Problem 1. (25 pts) A two dimensional correlation can be implemented more efficiently if the filter $w(x, y)$ is separable, meaning that it can be written as a product of two functions, one that is a function only of x and the other only of y. In other words, $w(x, y) = w_x(x)w_y(y)$ The 3×3 averaging filter is an example of a separable filter:

$$w(x,y) = \begin{cases} 1 & -1 \leq x \leq 1 \text{ and } -1 \leq y \leq 1 \\ 0 & \text{otherwise} \end{cases}, \quad \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

This is separable because it can be written as $w(x, y) = w_x(x)w_y(y)$ where

$$w_x(x) = \begin{cases} 1 & -1 \leq x \leq 1 \\ 0 & \text{otherwise} \end{cases}, \quad \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

$$w_y(x) = \begin{cases} 1 & -1 \leq y \leq 1 \\ 0 & \text{otherwise} \end{cases}, \quad \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

In the case of a separable filter, you can do a 1D correlation with $w_y(y)$ along the individual columns of the input image, followed by computing a 1D correlation with $w_x(x)$ along the rows of the previous result. Demonstrate the validity of this with an example.
Consider the small image matrix below (consider it to be padded outside with zeroes):

$$\begin{bmatrix} 1 & 3 & 0 & 1 & 2 \\ 0 & 1 & 2 & 1 & 0 \\ 1 & 0 & 0 & 3 & 2 \\ 0 & 1 & 2 & 1 & 0 \\ 1 & 2 & 0 & 3 & 2 \end{bmatrix}$$

By hand, compute the 2D correlation of this matrix with the full 3×3 averaging filter. Then compute the two 1D correlations as described above, and show that they are the same.

Let's start by padding the image matrix with zeroes:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 3 & 0 & 1 & 2 & 0 \\ 0 & 0 & 1 & 2 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 3 & 2 & 0 \\ 0 & 0 & 1 & 2 & 1 & 0 & 0 \\ 0 & 1 & 2 & 0 & 3 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Now we can compute the 2D correlation by passing the 3×3 ones matrix through the padded image matrix above. For demonstration purposes, if we center the 3×3 ones correlation

matrix on the 2nd row and 2nd column of the above, i.e. first entry of value 1, we have the following result where we sum up the products (starting with the first row):

$$[1(0)+1(0)+1(0)]+[1(0)+1(1)+1(3)]+[1(0)+1(0)+1(1)]=1+3+1=5.$$

This ends up being the result for our first entry. Doing this operation across all entries, we end up with the following 2D correlation result:

$$\boxed{\begin{bmatrix} 5 & 7 & 8 & 6 & 4 \\ 6 & 8 & 11 & 11 & 9 \\ 3 & 7 & 11 & 11 & 7 \\ 5 & 7 & 12 & 13 & 11 \\ 4 & 6 & 9 & 8 & 6 \end{bmatrix}}.$$

Now, we can consider applying the 1D correlation with $w_y(y)$ on the zero-padded image matrix. By centering the 3×1 ones column vector for $w_y(y)$ on the second row and first column, we get a zero correlation, given $1(0)+1(0)+1(0)=0$ due to the zero padded column. By centering the 3×1 ones column vector for $w_y(y)$ on the second row and second column of the padded matrix, we have $1(0)+1(1)+1(0)=1$ as the 1D y-correlation result. Sliding the ones column vector across the matrix, we end up with

$$\begin{bmatrix} 0 & 1 & 4 & 2 & 2 & 2 & 0 \\ 0 & 2 & 4 & 2 & 5 & 4 & 0 \\ 0 & 1 & 2 & 4 & 5 & 2 & 0 \\ 0 & 2 & 3 & 2 & 7 & 4 & 0 \\ 0 & 1 & 3 & 2 & 4 & 2 & 0 \end{bmatrix}.$$

We can now apply the 1D x correlation by applying the 1×3 ones row vector to the above result. The first correlation is $1(0)+1(1)+1(4)=5$. Computing the 1D x correlation to the above 5×7 matrix results in

$$\boxed{\begin{bmatrix} 5 & 7 & 8 & 6 & 4 \\ 6 & 8 & 11 & 11 & 9 \\ 3 & 7 & 11 & 11 & 7 \\ 5 & 7 & 12 & 13 & 11 \\ 4 & 6 & 9 & 8 & 6 \end{bmatrix}},$$

which matches the result of the 2D correlation boxed above.

Problem 2. (25 pts) Using the method of normalized cross-correlation, find all instances of the letter "a" in the image "textsample.tif". To do this, first extract a template subimage $w(x,y)$ of an example of the letter "a". Then match this template to the image (you can use OpenCV's "matchTemplate" function). Find local peaks in the correlation scores image. Then threshold these peaks (you will have to experiment with the threshold) so that you

Figure 1: The template.

get 1's where there are "a"s and nowhere else. You can use OpenCV's "connectedComponentsWithStats" function to extract the centroid of the peaks. Take the locations found and draw a box (or some type of marker) overlay on the original image showing the locations of the "a"s. Your program should also count the number of detected "a"s.
Turn in:

1. Describe your method of solution.
2. Give the program listing, with comments.
3. Show the template image and the scores image.
4. The output number of "a"s, and the overlay image showing their locations.

I used the "get-xy" function (referenced from the lecture notes) to identify the position of the click in an image. After experimenting with several clicks, I decided to hard-code the position that results in a 16×16 template image with only an "a" and not another letter. Next, I ran the "cv2.matchTemplate" function and determined the matched locations based on different threshold values. A threshold value of 0.7 resulted in a correct identification of the letter "a"s. I referred to the openCV documentation and applied NP.WHERE(C >= THRESHOLD) to find the x and y positions for which the correlations matrix is greater than or equal to the threshold value of 0.7. Using those positions, I drew rectangles on the "a"s in the original image and also placed a white pixel in an initially black scores image. I applied a threshold on the resulting scores image using "cv2.threshold" and then ran "cv2.connectedComponents" to determine the number of "a"s that have been detected. Refer to "HW2-Q2.py" for the program listing.

The template image is shown in Fig. 1 and the scores image is shown in Fig. 2. The total number of "a"s detected is 54 and the overlay image showing their locations is shown in Fig. 3. I keep counting only 53 "a"s, so I am not sure why the code adds an extra count. Slide 15 in lecture notes "4-1-BinaryImages.pdf" does show a case where the white labels are 18 and the black labels are 20. Given each white label has a black circle and there is a black background, we expect 19 black labels and not 20. I wonder if there are times where the connected components command counts an extra item. Either way, my code detects 54 "a" locations.

Figure 2: The scores image.

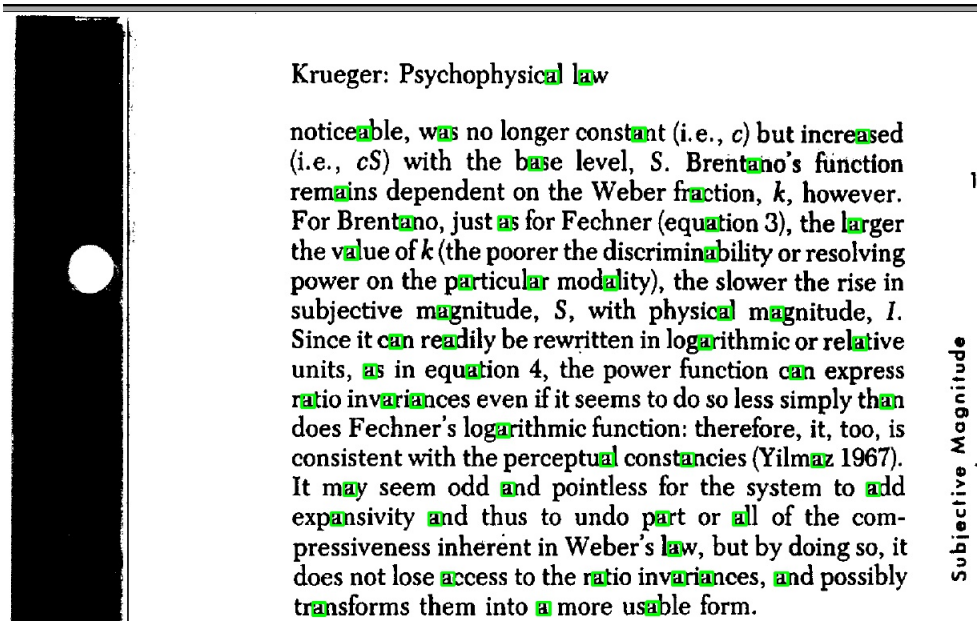

Figure 3: The image with detected "a"s.

Problem 3. (25 pts) You are given the binary image A and the structuring element B below (assume the origin of B is its center).

$$A = \begin{bmatrix} & & & & & & & \\ 1 & 1 & 1 & & & & & \\ 1 & 1 & 1 & 1 & & & 1 & \\ 1 & & & & 1 & 1 & 1 & 1 \\ & & & & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & & 1 & 1 & 1 & 1 \\ & & & & 1 & & & \\ & & 1 & 1 & 1 & 1 & 1 & 1 \\ & 1 & 1 & 1 & 1 & 1 & 1 & \\ & & & & & & & \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

1. Label (by hand) the 4-connected components, for foreground pixels (i.e. 1s).
   The two 4-connected components are distinguished by the label ID below and color :

$$A = \begin{bmatrix} & & & & & & & \\ 1 & 1 & 1 & & & & & \\ 1 & 1 & 1 & 1 & & & 2 & \\ 1 & & & & 2 & 2 & 2 & 2 \\ & & & & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & & 2 & 2 & 2 & 2 \\ & & & & 2 & & & \\ & & 2 & 2 & 2 & 2 & 2 & 2 \\ & 2 & 2 & 2 & 2 & 2 & 2 & \\ & & & & & & & \end{bmatrix}$$

2. Compute (by hand) the morphological opening of A with B.
   A morphological opening is erosion followed by dilation. The erosion step formed by structuring element B results in 1's only where the central element has adjacent 1's to

its left and right. The result is

$$A\Theta B = \begin{bmatrix} & & & & & & \\ & 1 & & & & & \\ & 1 & 1 & & & & \\ & & & & 1 & 1 & \\ & & & & 1 & 1 & \\ & 1 & 1 & 1 & 1 & 1 & 1 \\ & 1 & & & & 1 & 1 \\ & & & & & & \\ & & & 1 & 1 & 1 & 1 \\ & & 1 & 1 & 1 & 1 & \\ & & & & & & \end{bmatrix}.$$

Now we can apply the dilation of the eroded image with structuring element B, which yields left-right adjacent 1's wherever there is a 1 in the above eroded image. This results in the morphological opening of A with B:

$$(A\Theta B) \oplus B = \begin{bmatrix} & & & & & & & & \\ & 1 & 1 & 1 & & & & & \\ & 1 & 1 & 1 & 1 & & & & \\ & & & & & 1 & 1 & 1 & 1 \\ & & & & & 1 & 1 & 1 & 1 \\ & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ & 1 & 1 & 1 & & 1 & 1 & 1 & 1 \\ & & & & & & & & \\ & & & 1 & 1 & 1 & 1 & 1 & 1 \\ & & 1 & 1 & 1 & 1 & 1 & 1 & \\ & & & & & & & & \end{bmatrix}.$$

Problem 4. (25 pts) The video "fiveCCC.wmv" or "fiveCCC.avi" shows a target composed of five concentric contrasting circle (CCC) features.

- Read and display each image of the video, and print the frame number on the displayed image.
- Write a program that finds the five CCCs in each image, and marks each one with a cross hair or rectangle.
- Create an output video of your results and post it on YouTube.

Refer to file "HW2-Q4.py" for the code. I changed the frame to binary and then applied morphological closing then opening with a kernel of size $(2, 2)$. I used "cv2.connectedComponentsWithStats"

on the binary image and the reverse binary image to determine the centroids of the white components and black components, respectively. The combination of black and white components with centroids that were less than 1.3 pixels apart in both x and y were determined to be the CCCs. A "cv2.MARKER-CROSS" mark was added at the white centroid. Additionally, the frame number was added in the top left corner using "cv2.putText". The video can be seen here: https://youtu.be/6wwHubPuuGM. As I am having issues with my YouTube account (tied to the video I uploaded), you may find the video through my Google Drive (https://drive.google.com/file/d/1SqY4jwNKiZLvc8zHdU20QP69n180Hifz/view?usp=sharing) or for download through my github account (https://github.com/sterrab/ComputerVision/blob/master/HW2_Q4_video.avi).