Carson Stevens Homework 2

Sunday, September 22, 2019 5:43 PM

1. (25 pts) If a two-dimensional filter $w(x,y)$ is separable, that means that it can be written as the product of two one-dimensional filters; i.e., $w(x,y)=wx(x)wy(y)$. For example, the 2D Gaussian filter is separable, because $G(x,y)=12\pi\sigma 2e-x2+y22\sigma 2$ can be written as $G(x,y)=Gx(X)Gy(Y)=(12\pi\sigma e-x22\sigma 2)(12\pi\sigma e-y22\sigma 2)$. A 2D correlation of an image with a separable filter $w(x,y)$ can be can be computed by (1) computing a 1D correlation with $wy(y)$ along the individual columns of the input image, followed by (2) computing a 1D correlation with $wx(x)$ along the rows of the result from (1). In this way, the operation is much faster. Demonstrate this in Matlab using a 2D Gaussian filter on the "cameraman.tif" image[1], and show that the results from the two approaches are identical.
   Note that you can create Gaussian filters of any height and width using Matlab's "fspecial" function. For example, to create $Gx(x)$, use Gx = fspecial('gaussian', [1, width], sigma); I recommend setting the width to 6*sigma, so that the values of the Gaussian fall off close to zero at the edges of the filter. Use a large value of sigma, such as 20.
   Turn in (answersheet.pdf):

   1. An image of the result of the 2D correlation $G(x,y)\otimes I(x,y)$.
      White edges from poor cropping skills

   2. An image of the result of two 1D correlations, $Gx(x)\otimes(Gy(y)\otimes I(x,y))$.
      White edges from poor cropping skills

3. Compute the point-by-point difference of the two above images, and report the maximum value of the difference.

   Maximum Difference = 1 at row 1 column 27. $G(x,y)\otimes I(x,y) = 99$ and $Gx(x)\otimes(Gy(y)\otimes I(x,y)) = 98$. *This was this only pixel different.*

4. Time the two methods, using Matlab's tic and toc functions.

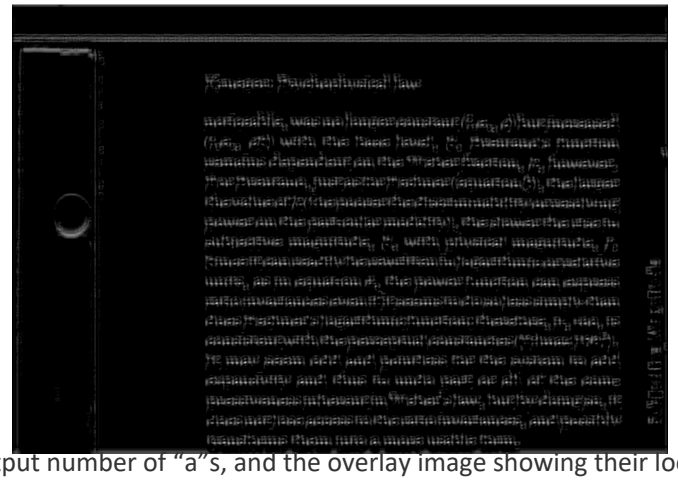
   $G(x,y)\otimes I(x,y)$ took 4.432900e-03 seconds to compute
   $Gx(x)\otimes(Gy(y)\otimes I(x,y))$ took 1.658700e-03 seconds to compute

2. (25 pts) Using the method of normalized cross-correlation, find all instances of the letter "a" in the image "textsample.tif" found at http://inside.mines.edu/~twilliams/courses/CV-507/textsample.tif. To do this, first extract a template subimage $w(x,y)$ of an example of the letter "a" (you can use Matlab's "imcrop" function). Then match this template to the image (you can use Matlab's "normxcorr2" function). Threshold the correlation scores (you will have to experiment with the threshold) so that you get 1's where there are "a"s and nowhere else. Now, you may get a small cluster of 1's where there is an "a" instead of a single 1. To avoid multiple detections, you can apply Matlab's "imregionalmax" function[2] to the scores image before thresholding.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. (answersheet.pdf) Describe your method of solution.

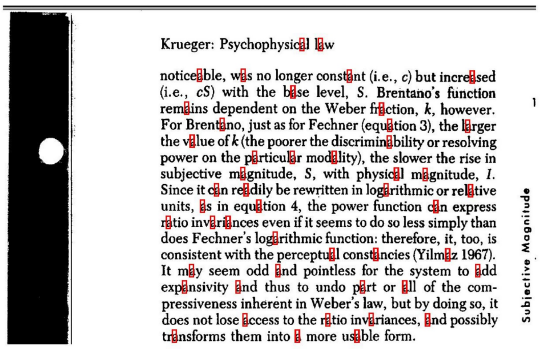
      First I extracted a template from the image. I used the ginput function to find an approximate (x,y) pixel location of an **a** *where I could get the least obstruction. Then I used the imcrop method to fine tune the* **a** *I wanted to crop for the template. I used the imregionalmax with the 8 connectivity to try and reduce some of the gaps within the text image. After that, I used the normxcorr2 function to find the corresponding pixel correlations for the template and the original image. After that, I iterated through the correlation matrix and drew rectangle at the locations where the values were greater than my threshold. I started with a threshold of 0.9, but it didn't get all the* **a***'s. I dropped that to 0.6 and then all of the* **a***'s were detected, but the count was too high (rectangles over the same* **a***'s). I tried adjusting further, but ended up losing* **a***'s before losing rectangles. I did a few things to compensate to atleast bring the number of extra ones down. I noticed that some of the letters drooped just over the tops of some of the* **a***'s that might get included in the template or cause confusing results. When I made the template taller to include the white space on top of the* **a***, the results got better. The next thing I did to try and fix the problem was decrease the width of the template. This cropped out letters that were dropping over and bleeding into that character. I also think this worked because it caused the template to be focused on a unique part of the shape that helped it be distinguished better. The last thing I did to improve the rectangle problem was increase the row I was on by the template width to try and skip over other rectangles once an* **a** *was detected. This method also improved the results. There were also* **a***'s that appeared vertically aligned which it didn't detect, but this makes sense because the template was for a horizontal* **a***.*

   2. (q2.m) Give the program listing, with comments.

3. (answersheet.pdf) Show the template image and the scores image.



4. (answersheet.pdf) The output number of “a”s, and the overlay image showing their locations.



There are 68 **a***'s (or atleast rectangles. I know there are still less, but when I decreased the threshold, I lost some of the* **a***'s and while still having too many rectangles in other parts).*

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

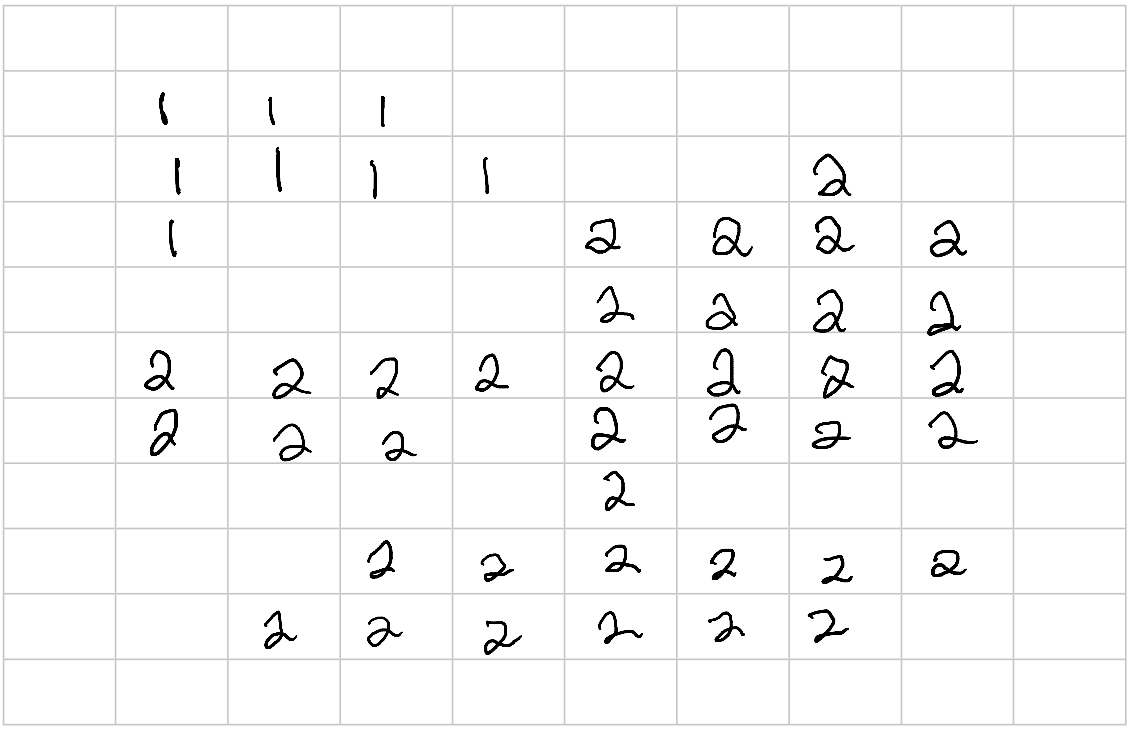
A:

	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		

B:

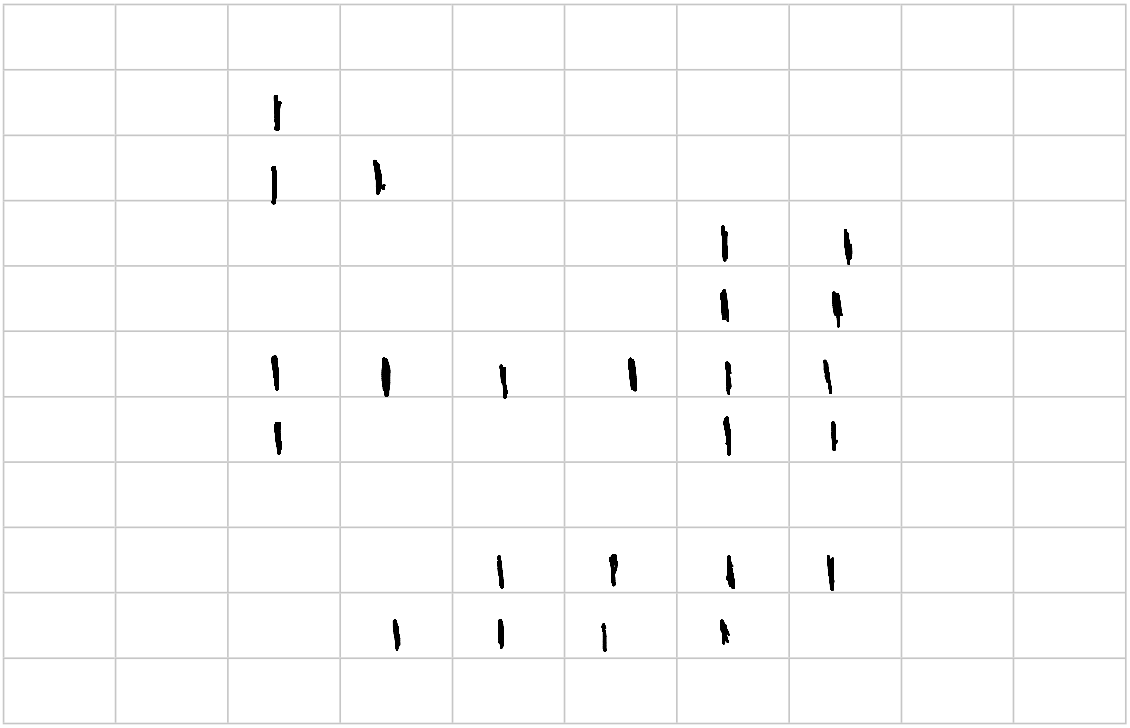
1	1	1

4 connected components

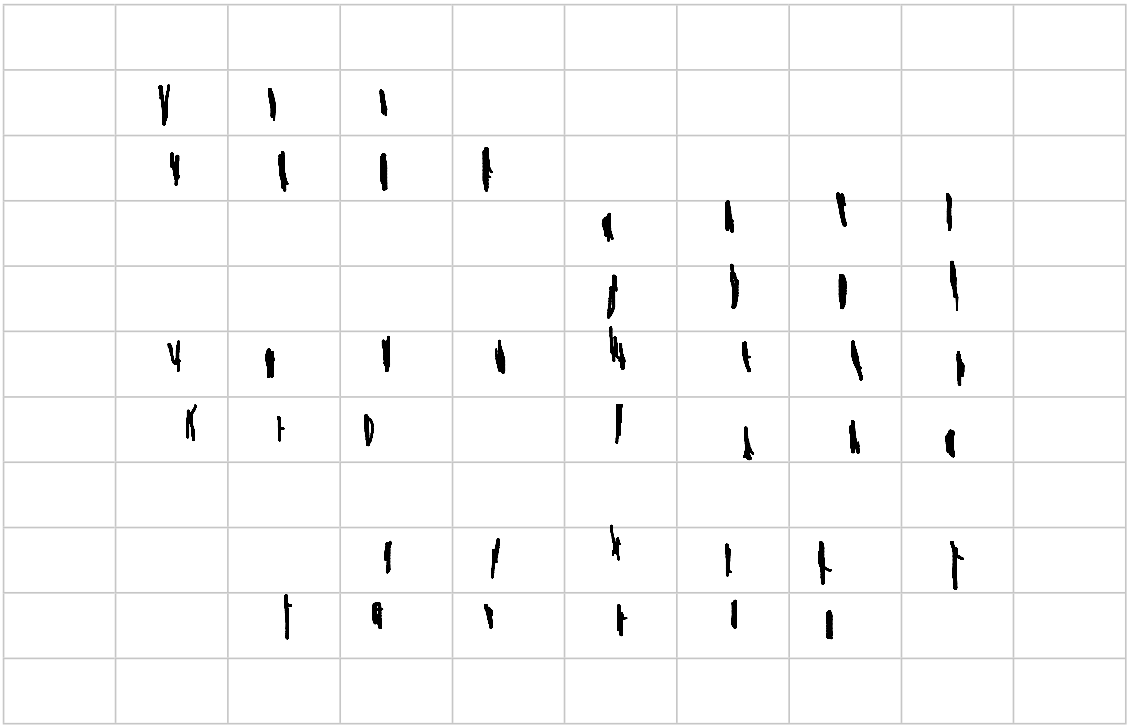


Morphological Opening of A and B (*Erosion followed by Dilation)*

Erosion



Dilation & Result of Opening



4.

- 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

The basic process was to iterate through each frame of the video and process the image to get a good recognition of each of the CCCs (The inner and outer circles). The black blobs were found through taking the BW image and then applying opening and eroding to the image and then taking its imcomplement. The white blobs were just converted to BW. Once each image is processes, both are sent bwlabel and finally regionprops to label each of the components and find the centroids of the clusters. From there, all the combinations of blob centroids from both groups are compared and if the centroids are less than the threshold distance (of 1 pixel), then the two centroids are paired and a rectangle is draw around the centroids. Then text for the frame is added to each image and then wrote to the output file. Finally the output is closed.

https://youtu.be/KA2QVGms8yQ