University at Buffalo
Department of Computer Science and and Engineering
CSE 473/573 - Computer Vision and Image Processing

Project #1
Due Date: 3/9/22, 11:59PM

1 Optical Character Recognition (100 points)

1.1 Goals

The goal of this task is to implement an optical character recognition system. You will experiment with connected component and matching algorithms and your goal is both detection and recognition of various characters.

The first input will be a directory with an arbitrary number of target characters as individual image files. Each will represent a character to recognize in the from of a ”template”. Code will be provided to read these images into an array of matrices. You will need to enroll them in your system by extracting appropriate features.

The second input will be a gray scale test image containing characters to recognize. The input image will have characters that are darker than the background. The background will be defined by the color that is touching the boundary of the image. All characters will be separated from each other by at least one background pixel but may have different gray levels.

1.2 Implementation

The OCR system will contain three parts, Enrollment, Detection and Recognition.

1. Enrollment
   - Code will be provided to read in a set of test images from a provided directory
   - You process an enrollment set of target characters from the provided directory and generate features for each suitable for classification/recognition in Part 3 (Recognition).
   - You may store these features in any way you want to in an intermediate file, that is read in by the recognizer. The file you store should NOT be the same as an image file. The reason for the intermediate file is so you do not have to run enrollment every time you want to run detection and recognition on a new image.
   - Rubric: 20 points - 10 points (code) and 10 points (features).
2. Detection

- Code will be provided to read in a test image.
- Once you have read in the test image, you will need to use connected component labeling (that you implement) to detect various candidate characters in the image.
- You should identify ALL possible candidates in the test image even if they do not appear in the list of enrolled characters.
- The characters can be between 1/2 and 2x the size of the enrolled images.
- Once you have detected the character positions, you are free to generate any features or resize those areas of the image in any way you want in preparation for recognition.
- The detection results should be stored for output with the recognition results in part 3, and should be in the original image coordinates.
- Rubric: 30 points - 10 points (code) and 20 points (evaluation)

3. Recognition

   - Taking each of the candidate characters detected from previous part, and your features for each of the enrolled characters, you are required to implement a recognition or matching function that determines the identity of the character if it is in the enrollment set or UNKNOWN if it is not.
   - Rubric: 50 points - 10 points (code), 40 points (evaluation)

1.3 Output

For the output, you are expected to generate an output file 'results.json'. It will be a list with each entry as {"bbox" : [x (integer), y (integer), w (integer), h (integer)],"name": (string)} :

- x, y are the top-left coordinates of the detected character. Consider origin (0,0) to be the top-left corner of the test_image and x increases to the right and y increases to the bottom.
- the h is the height, and w is the width of detection (from part 2),
- matching enrollment character identify from part 3 (if any). Use "UNKNOWN" for characters that are not in the enrollment set.

The order of detected characters should follow English text reading pattern, i.e.,list should start from top left, then move from left to right. After finishing the first line, go to the next line and continue.

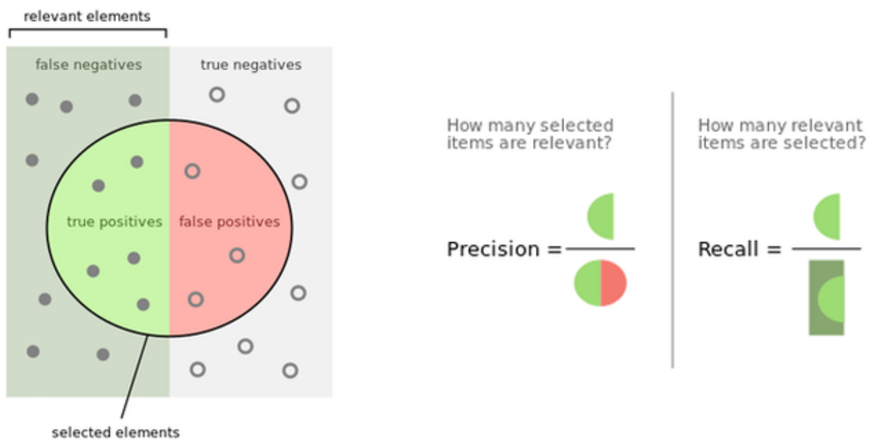
Note that

- the final code should read target enrollment images from the directory "characters", read the file "test_img" from the main directory, and write the output file"results.json" to the main directory.
- the size of the test characters MAY differ from the size of the enrollment images.

You also need to submit a report "report.pdf" (approximately 1 page) explaining how you computed features, performed detection, and performed recognition.

1.4 Evaluation

- For part 1 (20 points), computing appropriate features for the task would fetch you points.
- For part 2 (30 points), we will proportionately assign a score based on the number of characters detected. A character is considered to be detected if more than 50% of the character is covered by the bounding box.
- For part 3 (50 points), 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. The F1 will be used as the metric. you can assume that if you get an F1 measure > 0.6 you will get full credit. Note that "UNKNOWN" detections will not be counted towards f1 score.



- Irrespective of the F1 score, using template matching without generating features would result in a maximum of 90 points for the project.
- We have provided "groundtruth.json" for the data in "data" folder. You may use it to compute F1 score with 'evaluate.py'. However, for final evaluation, we will use similar but different test_image and character group (i.e., English alphabets and numbers).

2 Project Guidelines and Submission

- Do not modify the code provided.
- All work should be your own. You are not permitted to copy code from the internet.
- You are free to use any opencv (cv2 version 3.4.5 or 4.5.4 ONLY) and numpy (np) function for generating features.
- For other parts of your code, you may only use BASIC except "cv2" and "np" API calls such as "np.sqrt()", "np.zeros()", "np.ones()", "np.multiply()", "np.divide()", "cv2.imread()", "cv2.imshow()", "cv2.imwrite()", "cv2.resize()".

- **You may NOT use any other API provided by opencv (cv2) and numpy (np) for generating connected components, matching, computing distances, setting thresholds, etc.**
- Do not import any additional libraries (function, module, etc.) except native Python packages, e.g., pdb, os, sys.
- Please include a text file called 3.4.5.txt or 4.5.4.txt with your submission to indicate the version of opencv you have used.
- Compress the python files, i.e., "task1.py", "data" folder, features folder (from part 1), "results.json", "report.pdf", and the version file from the previous item, into a zip file, name it as "UBID.zip" (replace "UBID" with your eight-digit UBID, e.g., 50305566) and upload it to UBLearns before the due date.
- Late submission guidelines apply for this project.
- Not following the project guidelines may result in penalty.