ENGR 13300
Python Group Project Fall 2022

Project Description:

Object and feature detection is one of the key issues in computer vision (CV) and has wide applications. For example, self-driving automobiles need traffic sign detection for scene understanding, better cruise control, and route design. Materials engineers use CV feature detection methods to perform phase recognition and identify grain boundaries to study the performance of different metal alloys. Those methods can also be used for Magnetic Resonance Imaging (MRI) images to find specific features which are hard to detect by eye. In recent years, with the development of the neural network, the efficiency and accuracy of object detection have been significantly improved and have surpassed human vision when doing certain tasks like classification of numbers.

In this project, you will implement a classic object detection method called template matching. You will need to program three types of template matching methods based on different metrics, namely the sum of squared differences (SSD) method, the correlation coefficient method, and the normalized cross-correlation method. You will first use these to find the location of Waldo with templates given to you. Then you will find/create your own template and implement those methods for a real-life application of your choice.

Requirements:

1. You will be only allowed to use `numpy`, `time`, and `matplotlib`, and the Python standard library.
2. You are **NOT** allowed to use `OpenCV`, `scipy`, or any other advanced libraries.
3. **Utilize function(s) in this project to allow for modularity and reusability.** This will also allow your project team to assign each group member with a specific task, and the member will individually work on their task and build their user-defined function that accepts inputs they need and return the desired output. If your team only has three members, you do not need to program the SSD method.

Deadlines:

In class **demo: October 18th**

Final Reports and Programs **uploaded to Gradescope, Tuesday, October 18th at the start of class**.

Part 1: Image importation

In the first part of this project, you will develop a Python program to import the image and the template. You can find ways to import images using `matplotlib` in the resources provided and in previous Python homework assignments.

Your program should:

- **Prompt the user for the name of a color image and a color template file.**
- **Check the data type and the range of the image data. The data type of the image should be uint8 integers, which range from 0 to 255. If the data type and range are not correct, you will need to convert the image data to the correct range and make sure the image data is in the correct data type.**
- **Convert the source image and template to grayscale.**
- **Pass the image data to other UDFs to perform template matching and generating the required output.**

Try to answer the following questions in your report:

- **What are the data type and the range of the data types if you try to import a PNG file using `matplotlib`? How about with a JPG image?**
- **If the data type of the image data was not of the unit8 type, how did you convert that?**

Part 2: Finding Waldo

Template matching is a commonly used computer vision technique that identifies the object in an image that matches a predefined template. Template matching is flexible and relatively straightforward to use, which makes it one of the most popular methods of object detection.

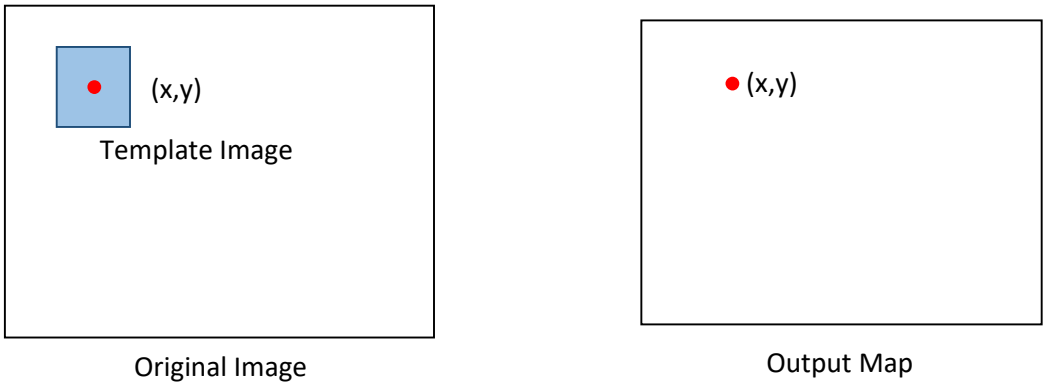


Figure 1. illustration of template matching

During the matching process, your code will move the template, one pixel at a time, to all possible positions in the large source image and compute a numerical value for each location which indicates how well the template matches the image in that location. After the calculation, an output map can be

generated, and the location of the object (e.g. Waldo) can be determined from that map. In this part of the project, you will work with grayscale images for template matching.

Sum of Squared Differences (SSD)

The Sum of Squared Differences (SSD) directly compares the pixel values of the template and the original image and calculates the difference. It can be described by the following equation:

$$R(i,j) = \sum_{k=0}^{w-1}\left(\sum_{l=0}^{h-1}\big(I(i+k,j+l) - T(k,l)\big)^2\right) \quad (1)$$

Where

I is a matrix representing the source image

R is the output map matrix

T is a matrix representing the template image

i, j are the indexing of the output map

k, l are the indexing of the template

w, h are the width and height of the template image, the unit is in number of pixels.

The values in *R* represent how similar the template is to the source image at each location. For this algorithm, the smaller the value in *R* is the more similar the template is to the source image at that location.

Correlation Coefficient

The correlation coefficient measures the degree to which the pixel values of two images are associated. It can be described by the following equation:

$$R(i,j) = \sum_{k=0}^{w-1}\left(\sum_{l=0}^{h-1} I'(i+k,j+l)T'(k,l)\right) \quad (2)$$

$$I' = I - \frac{1}{wh}\sum_{k=0}^{w-1}\left(\sum_{l=0}^{h-1} I(i+k,j+l)\right) \quad (3)$$

$$T' = T - \frac{1}{wh}\sum_{k=0}^{w-1}\left(\sum_{l=0}^{h-1} T(k,l)\right) \quad (4)$$

Where

I is a matrix representing the source image
R is the output map matrix
T is a matrix representing the template image
i, j are the indexing of the output map
k, l are the indexing of the template
w , h are the width and height of the template image, the unit is in number of pixels
W,H are the width and height of the source image, the unit is in number of pixels
I' is the result of finding the mean of I in a window of size (*w,h*) and subtracting that mean from each element of I in that window .
T' is the result of subtracting the mean value of T from each element of T.

For this algorithm, the larger the value in *R* is the more similar the template is to the source image at that location.

Normalized Cross-Correlation

Cross-correlation is a measure of the similarity of two series as a function of the displacement of one relative to the other. In CV, this concept was applied to template matching. To minimize the effect of illumination discrepancy (e.g. intensity change due to different exposure time), the pixel value range of the template and the source image will be normalized. The equation for the output map is:

$$\mathrm{R(i,j)} = \frac{\sum_{\mathrm{k=0}}^{\mathrm{w-1}}\left(\sum_{\mathrm{l=0}}^{\mathrm{h-1}} I(i+k, j+l)T(k,l)\right)}{\sqrt{\left(\sum_{\mathrm{k=0}}^{\mathrm{w-1}}\left(\sum_{\mathrm{l=0}}^{\mathrm{h-1}} I(i+k, j+l)^2\right)\right) \cdot \left(\sum_{\mathrm{k=0}}^{\mathrm{w-1}}\left(\sum_{\mathrm{l=0}}^{\mathrm{h-1}} T(k,l)^2\right)\right)}} \quad (5)$$

For this algorithm, the larger the value in *R* is the more similar the template is to the source image at that location. *(Note: $I(i+k, j+l)^2$ and $T(k,l)^2$ refer to squaring each element of the matrix*

There are source images and templates of Waldo on Brightspace. You will implement these algorithms to find Waldo, and your algorithms should work for any of the image-template pairs. You will need to show the output map, the source image with a bounding box that highlights Waldo, and a cropped image showing the details, as shown in Figure 3. You will also need to print out the pixel indices of Waldo's location.

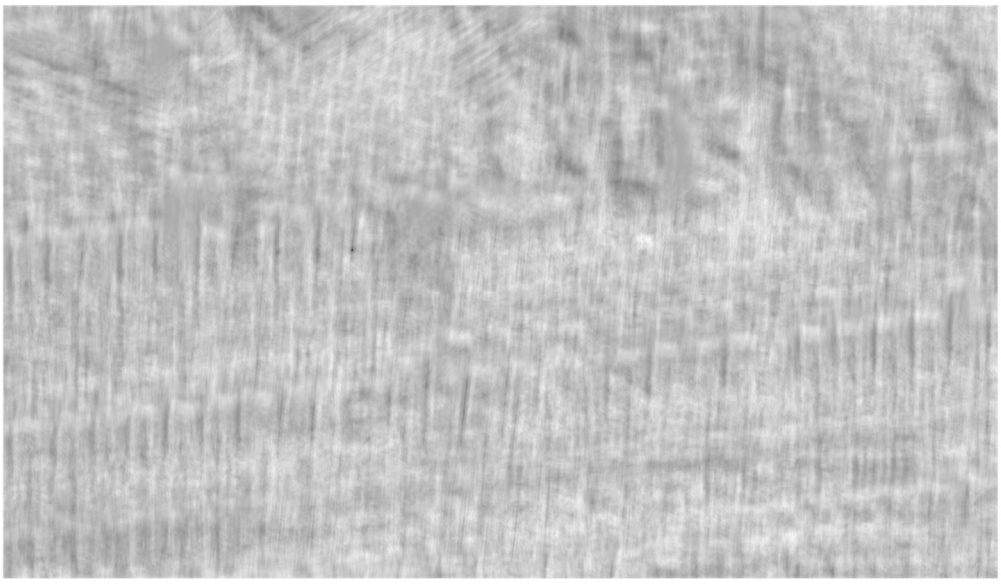
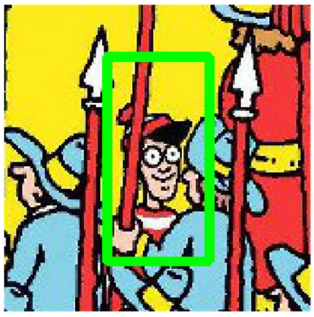


Figure 1. An output map

Figure 2. Original image with bounding box



Cropped image

Figure 2. Example output of Part 2

Your program should:

- **Prompt the user for the type of template matching method**.
- **Show and output the output map, the original image with a bounding box highlighting the template, and a cropped image of Waldo with a bounding box highlighting the template.**
- **Print out the location of Waldo in the source image.**

Try to answer the following questions in your report:

- **What's the difference between these three template matching methods? How well were those methods performed?**
- **Which method took less time to calculate? You can use the `time.time()` to estimate the time took for each method.**

Part 3: Real-life application

In this part of the project, you will pick your own template and image. You will apply the template matching for a real-life scenario. It can either be traffic sign detection, street number identification, human detection, or any other area where template matching can be used. Be creative! You will need to compare the performance of the three methods.

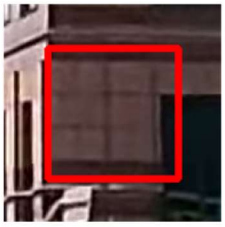
- Note that the template matching methods above may not give you the correct location. You may need to think about how to combine these three methods together to achieve the optimal result. *(Hint: How could you combine the output map of each method into a meaningful result?)*

Example Template

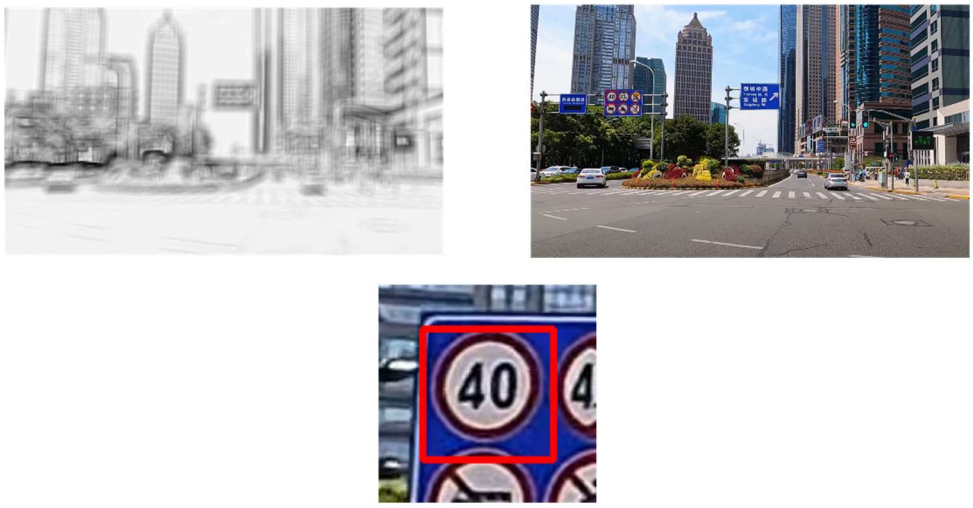
Example Image

SSD

Correlation Coefficient



Normalized Cross-Correlation



Combined Method

Figure 3. Implementation of traffic sign detection using template matching techniques

Your program should:

- **Prompt the user for the type of template matching method**.
- **Show and output the output map, the original image with a bounding box highlighting the template, and a cropped image with a bounding box highlighting the template.**

Try to answer the following questions in your report:

- **How did you get the template and the source image? What's interesting about the image you picked?**
- **How well did those methods perform? Why are those methods performed like this?**
- **What method did you use, or what changes did you make to get the optimal solution?**

Deliverables

Each team will submit a project report outlining their research into object/feature detection, the application of object/feature detection, and the program the team developed. The project report will follow the common structure of a technical report. The Python files, along with a report, will be uploaded to Gradescope as the final submission. Make sure that the files will run as they are uploaded (e.g. have them all in the same folder). You may zip all your files together. The report will consist of:

1. **Project Motivation**

   You will need to describe how object/feature detection and template matching methods are used in an engineering discipline citing at least three credible sources using APA formatting. This section will outline the major problems trying to be solved in these areas and the state-of-the-art methods that have been developed or are widely used.

2. **Project Overview and Methods**

   In this section, you will describe the background of each of the template matching methods and how they work.

3. **Discussion of Algorithm Design**

   This section will include flow charts of the overall program and each of the user-defined functions. This section should describe the design rationale for modularizing the code to meet the needs of future uses of object detection. You will need to summarize the algorithm you designed and used for Python programming. Answer the questions from each part above in this part of the report.

4. **References:**

   Include appropriate citations, using the APA format, for any resources your team used in writing your program, researching the applications of object detection, or researching template matching in different disciplines.

5. **Appendices: Include as appendices to your main report**
   a. **User manual** with sample inputs and outputs (including pictures).
   b. **Project management Plan**: Summary of the contributions of each member of the team to the project. Describe your team's methods for collaboration that facilitated active participation and learning by all team members. Include a discussion of opportunities for improvement for future team projects.
   c. **Discussion of Design Process**: Description of how your team followed the design process, which is defined as "a process of devising a system, component, or process, to meet desired needs within constraints. It is an iterative, creative, decision-making process that involves developing requirements, performing analysis and synthesis, generating multiple solutions, evaluating solutions against requirements, considering risks, and making trade-offs."
   d. **Code:** The team should upload the Python files, but they should also include the code as an appendix in the report. Code should be well commented to increase readability for future users of the algorithm.

Project Grading Summary (a detailed grading rubric will be provided):

Points	Code
70	Code runs with no errors, and image processing and template matching portions are implemented using only `matplotlib` and `NumPy` modules
6	Errors handled and structured appropriately
10	Comments in code, appropriate variable, the code is readable
	Report
5	Project motivation and application to major or career interest(s)
10	Project overview and methods discussion
10	Algorithm Design rationale and Flow charts
5	User manual for code
3	Project Management Plan (Summary of contributions)
3	Discussion of Design Process
3	Appendices include the code
125	**Total**

In class demo

You will need to show the followings:

- Finding Waldo with a given template using different algorithms. You will need to show the original image with bounding box and the cropped one which highlights the location of Waldo.
- Show application of your program of a real-life application you pick.
- How your program is dealing with invalid inputs.