CS419-Assignment 2

Ahmet Furkan Ün

December 16, 2023

Question 1

1. **Red Channel Function:**

$$R(p) = \begin{cases} 255 - 2p & \text{if } p < 128 \\ 255 & \text{otherwise} \end{cases}$$

The red channel colorization function intensifies the red component for darker areas and sets it to the maximum value for brighter areas. For pixel values p less than 128, the function $R(p)$ increases as p decreases, resulting in a stronger red color in darker regions. On the other hand, for pixel values greater than or equal to 128, the red component is set to the maximum value of 255, creating a consistent bright red component for brighter areas to seen as orange.

2. **Green Channel Function:**

$$G(p) = p$$

The green channel colorization function maintains a linear relationship between the grayscale pixel value (p) and the green color component. This results in a straightforward mapping where the green intensity is directly proportional to the grayscale intensity, producing a natural and linear gradient. Since purple has no green and orange needs green

3. **Blue Channel Function:**

$$B(p) = \begin{cases} 255 - 2p & \text{if } p < 128 \\ p - 128 & \text{otherwise} \end{cases}$$

The blue channel colorization function enhances the blue component for darker areas and creates a gradient for brighter areas. For pixel values p less than 128, the function $B(p)$ increases as p decreases, resulting in a stronger blue color in darker regions. For pixel values greater than or equal to 128, the function $B(p)$ introduces a linear gradient, contributing to a smooth transition from other colors to orange for brighter areas. Bright orange only need small amount of blue.

The graphical plots of these colorization functions were visualized for better understanding. The plots show the relationship between grayscale pixel values and corresponding color values for each channel.

The colorized image was obtained by applying the designed colorization functions to each channel of the grayscale image.

To sum up, the non-linear colorization functions I generated effectively transformed the grayscale image into a pseudo-colored image, with dark areas appearing purple and bright areas appearing bright orange.

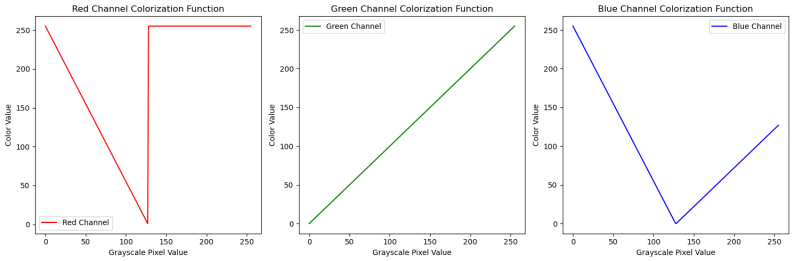


Figure 1: Colorization Functions

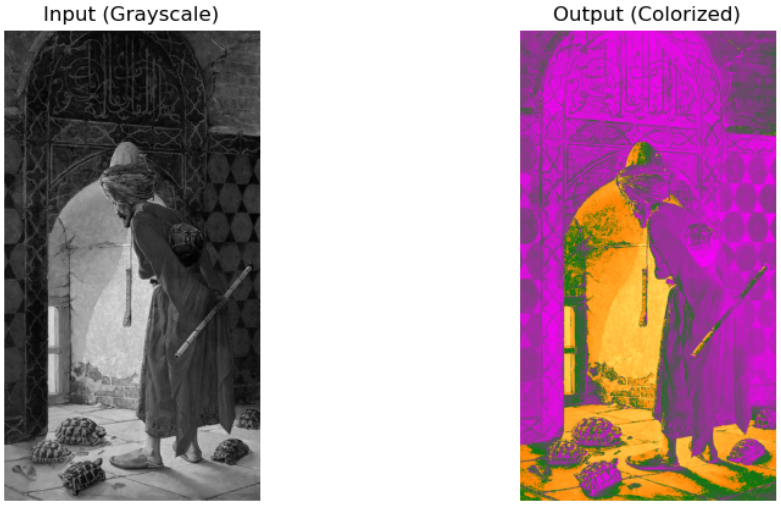


Figure 2: Input and Output Images

Question 2

For this question, I implemented the grayscale erosion operator. For each pixel, I considered its neighbors and gets the minimum of them.

After implementing the erosion, I started implementing reconstruction by erosion. What this does is that it erodes the image but until the marker. So we can not erode deeper than the marker. We need to erode and reconstruct until it becomes idempotent.

For hole filling algorithm, I referenced the P. Soille's book. He proposed the solution to hole filling as creating a marker with original values at the edges of the image, and max for the remaining parts. If we erode this marker and reconstruct it using original image, we will reach the filled version of it.

My fill(image) function first creates the marker by copying original values for borders and max value for the remaining parts. After that the function apply reconstruction by erosion to the marker and using the original image for reconstruction. It repeats this until it reaches idempotence. The resulting image is filled version of the original image.

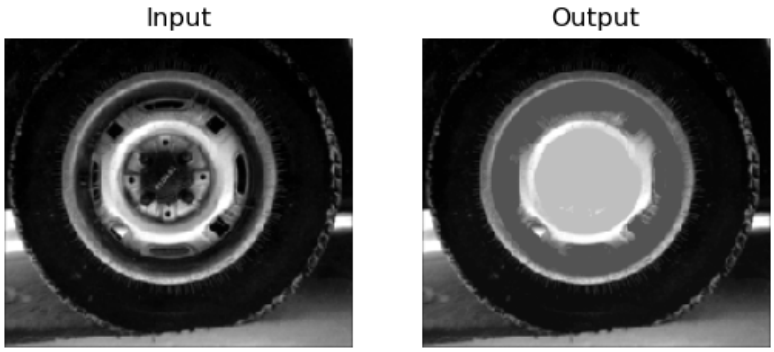


Figure 3: Input and Output Images

Question 3

In this task, we are trying to restore a corrupted aerial image of Pompeii, Italy, which suffered from sinusoidal noise caused by electrical interference. To achieve this, I used a Fourier transform to analyze the frequency components present in the image. Subsequently, I implemented a band reject filter in the frequency domain to selectively eliminate unwanted frequencies responsible for the noise. The result is a denoised version of the original image, effectively enhancing the visual clarity of the Pompeii aerial view.

The first step involved computing the Fourier transform of the image. The Fourier transform provides insight into the frequency composition of the image. By visualizing the magnitude spectrum, it became apparent that sinusoidal noise was present, manifesting as distinct frequencies. Two circles on the Fourier transform plot indicated the range of frequencies targeted for removal.

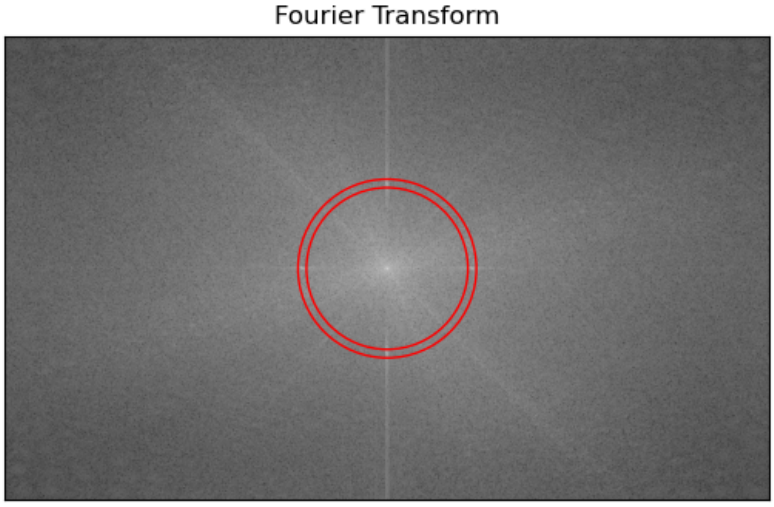


Figure 4: Magnitude Spectrum - Pompeii

The core of the denoising process lay in the circular band reject filter. This filter selectively blocked frequencies within the specified range, effectively isolating and removing the sinusoidal noise from the Fourier transform. The circular mask, when applied to the Fourier transform,

acted as a spatial filter, attenuating unwanted frequencies. After the filter was applied, the inverse Fourier transform was performed to obtain the denoised image.

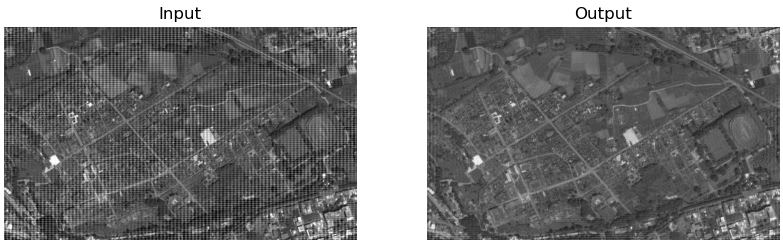


Figure 5: Input and Output Images

Question 4

The objects of interest are sometimes brighter and sometimes darker with respect to their surrounding. In order to address this problem, at first I tried to apply histogram equalization. However it makes the color differences of the objects significant. So I decided not to use it. However opening is not sufficient only on its own because the objects of interest are sometimes brighter and sometimes darker with respect to their surrounding. So I decided to use opening by reconstruction since it preserve the details more and does not effected by the color differences of the objects.

To determine the shapes of the structuring elements, I follow a similar approach to Gonzales and Wood's approach in their book. I calculated the surface values of each sample applied to opening by reconstruction and for different structuring element sizes. Then I calculated the differences of the subsequent surface values. This calculation gave as the peaks where size of most of the elements is that structring element.

Upon analyzing the graphical representations (Figures 6-8), I have determined the sizes of the gravels in the unlabeled images to be 140, 45, and 200, respectively. To predict the label of these unlabeled images, a crucial step involves applying opening by reconstruction with structuring elements of the aforementioned sizes. The objective is to compare the resulting surface areas with those obtained by applying the same procedure to the labeled images. Our preferred metric for this comparison is the cityblock (Manhattan) distance, chosen for its capacity to reflect the overall difference in all dimensions. Cityblock distance is particularly apt for this task as it considers the total dissimilarity between corresponding elements of vectors, prioritizing the comprehensive evaluation of differences over spatial distances. In this context, where capturing dissimilarity across all dimensions is crucial, cityblock distance emerges as a more appropriate metric. After applying the opening by reconstruction and calculating cityblock distances, I meticulously compared the results, enabling us to predict the labels accurately. By selecting the label associated with the smallest cityblock distance, I could pinpoint the closest match for each unlabeled sample. Notably, our approach achieved a high level of accuracy, correctly predicting the labels for all unlabeled samples. This underscores the effectiveness of our methodology in leveraging size-specific structuring elements and cityblock distance to achieve precise and reliable predictions in this gravel labeling task.

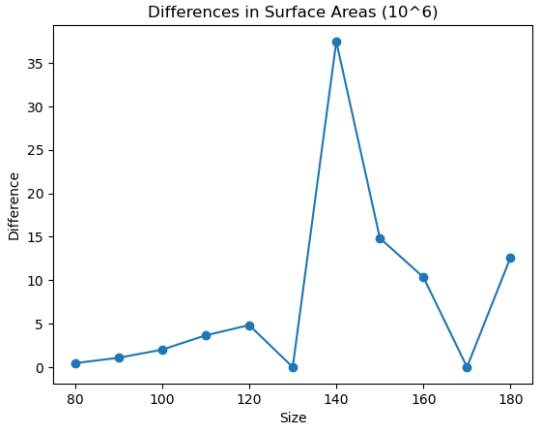


Figure 6: Surface Area Difference of Structuring Elements - Labeled Image 1

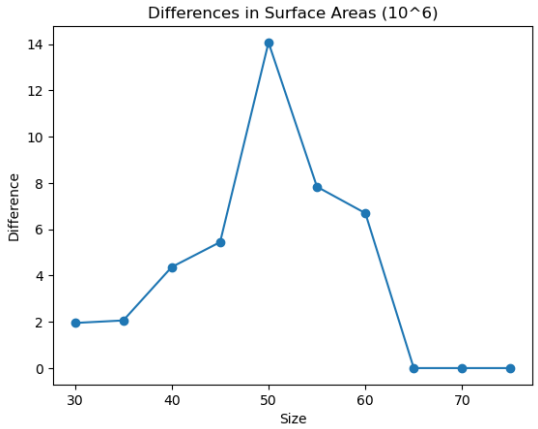


Figure 7: Surface Area Difference of Structuring Elements - Labeled Image 2

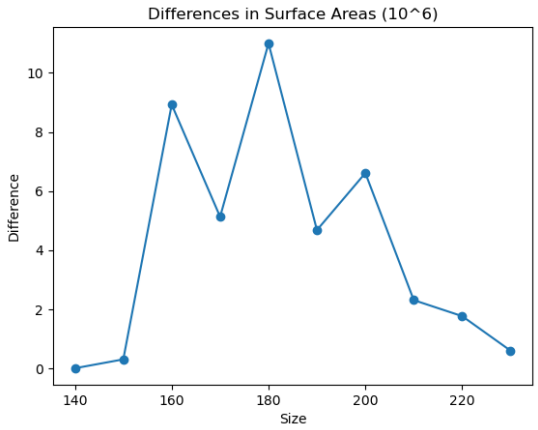


Figure 8: Surface Area Difference of Structuring Elements - Labeled Image 3

Question 5

(a) Implementation Details of Median Filtering Strategies

(i) Marginal Strategy

In this approach, we need to get the medians of all channel values of the neighbors separately. To do this, for every pixel, the function traverses the neighbors based on the filter size. After traversing, it stores every channels value in separate arrays and putting their medians into current pixel's channel values. By doing that we can denoise the images.

(ii) Vector Strategy - Lexicographical Ordering

In Vector Strategy, we need to get the medians of color vectors of the neighbors separately. But, to find the median, we need to propose an ordering relation. One of the ordering relations is Lexicographical ordering. It basically compares the vectors by their first element, if they are equal compares the second and lastly third. To do this, the function traverses the neighbors and stores their color vectors in a list. After sorting them lexicographically, it picks the value in the middle and puts it instead of the color vector of the current pixel. By doing that we can denoise the images.

(iii) Vector Strategy - Bitmix Ordering

Mostly same as the previous one since they are both using Vector Strategy. The only difference is the ordering relation. In this ordering relation we need to mix the bits of the colors into one long bit sequence. Every color is 8 bits so our mixed bits will be 24 bits long. If we consider the bits of the colors as
R1 - R2 - R3 - R4 - R5 - R6 - R7 - R8
G1 - G2 - G3 - G4 - G5 - G6 - G7 - G8
B1 - B2 - B3 - B4 - B5 - B6 - B7 - B8
Mixed bits will be:
R1 - G1 - B1 - R2 - G2 - ... - R8 - G8 - B8
To do this, the function traverses the neighbors and stores their color vectors in a list and converts them into bitmix sequences. After sorting them, it picks the value in the middle and puts it instead of the color vector of the current pixel. By doing that we can denoise the images.

(iv) Vector Strategy - Norm Based Ordering

Mostly same as the previous two versions since they are all using Vector Strategy. The only difference is the ordering relation. In this ordering relation we need to calculate the norm values of color vectors to compare them.

$$\text{norm} = \sqrt{R^2 + G^2 + B^2}$$

To do this, the function traverses the neighbors and stores their color vectors in a list and converts them into norms of them. After sorting them based on the calculated norms, it picks the value in the middle and puts it instead of the color vector of the current pixel. By doing that we can denoise the images.

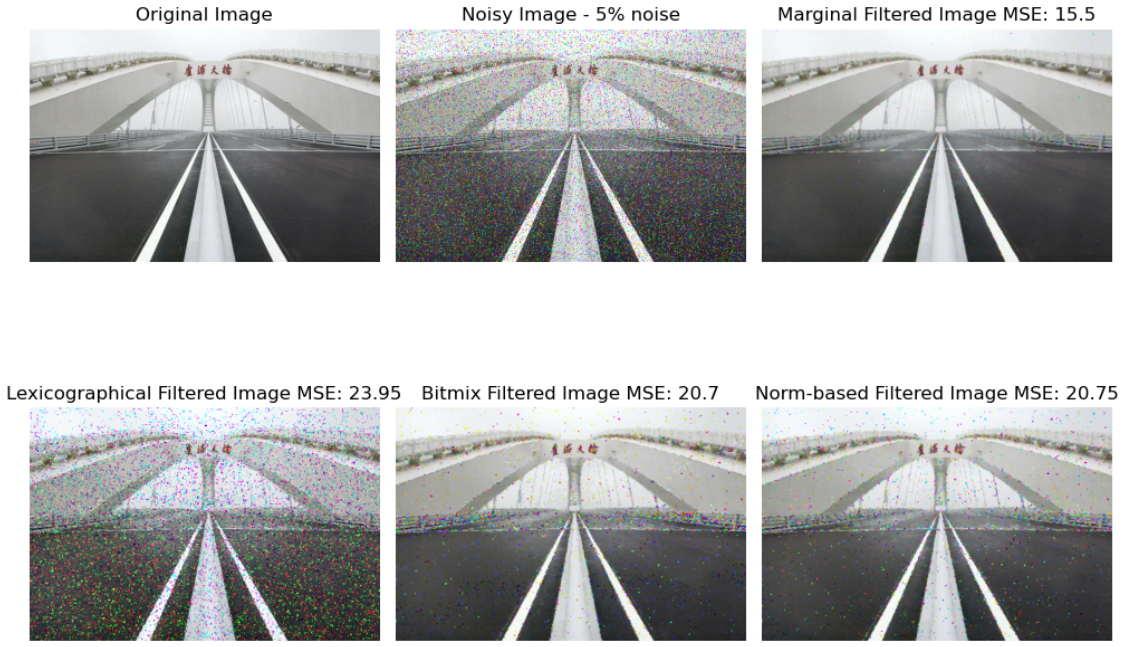


Figure 9: Differences of Median Filtering Strategies

(b) Comparison of Median Filtering Strategies

(i) Which one filters the images best, and by how much?

In order to determine the performances of the strategies, I created 90 different test case for each strategy. I tried 1%, 5% and 30% noise rates and filters of sizes 3, 7 and 15 on 10 different RGB image. I stored the results in a pandas DataFrame. By calculating the mean MSE values of each strategy I obtained the following table.

Table 1: Mean MSE for Different Strategies

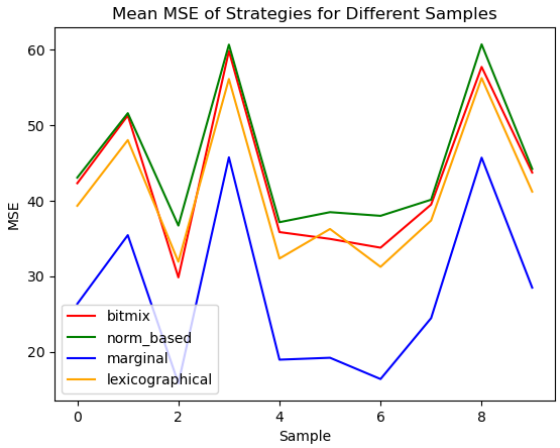
Strategy	Mean MSE
Marginal	27.654096
Vector (Lexicographical)	40.994409
Vector (Bitmix)	42.866167
Vector (Norm Based)	45.055061

If we look at the mean MSE values, we can say that in overall, Marginal strategy performs significantly bettter than the others.

(ii) Does their relative performance depend on the image?

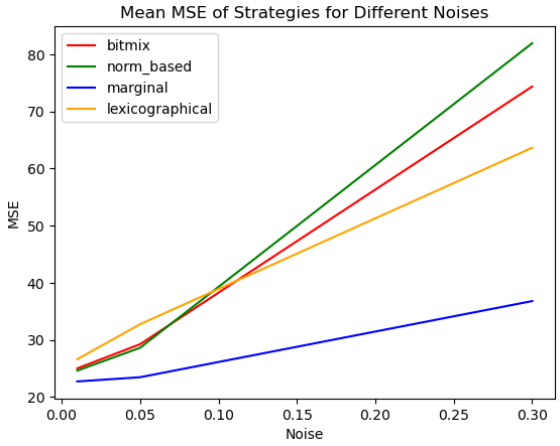
If we look at the graph above, we can see the mean MSE values of different strategies for different sample images. For every image, I tried 1%, 5% and 30% noise rates and filters of sizes 3, 7 and 15 and calculated the mean MSE of each trial. Again the Marginal Strategy significantly performed the best. For almost all images, the relative order of performance seems remained same except for sample 2 and 5. For sample 2 and 5, bitmix ordering performed better than lexicographical

ordering. However, by looking at the whole we can say that the relative performance remained almost same.



(iii) Does their relative performance depend on level/correlation of noise?

By looking at the graph, we can say that lexicographical ordering become better by the increasing noise compared to norm based and bitmix orderings. Again the marginal strategy significantly performed better for all noise levels. However, we need to focus our attention to the difference in the relative ordering of the vector strategy median filter approaches compared to the overall performance. Apparently, they bitmix and norm based orderings are great for a decent amount of noise. However when the noise is great, the lexicographical ordering performs slightly better than other vector strategy median filtering approaches.



In terms of correlation, I did not created automated testing setup but I tried some tests and observed that in correlated noise, meaning that we have same noise in every channel, they performed equally good as can be seen from Figure 10.

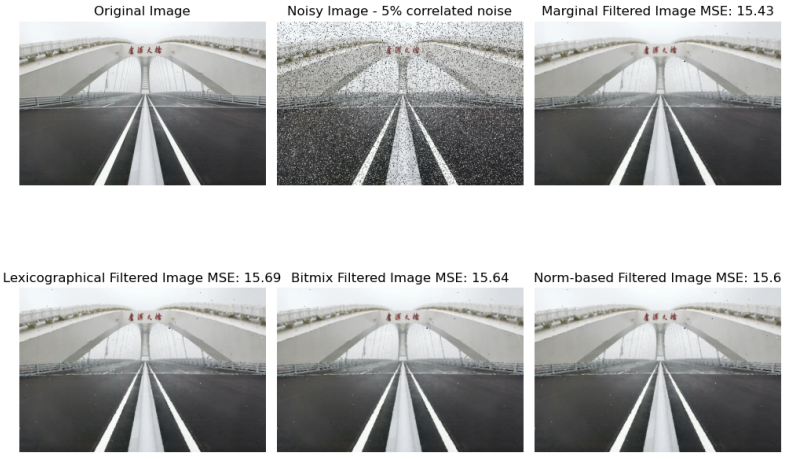
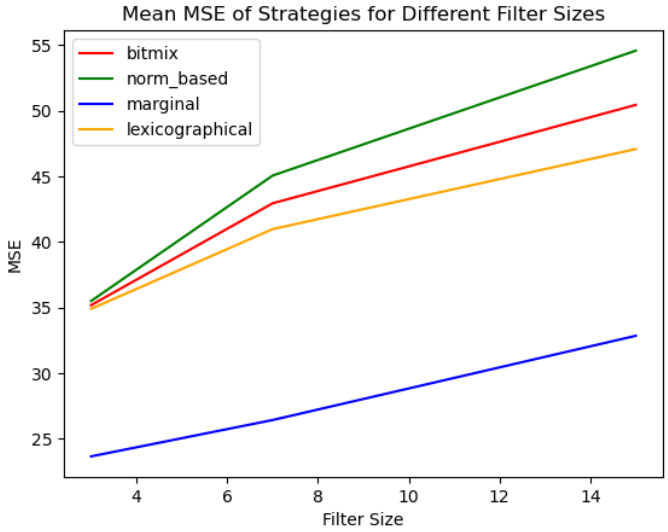


Figure 10: Differences of Median Filtering Strategies for Correlated Noise

(iv) Does their relative performance depend on filter size?

By looking at the graph, we can say that filter size have no effect on the relative performance of different median filtering approaches. Again the marginal strategy significantly performed better for all filter sizes. Also, the relative ordering here is the same as the overall performance. However, we can say that the performance of the strategies getting worse when the filter size is increased. This is a common behaviour for all of the approaches.



(v) Does their relative performance depend on the color space?

For all approaches, I implemented and tested the RGB versions. However, the performance of these strategies can indeed depend on the color space used. In the RGB color space, the marginal strategy seems to work well since each channel represents a primary color. However, it might miss out on some detailed color relationships. On the other hand, the vector strategy is useful for keeping

color information intact, especially when colors are related to each other. In the YUV color space, the marginal strategy is good for the intensity (Y) channel but might not capture chrominance relationships effectively. The vector strategy in YUV seems better at handling the connection between intensity and color information, offering a more comprehensive approach. Moving to the LAB color space, the marginal strategy is known for maintaining perceptual uniformity, ensuring that color differences align with human perception. Yet, it might not be the best at capturing intricate color correlations. Meanwhile, the vector strategy in LAB can be advantageous as it keeps both color and intensity relationships, making it suitable for cases where these details matter.