CS419 hw2

Doruk Benli

December 2023

Question 1

for making the darker areas appear purple and brighter areas appear bright orange, we have to add psuedo colors. Since dark areas has to be purple, we need blue color starting from lower values of intensity values. For purple, we also need red, but for bright orange we also need red so we have to select a red channel that co-exist with low and high intensity values. We also have to chose functions that non linear. for these cases we chose the following function for red

$$R(p) = 1 - e^{-0.015p}$$

This function will start from 0 and rapidly increase so darker regions will also be purple, not entirely blue. For green we will chose sigmoid, since we want it to exist in brighter areas and do not exist in less bright areas, For blue we want the opposite, we will take 1-G(p) because its functionality is inversed, we want it to exist in dark regions but not exist in bright regions.

$$G(p) = \frac{0.75}{1 + e^{-0.1(p-120)}} B(p) = 1 - \frac{0.75}{1 + e^{-0.1(p-120)}}$$

here we have 0,75 for scaling issue, we want to decrease the existence of green for making yellow, more orange. Here p is the pixel intensity values.

Question 2

In this question we are supposed to fill the hole in the tire. For this operation in grayscale, we use morphological hole filling. We will use reconstruction by erosion. for we take the max value in the original image and make it whole image. we will erode that and we will use base image as boundary. For reconstruction we start eroding our marker(which is image filled with max values) then keep on apply opening to the image with through this process, we also check if next marker == marker, if we have same 2 output, then we means we have react to the result of opening by reconstruction, so we stop the loop by using break.

1 Question 3

This question is about removing periodic noise from an image by using band reject filter. This is done by analyzing the fourier spectrum of the noisy image. we simply use numpy's fast fourier trasnform, and plot the fourier spectrum of the noisy image. From there we can vividly see the noise on vertical and horizontal axis, around the pixel value 100 from center. so we set our d0 = 100 in band reject fliter, and apply the gaussian band reject filter to the image, to reduce the ringing effect. The output image is noise free version of the noisy image.

Question 4

option-a

If we can distinguish objects we want to filter out directly then we can use opening, if we cannot distinguish them, then we may have to use other morphological operators such as tophat or bothat. Since we have images seperated by size, we may not need to directly use them, but if we have lightning changes, then we may have to use opening by reconstruction. For my implementation, since we take the mean of openings and serialize them, we will have a specific mean value for labeled picture with predefined S.E radiuses, we have specific results when we compare them with any distance metric

option-b

the images we are analyzing generally has oval shaped structures, which are stones. That's why we choose circular structuring elements. Otherwise if we were to chose square or rectangular structuring elements, this would cause errors and maybe unwanted results in the morphological operation, since some parts that should not be effected by dilation and erosion will be effected.

option-c

Its possible to implement granulometries with the following operations. Particularly, if we want to preserve the structure of large objects while removing the small ones, we can use opening by reconstructions. Here do not have big or small type of objects. Since in this approach, all stones types are given and they are of different sizes we can use opening. Since the stones are not mixed in size within each image, this comparison should yield clear and distinct profiles for each category, making classification more straightforward and computationally faster.

option-d

In my implementation, for all labeled types, i take take the opening with respect to different structuring element sizes, then take the means of opening and push them to an array, then i do this for unlabeled picture, then comapre the L1 norm L2 norm and chebyshev distance, this gave me correct labels for all the images.

Question 5

option-a

from all the filters, that we test on different noise levels and different kernel sizes with different color spaces, it appears that the best filter is marginal/scalar median filter in terms of mean squared error.

option-b

yes, relative performance depends on the image, as we work on different images, we get different MSE for each filter however the best filter is always the marginal median filter. It always performed better in the implementation. This can be seen from the table of option c with equal noise levels, we see MSE changes from image to image with same noise level. In addition, as image size increases algorithm also works slowly since its not optimized. This slow running process particularly happens in the 6th image since its larger than rest of the images.

Option-c

We apply all the filters for different noise levels from 5 percent to 10 percent to 20 percent. The results are recorded in Table 1. As we can see and also analyze from the code, as we increase the noise level we increase the mean squared error(MSE) of the filters.

option-d

here we analyze filters under different kernel sizes, ranging from 3, 5 and 7. Results are recorded in table 2. As we can see the results are persistent with what we have before. Again the lowest one is Marginal median filter. In addition as we increase our kernel size the error increases.

option-e

for additional color space, i have chosen the l*a*b color space and converted noisy images to that color space, then i also applied the same filters to the images. From the output of the code, we have clear bar charts that visualizes the result of different color space results. For all the images, l*a*b color space resulted in smaller amount of Mean Squared Error overall. Of course by using some other color space we can get different results than that. But changing the color space will change the mean squared error.

Table 1: MSE Values for Different Strategies and Noise Levels

Image	Median Lexicographical	Median Bitmix	Median Norm	Marginal Median
1 (5%)	18.08	20.27	17.63	15.04
1 (10%)	19.33	21.93	19.43	16.32
1 (20%)	22.71	25.98	23.28	19.61
2 (5%)	43.42	44.67	43.11	40.98
2 (10%)	46.03	47.73	46.15	43.36
2 (20%)	51.00	54.30	52.61	48.44
3 (5%)	25.79	27.45	24.96	23.66
3 (10%)	26.79	28.64	26.15	24.71
3 (20%)	28.75	31.77	29.20	26.83
4 (5%)	39.42	37.80	36.17	32.07
4 (10%)	41.46	40.35	38.76	34.77
4 (20%)	45.95	45.71	44.60	40.45
5 (5%)	25.30	26.52	24.59	23.30
5 (10%)	26.71	28.26	26.38	24.61
5 (20%)	29.73	32.04	30.39	27.69
6 (5%)	51.25	51.58	50.93	47.30
6 (10%)	52.16	52.84	52.24	48.33
6 (20%)	54.26	55.86	55.17	50.67
7 (5%)	11.20	13.28	11.37	9.33
7 (10%)	12.56	14.84	13.00	10.67
7 (20%)	15.94	18.11	16.55	13.54
8 (5%)	19.45	21.47	19.43	17.53
8 (10%)	21.33	23.80	21.66	19.28
8 (20%)	25.71	28.82	26.86	23.28
9 (5%)	25.76	27.46	24.76	22.85
9 (10%)	26.70	29.12	26.10	23.95
9 (20%)	28.68	31.86	29.18	26.53
10 (5%)	15.26	16.66	15.02	14.02
10 (10%)	17.22	19.04	17.44	15.83
10 (20%)	21.25	24.31	22.81	19.84

Table 2: MSE Values for Different Strategies and Kernel sizes

Image	Median Lexicographical	Median Bitmix	Median Norm	Marginal Median
1 (Kernel 3)	19.52	21.89	19.42	16.64
1 (Kernel 5)	29.04	43.27	28.88	24.28
1 (Kernel 7)	35.69	56.28	35.80	30.33
2 (Kernel 3)	45.83	47.63	46.13	43.28
2 (Kernel 5)	64.65	77.04	65.05	62.03
2 (Kernel 7)	75.34	88.85	75.32	72.80
3 (Kernel 3)	26.79	28.50	26.20	24.64
3 (Kernel 5)	34.65	44.52	33.86	31.86
3 (Kernel 7)	38.77	54.03	38.17	35.68
4 (Kernel 3)	41.61	40.43	39.25	34.87
4 (Kernel 5)	58.05	65.62	57.05	52.55
4 (Kernel 7)	66.07	75.85	66.28	61.83
5 (Kernel 3)	26.70	28.38	26.28	24.75
5 (Kernel 5)	36.83	47.48	35.84	33.70
5 (Kernel 7)	42.28	58.58	41.76	39.05
6 (Kernel 3)	52.16	52.85	52.24	48.33
6 (Kernel 5)	62.28	65.36	62.33	57.04
6 (Kernel 7)	66.79	71.75	66.91	61.01
7 (Kernel 3)	12.75	14.87	13.04	10.72
7 (Kernel 5)	21.35	29.62	21.72	19.12
7 (Kernel 7)	26.29	35.92	26.75	24.31
8 (Kernel 3)	21.46	23.69	21.66	19.32
8 (Kernel 5)	32.35	46.52	32.39	28.63
8 (Kernel 7)	39.99	60.75	40.11	35.54
9 (Kernel 3)	26.39	28.72	25.83	23.84
9 (Kernel 5)	35.02	46.23	33.07	29.90
9 (Kernel 7)	40.79	59.15	38.13	34.31
10 (Kernel 3)	17.30	19.16	17.51	15.90
10 (Kernel 5)	28.22	43.44	28.47	25.94
10 (Kernel 7)	35.35	58.41	35.90	32.86