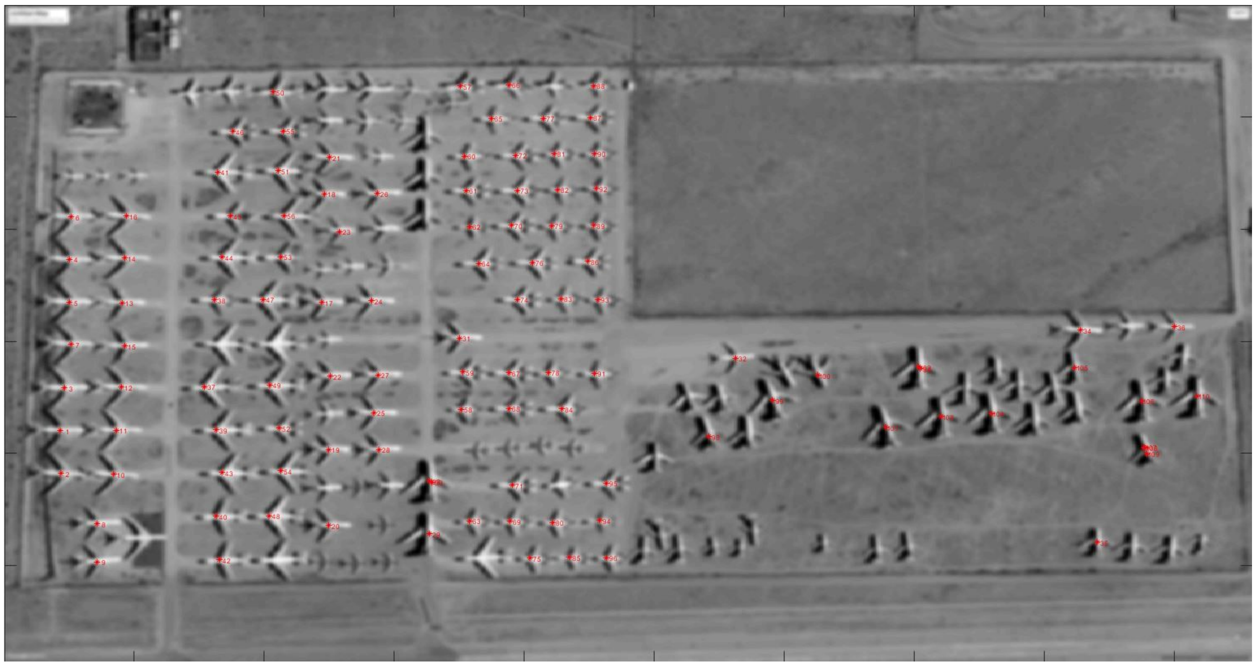
The most important evaluation metrics we care about are true positives and false negatives since we want to maximize the amount of planes we detect and minimize the amount we miss. False positives are important as we want to minimize the times we predicted a plane that isn’t actually there. The results on dma are listed below:

        - True Positives: 794/858 predictions are on a plane.
        - False Positives: 64/858 predictions are not on a plane.
        - False Negatives: 290/1084 planes were missed.
        - Precision: 92%
        - Recall: 73%
    - Results for vcv:
        - True Positives: 110/110 predictions are on a plane
        - False Positives: 0/110 predictions are not on a plane
        - False Negatives: 65/172 planes were missed

- Precision: 100%
- Recall: 62%

We are content with these results, as our precision is very high and our recall is good. Our final images are as follows:





Contributions

Brian Nguyen

- Worked on creating the pre-processing, convolution, non-max suppression, and thresholding pipeline. Created Summary and Approach sections of the report. Created pipeline visualization and documented code.

Kyle Bradley

- Worked on creating the pre-processing, convolution, non-max suppression, and thresholding pipeline. Created templates and tested results. Converted code to functional style.

Jordan Reed

- Worked on creating the pre-processing, convolution, non-max suppression, and thresholding pipeline. Resized templates. Created Intermediate Results section of report. Figured out how to add numbers to an image in Matlab.

Drice Bahajak

- Worked on creating the vcv templates and code. Helped to troubleshoot the code when it was not working the way it was expected to. Created the Observations section of the report.