

|  |  |
| --- | --- |
| Digital Image Processing  Report |  |
| By:Belal Khaled(2136873) **Assignment 1**  Assignment 1.A Manual Code: Libraries Code: **from PIL import Image**  **import numpy as np LenaGray = Image.open(r'LenaGray.jpg')**  **from PIL import Image LenaGray.show()**  **img0=Image.open(r"LenaGray.jpg") PepperGrey=Image.open(r'PeppersGrey.jpg')**  **img1=Image.open(r"PeppersGrey.jpg") PepperGrey.show()**  **img0.show()**  **img1.show()** |  |

Output 1.A



**PepperGrey.jpg**

**LenaGray.jpg**

Assignment 1.B

# Manual Code: Libraries Code:

**image0\_pixels=np.array(img0.getdata()) j = Image.new('L' , (256,256))**

**image1\_pixels=np.array(img1.getdata()) for y in range(256):**

**image0\_pixels=image0\_pixels.reshape(256,256) for x in range (256):**

**image1\_pixels=image1\_pixels.reshape(256,256) if y <= 128:**

**image0\_pixels=image0\_pixels[:128, :] gs\_value = LenaGray.getpixel((x,y))**

**image1\_pixels=image1\_pixels[128:, :] j.putpixel((x,y) , gs\_value)**

**image1\_2=np.concatenate((image0\_pixels,image1\_pixels),axis=0) else:**

**final\_result=Image.fromarray(image1\_2) gs\_value = PepperGrey.getpixel((x,y))**

**final\_result.show() j.putpixel((x,y) , gs\_value)**

**final\_result=final\_result.convert("L") j.save('J.jpg')**

**final\_result.save("J.jpg") j.show()**

Output 1.B

A person wearing a hat and a bell pepper

Description automatically generated

**J.jpg**

Assignment 1.C

# Manual Code: Libraries Code:

**img3=Image.open("J.jpg") j = Image.open('j.jpg')**

**k\_pixels=np.array(img3.getdata()) k = Image.new('L', (256, 256))**

**size=img3.size for y in range(256):**

**k\_pixels=k\_pixels.reshape(size[0],size[1]) for x in range(256):**

**part1=k\_pixels[:256//2, :] gs\_value = j.getpixel((x, y - 128))**

**part2=k\_pixels[256//2:, :] k.putpixel((x, y), gs\_value)**

**partAll=np.concatenate((part2,part1),axis=0) k.save('K.jpg')**

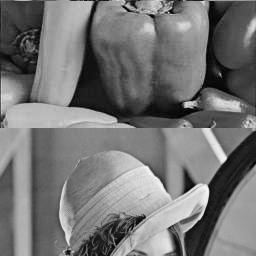
**k\_final=Image.fromarray(partAll) k.show()**

**k\_final=k\_final.convert("L")**

**k\_final.save("K.jpg")**

**k\_final.show()**

Output 1.C



**K.jpg**

|  |  |
| --- | --- |
| Summary Of Assignment 1  Assignment 1.A  **In A section we just used the pillow library**  **to load the image on each code.** |  |
| Assignment 1.B    **In B section in the manual code we load the both images and assigned them to two Variable. then we took the data(pixels) and assigned them to one dimension array by numpy Library then again we reshaped the array to two dimension array, splitted the first half of the 2d array From the first image and the second half of the 2d array in the second image and concatenate them(numpy) then convert it to one image by** **pillow library.**    Assignment 1.C  **In C the same as section B but the opposite in**  **the splitting processes.**  Top of Form |  |
| **Assignment 2**  Assignment 2.i Manual Code: **#image negative function**  **def apply\_image\_negative(input\_array):**  **input\_array = np.asarray(input\_array, dtype=np.uint8)**  **negative\_array = 255 - input\_array**  **return negative\_array**  **img3=Image.open("LenaGrayNoisy.jpg")**  **size=img3.size**  **image3\_pixels=np.array(img3.getdata())**  **image3\_pixels=image3\_pixels.reshape(size[0],size[1])**  **img4=Image.open("PeppersGreyNoisy.jpg")**  **imag4\_pixels=np.array(img4.getdata())**  **imag4\_pixels=imag4\_pixels.reshape(size[0],size[1])**  **x=apply\_image\_negative(image3\_pixels)**  **za=apply\_image\_negative(imag4\_pixels)**  **x=Image.fromarray(x)**  **za=Image.fromarray(za)**  **x.save("Negative lena.jpg")**  **x.show()**  **za.save("Negative peppers.jpg")**  **za.show()** Libraries Code: **LenaGrayNoisy = Image.open('LenaGrayNoisy.jpg')**  **LenaGrayNoisy.show()**  **PeppersGrayNoisy = Image.open('PeppersGreyNoisy.jpg')**  **PeppersGrayNoisy.show()**  **LenaGrayNoisy\_Negative = Image.new('L' , (256,256))**  **PeppersGrayNoisy\_Negative = Image.new('L' , (256,256))**  **#Applying Negative To PeppersGrayNoisy:**  **for y in range(256):**  **for x in range(256):**  **gs\_value = PeppersGrayNoisy.getpixel((x,y))**  **PeppersGrayNoisy\_Negative.putpixel((x,y) , 255 - gs\_value)**  **PeppersGrayNoisy\_Negative.save('NegativePeppers.jpg')**  **PeppersGrayNoisy\_Negative.show()**  **#Applying Negative To LenaGrayNoisy:**  **for y in range(256):**  **for x in range(256):**  **gs\_value = LenaGrayNoisy.getpixel((x,y))**  **LenaGrayNoisy\_Negative.putpixel((x,y) , 255 - gs\_value)**  **LenaGrayNoisy\_Negative.save('NegativeLena.jpg')**  **LenaGrayNoisy\_Negative.show()**  Output 2.i |  |
| **LenaGrayNoisy.jpg** | **PeppersGreyNoisy.jpg** |

A group of peppers with small dots

Description automatically generated

**NegativePeppers.jpg**

**NegativeLena.jpg**

**Assignment 2**

Assignment 2.ii

# Manual Code:

**#image negative function**

**def apply\_image\_negative(input\_array):**

**input\_array = np.asarray(input\_array, dtype=np.uint8)**

**negative\_array = 255 - input\_array**

**return negative\_array**

**#Median filter function:**

**def apply\_median\_filter(input\_array):**

**rows, cols = input\_array.shape**

**filtered\_array = np.zeros\_like(input\_array, dtype=np.uint8)**

**for i in range(1, rows - 1):**

**for j in range(1, cols - 1):**

**window = input\_array[i - 1:i + 2, j - 1:j + 2].ravel()**

**filtered\_array[i, j] = np.median(window)**

**return filtered\_array**

**img3=Image.open("LenaGrayNoisy.jpg")**

**size=img3.size**

**image3\_pixels=np.array(img3.getdata())**

**image3\_pixels=image3\_pixels.reshape(size[0],size[1])**

**img4=Image.open("PeppersGreyNoisy.jpg")**

**imag4\_pixels=np.array(img4.getdata())**

**imag4\_pixels=imag4\_pixels.reshape(size[0],size[1])**

**x=apply\_image\_negative(image3\_pixels)**

**y=apply\_median\_filter(image3\_pixels)**

**xy=apply\_median\_filter(x)**

**y.save("Median lena.jpg")**

**y.show()**

**xy.save("Both lena.jpg")**

**xy.show()**

**zs.save("Median peppers.jpg")**

**zs.show()**

**zg.save("Both peppers.jpg")**

**zg.show()**

**za=apply\_image\_negative(imag4\_pixels)**

**zs=apply\_median\_filter(imag4\_pixels)**

**zg=apply\_median\_filter(za)**

**y=Image.fromarray(y)**

**xy=Image.fromarray(xy)**

**zs=Image.fromarray(zs)**

**zg=Image.fromarray(zg)**

**Assignment 2**

Assignment 2.ii

# Libraries Code:

**import statistics**

**LenaGrayNoisy\_Median = Image.new('L' , (256,256))**

**PeppersGrayNoisy\_Median = Image.new('L' , (256,256))**

**#Function that returns 3 × 3 window:  
def window3(pixel, width, height):**

**x = pixel[0]**

**y = pixel[1]**

**if x == 0 or y == 0 or x == width - 1 or y == height - 1:**

**raise ValueError**

**return [(x-1, y-1), (x, y-1), (x+1, y-1),**

**(x-1, y), (x,y), (x+1, y),**

**(x-1, y+1), (x, y+1), (x+1, y+1)]**

**#Applying Median Filter on the original images(LenaGrayNoisy.jpg and PeppersGreyNoisy.jpg**)**:**

**for y in range(256):**

**for x in range(256):**

**try:**

**pixelwindow = window3((x, y), 256, 256)**

**Lenawindowpixels = [LenaGrayNoisy.getpixel(i) for i in pixelwindow]**

**median = statistics.median(Lenawindowpixels)**

**LenaGrayNoisy\_Median.putpixel((x,y), median)**

**Pepperswindowpixels = [PeppersGrayNoisy.getpixel(i) for i in pixelwindow]**

**median = statistics.median(Pepperswindowpixels)**

**PeppersGrayNoisy\_Median.putpixel((x,y), median)**

**except ValueError:**

**LenaGrayNoisy\_Median.putpixel((x,y), 0)**

**PeppersGrayNoisy\_Median.putpixel((x,y), 0)**

**LenaGrayNoisy\_Median.save('MedianLena.jpg')**

**PeppersGrayNoisy\_Median.save('MedianPeppers.jpg')**

**LenaGrayNoisy\_Median.show()**

**PeppersGrayNoisy\_Median.show()**

**#Applying Median Filter on the Negative images(NegativeLena.jpg and NegativePeppers.jpg):**

**for y in range(256):**

**for x in range(256):**

**try:**

**pixelwindow = window3((x, y), 256, 256)**

**Lenawindowpixels = [LenaGrayNoisy\_Negative.getpixel(i) for i in pixelwindow]**

**median = statistics.median(Lenawindowpixels)**

**LenaGrayNoisy\_Median.putpixel((x,y), median)**

**Pepperswindowpixels = [PeppersGrayNoisy\_Negative.getpixel(i) for i in pixelwindow]**

**median = statistics.median(Pepperswindowpixels)**

**PeppersGrayNoisy\_Median.putpixel((x,y), median)**

**except ValueError:**

**LenaGrayNoisy\_Median.putpixel((x,y), 0)**

**PeppersGrayNoisy\_Median.putpixel((x,y), 0)**

**LenaGrayNoisy\_Median.save('MedianNegativeLena.jpg')**

**PeppersGrayNoisy\_Median.save('MedianNegativePeppers.jpg')**

**LenaGrayNoisy\_Median.show()**

**PeppersGrayNoisy\_Median.show()**

Output 2.ii

A close-up of a pile of vegetables

Description automatically generatedA person wearing a hat

Description automatically generated



A close-up of several peppers

Description automatically generated

**MedianPeppers.jpg**

**MedianLena.jpg**

A group of peppers with small dots

Description automatically generated

A statue of a person wearing a hat

Description automatically generatedA close-up of several peppers

Description automatically generated

**MedianNegativeLena.jpg**

**MedianNegativePeppers.jpg**

Summary Of Assignment 2

Assignment 2.i

**We first showed the images that we will work on them , then we Defined the Negative Transformation Function basically the function takes an input array and subtract from each pixel the value 255 then we return the negative array, we imported the LenaGrayNoisy.jpg and PeppersGreyNoisy.jpg image to the function and showed the result ,but before that we converted the array(numpy) to image by using pillow library.**

Assignment 2.ii

**We defined the median filter function to reduce noise by replacing each pixel value with the median value of its neighboring pixels basically the function Iterates over each pixel in the interior of the input array ,defines a 3x3 window around each pixel and flattens it to a 1D array using ravel() Calculates the median of the window and assigns it to the corresponding pixel in the filtered\_array and returns it as 2d array , we imported the image array of both PeppersGreyNoisy.jpg and LenaGrayNoisy.jpg to the function and we converted them to images by using pillow library.**

**Discussion of how the results of the different operators are:**

Negative 2.i

A statue of a person wearing a hat

Description automatically generated **Applying the negative operator to a black color would result in a white color, and applying it to a white color would turn it black(on a scale). It's like flipping the colors on a photographic negative. This technique is often used for various purposes, including artistic effects, enhancing visibility, or inverting colors for accessibility reasons**. **lock at the example below:**

**The picture after**

**applying the negative operator.**

**Black and white picture.**

Median filter 2.ii

**A median filter is a common image processing technique used to reduce noise in images while preserving edges and details.**

**Noise Reduction: The median filter helps in reducing salt-and-pepper noise or random isolated bright or dark pixels in an image without significantly blurring the edges.**

**Lock at the example below:**

A person wearing a hat

Description automatically generated

**The picture after**

**applying the median filter operator.**

**A noisy picture.**



**References:**

https://www.geeksforgeeks.org/negative-transformation-of-an-image-using-python-and-opencv/

https://medium.com/@florestony5454/median-filtering-with-python-and-opencv-2bce390be0d1