CMPEN/EE 454, Project 1, Fall 2018

Due Fri Sep 14 by 11:59pm EST, submitted in Canvas

1 Motivation

The goal of this project is to implement template matching for the detection of objects in images. One of the most common tasks in computer vision is object detection; be it cars, faces, motion capture markers, or in this case airplanes.

Specifically, the goal is to detect airplanes from a satellite view. In this project you will be provided Google Earth images of airplane depots, with the goal of implementing a template matching based algorithm to detect and count the airplanes in each image.

While this appears to be a simple task at first, you will discover that practical applications have hurdles to overcome. Specifically, in each picture there are multiple different models of airplanes, and these planes are in different orientations in the image.

The specific project goals include:

- Gain experience in Matlab programming to implement computer vision algorithms
- Successfully import, manipulate, and convert images/matrices in Matlab
- Establish a method to select the templates used for matching
- Use correlation to score the similarity of the template to the image
- Use non-maximal suppression to locate and enumerate template matches
- Overlay graphics to show location/identification of each detection
- Successfully explain and visualize each step of the final algorithm

2 The Basic Operations

The following steps will be essential to the successful completion of the project:

1. Image input and conversion for manipulation: Each image needs to be read into Matlab and converted from a standard color image (MxNx3) to a grayscale image (MxNx1).
2. Selection of a template to use in the correlation: to detect airplanes using template matching the image must be correlated with an airplane template. To generate a template, there needs to be some automatic / manual template identification / selection / computation.
3. Localizing peaks from the template match: the output of template correlation will be a matrix with multiple peaks where the peaks are higher when the content in the image

has increased similarity to the template. Use non-maximum suppression to provide single pixel locations of each peak.

4. Use graphic overlays to enumerate each detection: the final locations need to be counted with a marker showing the detected airplane and a count added as a graphical overlay to the original image.

3 Evaluation

Quantitative

For this project, quantifying the accuracy of your algorithm's implementation is easy for you as a human. There are 3 values you should use:

• True positives: airplanes that the algorithm detects that do exist in the picture based on human perception.
• False positives: airplanes that the algorithm detects that do not exist in the picture based on human perception.
• False negatives: airplanes that the algorithm does not detect that do exist in the picture based on human perception.

It is essential to show specific examples of the successful detections and failure cases. It is important to identify, enumerate, and evaluate failure cases as part of a process of cyclical improvement to the algorithm's implementation.

Qualitative

As part of the report, each intermediate step of the process should be visualized to demonstrate the proper functionality of the algorithms implemented for the project. This will include visualizing the input image, the template(s) utilized, the correlation score matrix, the results of non-maximum suppression, and the final graphic overlay results. Figure 1 shows an example input and output image (this is a small sample from the provided test images.

The final report should provide results for each of the input files provided in the assignment. There are two (2) files (dma.jpg and vcv.jpg) provided for testing and evaluation which are both to be included in the report. Additionally, the example1.jpg input image shown in Figure 1 is provided for initial testing. It is recommended that during the initial development of the project code that only a portion of the full resolution test images be used as processing and testing the full images can be time consuming depending on the efficiency of the algorithms implementation. Each of the evaluation images is approximately 14MP with a resolution of 4800x2935.

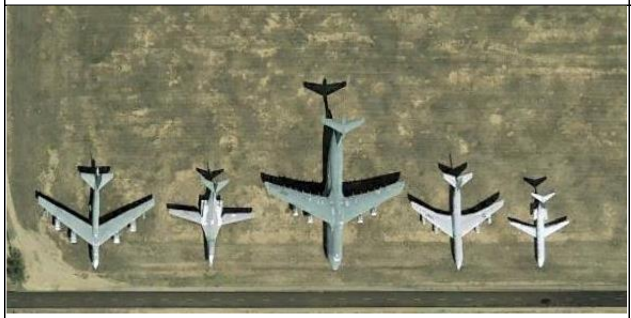
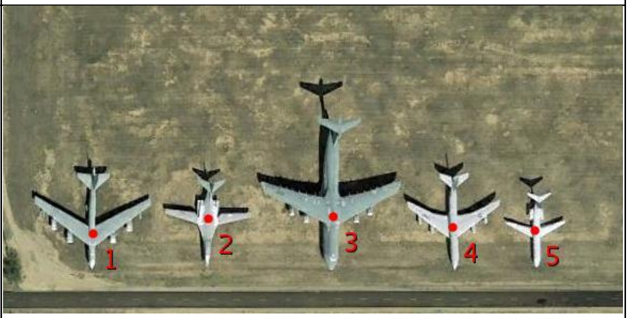
Input Image (example1.jpg)	**Output Image (example1_output.jpg)**
	

Figure 1 – Input (left) and output (right) example images. The output image presents an acceptable graphical overlay to indicate the locations and enumerations of each airplane. This example is a small sample of one of the images provided.

4 What Libraries Can I Use?

The intent is that you will implement the various computational modules described using general processing functions (https://www.mathworks.com/help/matlab/functionlist.html) in Matlab. The single notable exception is that you MAY use the image processing toolbox (https://www.mathworks.com/help/images/index.html). Specifically, you MAY NOT use anything from the computer vision toolbox, or any third-party libraries/packages. Don't plagiarize any code from anywhere or anyone else. It saddens us to even have to say that, but you'd be surprised what we discover each and every semester.

5 Project Reporting and Evaluation

Half of your grade will be based on submitting a fully operation program, and the other half will be based on a written report discussing your program, design decisions, and experimental observations. The following components will be submitted:

1) Submit a written report in which you discuss at least the following:
   a) Summarize *in your own words* what you think the project was about. What were the tasks you performed; what did you expect to achieve?
   b) Present an outline of the procedural approach along with a flowchart showing the flow of control and subroutine structure of your Matlab code. Explain any design decisions you had to make. For example, how did you choose or extract airplane templates? If you used raw cross-correlation, did you subtract off the mean of the template first? Did you use normalized cross-correlation instead? How did you implement non-maximum suppression? Basically, explain at each step why you chose to do whatever you did. Be sure to document any deviations you made from the above project descriptions (and

   why), or any additional functionality you added to increase robustness or generality of the approach. This project is a little open-ended, so detailed explanation of everything you did is necessary for us to fully appreciate your approach to solving this project.
   c) Experimental observations. What do you observe about the behavior of your program when you run it? Does it seem to work the way you think it should? Run the different portions of your code and show pictures of intermediate and final results that convince us that the program does what you think it does.
   d) Extensions and Explorations. If you are in an exploratory mood, find some images of your own on the web and input them to your code. Try an airport like Pittsburgh International (PIT), or Los Angeles International (LAX). Can your implementation find some or all of the airplanes?
   e) Document what each team member did to contribute to the project. It is OK if you divide up the labor into different tasks (it is expected), and it is OK if not everyone contributes precisely equal amounts of time/effort on each project. However, if two people did all the work because the third team member could not be contacted until the day before the project was due, this is where you get to tell us about it, so the grading can reflect the inequity.
2) Turn in a running version of your main program and all subroutines in a single zip or tar archive file (e.g. put all code, subroutines, etc in a single directory, then make a zip file of that directory for submission via Canvas). We can then unzip/untar everything and go from there.
   a) Include one or more demo routines that can be invoked with no arguments and that load any input needed from the local directory and display intermediate **and** final results that show the different portions of your program are working as intended.
   b) We might be running the code ourselves on other input, so include a routine that takes as input a single image and that outputs the matching location results.
   c) Include enough comments in your functions so that we have a good idea what each section of code does. The more thought and effort you put in to demonstrating / illustrating in your written report that your code works correctly, the less likely we feel the need to poke around in your code, but in case we do, make your code and comments readable.

FAQ: How long should your report be? We don't have a strict number of pages in mind, but as a general rule-of-thumb, if the length of your report (excluding figures) is less than the length of this project description document, you probably haven't included enough detail about what you did/observed.

FAQ: Will the submitted code be evaluated for functionality and coding quality? Yes. Your code and all subsequent functions you write and submit can be reviewed and tested. Please make sure the code is your own unique work and no one else's, and that each function and what it does is easy for graders to identify.

6 Grading Criteria

Project 1 grade = (100 – Deductions + Additions) × (1– Late Penalty)

Deductions - Implementation

- Implementation of template matching [10pts]: correlation scores calculated for each image for each template selected.
- Implementation of non-maximal suppression [10pts]: processing of the correlation scores to identify each location of a template match
- Implementation of graphical overlay [10pts]: each detected airplane is to be identified with a location marker and count value as a graphic over top of the original image
- Program structure and readability [10pts]: The code should have enough comments to explain the functions and procedures while being properly divided into small modules (function calls).
- Code should be runnable [10pts]: Each function created should operate properly when called as part of your implementation and should not cause failures in any demonstration code. Functions may be spot checked to verify proper coding style and commenting.

Deductions - Report

- Implementation Description [10pts]: outline of your code along with proper diagrams. Unless the operation is predefined in Matlab, explanation of what operation is being applied is required. This is a step by step of WHAT was done to achieve the desired outcome.
- Experiment Observations and Explanations [15pts]: This is a step by step explanation of WHY an operation was used in the implementation.
- Experiment Visualizations [15pts]: This is a step by step visual representation of intermediate and final results.
- Evaluation and Discuss [10pts]: This is an explanation of the results, the overall successes and failures of the implementation as compared to the ideal result (what a human can do).

Additions - Implementation

- Good demo routine [5pts]: clearly demonstrating the functionality and correctness of your code. It can be a big demo for the whole project or small demos for function parts. A good demo is a script that can be opened in Matlab and run to demonstrate the code producing the reported results.

Additions - Report

- Separation of airplane models [5pts]: the detection, location, and enumeration of airplanes of different models. Each type of airplane model should be counted as a separate group.
- Accommodating changes of scale [5pts]: modifying the template / image scale to find airplanes of different sizes using a single template.
- Accommodating changes of rotation [5 pts]: modifying the template/image rotation to find airplanes having different orientations in the image using a single template.

Late Penalty

Deduction	**Late Time**	**Time Stamp Submitted**
0%	Not Late	By 11:59**pm** 09/14/18
10%	Up to 12 hours	By 11:59**am** 09/15/18
20%	Up to 24 hours	By 11:59**pm** 09/15/18
40%	Up to 36 hours	By 11:59**am** 09/16/18
80%	Up to 48 hours	By 11:59**pm** 09/16/18
100%	More than 48 hours	After 11:59**pm** 09/16/18