University at Buffalo
Department of Computer Science and Engineering

CSE 473/573 – Computer Vision and Image Processing

Spring 2020
TuTh 9:30AM - 10:50AM
HOCH 114

Project #1

Due: 2/27/20, 11:59pm

1. Edge Detection [35 points]

The goal of this task is to experiment with two commonly used edge detection operator, i.e., Prewitt operator and Sobel operator, and familiarize you with tricks, e.g., padding, commonly used by computer vision practitioners. Specifically, the task is to detect edges in a given image, which is named "proj1-task1.jpg" and is stored in "./data/".

You are required to implement all the functions that are labelled with "# TODO" in "task1.py" and "utils.py". In "task1.py" and "utils.py", we not only provide hints to you, but also provide utility functions that could be used as building blocks for you to complete this task. Therefore, you only need to write about 40 lines of code. Your code should be able to generate images that are identical to those stored in "./results/" (Note that images in "./results/" are provide to you for reference only. At test time, a different image will be used as input and the correct output images will be different from those already stored in "./results/").

Comment the lines "raise NotImplementedError" instead of deleting them, when you implement the functions labelled with "# TODO".

2. Character Detection [65 points]

The goal of this task is to experiment with template matching algorithms. Specifically, the task is to find a specific character (or set of characters) in a given image, which is named "proj1-task2.pgm" and is store the results in "./data/". You are required to implement a function named "detect" in "task2.py", which detects a character in an image. The function "detect" takes a given image and a given template (that you will implement) that contains a character as input and returns the coordinates (i.e., coordinates of the top-left pixel) of the character contained in the template.

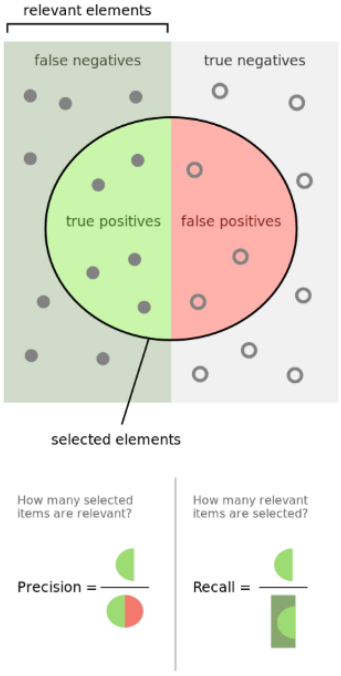
This task is composed of the following three sub tasks:

- Detect character "a" (lower case "a"). [3 point]
- Detect character "b" (lower case "b"). [3 point]
- Detect character "c" (lower case "c"). [3 point]

You need to customize your own templates. The templates containing characters "a", "b" and "c" should be named as "a.pgm", "b.pgm" and "c.pgm" respectively. They should be stored in "./data/". Note that only ONE template will be used to detect all instance of a particular (lowercase) character, but it can be scaled or rotated, for example, in your program.

Evaluation

The F1 measure is the harmonic mean of Precision and Recall, with Precision being the # of true positives / # one says are positive, and the recall is the # of true positive / # of positives that exist.



$$\text{Precision} = \frac{\text{True Positive}}{\text{Predicted Positive}}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{Actual Positive}}$$

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \textit{Recall}}{\text{Precision} + \text{Recall}}$$

The FI will be used as the metric. You are not expected to get all the characters, but so you can assume that if you get an F1 measure > 0.6 you will get full credit

Hints:

- You may consider designing a grayscale template that emphasizes some pixels more than others.
- You may consider combining several "exact" template in an appropriate way to be able to recognize different variations of the 3 lower case characters in question.
- Once you read in your base template your program can further manipulate it for, for example, size.
- After you have a base system, you may also experiment with templates which only contain edges for template matching (detecting edges in both the image and the template). This might give better results than using the original image and template for template matching, as edges only preserve the scales and shapes of characters, which are important for template matching. Distracting information, e.g., colors and fonts, of the characters are partially eliminated in the image and template that only contain edges. (Functions that we provide to you in "utils.py" and functions that you implement in "task1.py" could be used to detect edges.)
- PGM images are a simple raw representation of any image, and additional information can be found here: http://netpbm.sourceforge.net/doc/pgm.html

Project Guidelines

- Do **not** modify the code provided.
- Do **not** use any API provided by opencv (cv2) and numpy (np) in your code (except "np.sqrt()", "np.zeros()", "np.ones()", "np.multiply()", "np.divide()", "cv2.imread()", "cv2.imshow()", "cv2.imwrite()", and "cv2.resize()").
- Do **not** import any additional libraries (function, module, etc.) except native Python packages, e.g., pdb, os, sys.
- You will upload your project to be UBLearns and to Autograder (https://autograder.cse.buffalo.edu/), so please try both early in the process to make sure you understand

how they work.

Ask for the TA's approval if you would like to use a programming language other than Python, C, C++, C# and Java.