ECE5554 SU22 - Prof. Jones – HW 2 – CORRECTED!

Due Wednesday, July 20, 2022 – 11:59 PM via Canvas

You are to write and test a Python/OpenCV program that will perform the first steps of QR code recognition – that is, to find the QR code in an image and produce a rectified image of the code alone. Notice that a QR code is delineated by three corner markers that look like this:

To take any general image containing a QR code, you must find three occurrences of this marker in the image, define an affine mapping to transform the image into a new image that is the proper size and rotation, and transform the image. So, for an input image that looks like the image on the left, I want an output image that looks like the one on the right (which is 300 by 300 pixels in size):

I have given you a set of five QR code images. Your program should do the following:

1. Create a suitable template for use in template matching. This should be an 8-bit grayscale image of just the marker, 128 by 128 pixels in size.
2. For QR_A through QR_B (save QR_E for a bit later):
    a. Use cv2.matchTemplate() to find the three locations of the markers in each input image. Note that it takes a bit of work to examine the output of matchTemplate to find the three best marker locations!
    b. Print to the console the file name and the found locations of the three markers.
    c. Write the results array returned by matchTemplate() to disk as a cv2 image.

d. Create a copy of the input image, with squares showing the found locations of the three markers in red. Write this image to disk; it should look something like this:

e. **NOTE CORRECTION ON THE NEXT LINE!!!**
Define an affine transform to map the three marker locations in the image to the points (50, 50), (250, 50) and **(50, 250)** ~~(250, 250)~~. This is done with cv2.getAffineTransform().

f. Use cv2.warpAffine() to geometrically transform the image into a new 300 by 300 result image, as shown above.

g. Write the affine transformed image to a disk file.

3. Once you have processed images A through B, test your code on QR_E.png. Did it work? If not, modify your code so that it works for all five images without changing parameters. You also may NOT resize any of the images to accomplish this!

Be sure and obey proper practice for loading, converting and using image files. Use numpy functions for the image arithmetic. You will find this documentation helpful:

- https://docs.opencv.org/3.4/d4/d61/tutorial_warp_affine.html
- https://docs.opencv.org/4.x/da/d54/group__imgproc__transform.html

Your submission will consist of a Word or pdf file and a .py file (no .ipnyb files) containing your code, as well as fifteen image files (the matchTemplate result, the image with red target squares and the final affine-transformed image for each of the five inputs). Do NOT put all of your files into a zip file; attach them separately. Your Word file should contain:

- your complete Python code (pasted in as plain text, no screenshots or dark mode);
- the console output of your program as described above, pasted in as plain text;
- the five final affine-transformed images from your program, sized large enough to see the detail.