*** Image enhancement in frequency domain**

Task 1.ipynb U

Lab 6 > Lab6_CSE457 > Task 1.ipynb > ...

+ Code + Markdown | Run All Clear All Outputs | Outline ···

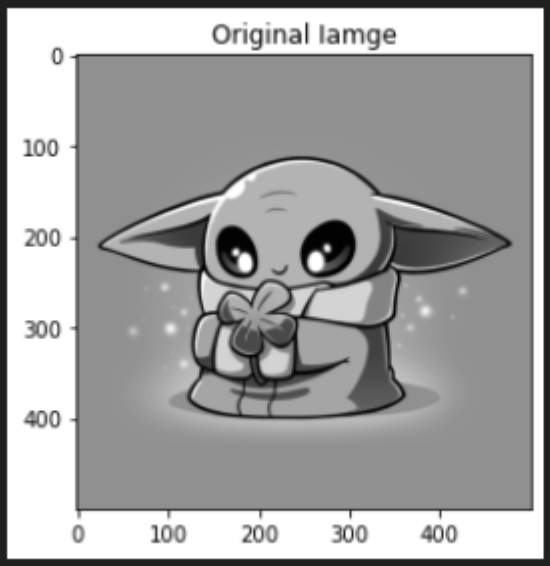
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

[4]

```
image = cv2.imread('pic.jpg',0)
plt.imshow(image,'gray'), plt.title('Original Iamge')
```

[5]

```
(<matplotlib.image.AxesImage at 0x1a5fd70ee20>,
 Text(0.5, 1.0, 'Original Iamge'))
```



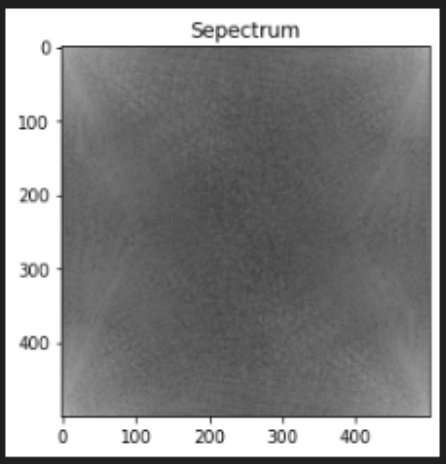
Task 1.ipynb U

Lab 6 > Lab6_CSE457 > Task 1.ipynb > ...

+ Code + Markdown | Run All Clear All Outputs | Outline ...

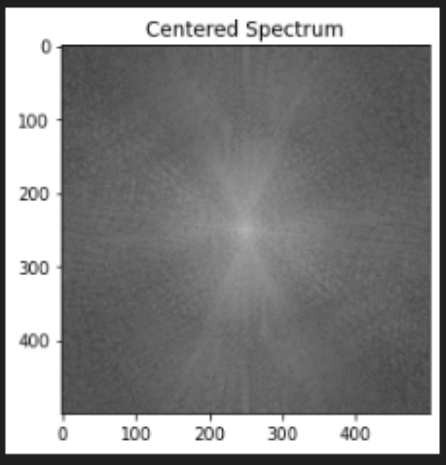
[7]
```
image_c2 = np.fft.fft2(image)
plt.imshow(np.log(1+np.abs(image_c2)),'gray'), plt.title("Sepectrum")
```

```
(<matplotlib.image.AxesImage at 0x1a5fdc460d0>, Text(0.5, 1.0, 'Sepectrum'))
```



[8]
```
image_c3 = np.fft.fftshift(image_c2)
plt.imshow(np.log(1+np.abs(image_c3)),'gray'), plt.title('Centered Spectrum')
```

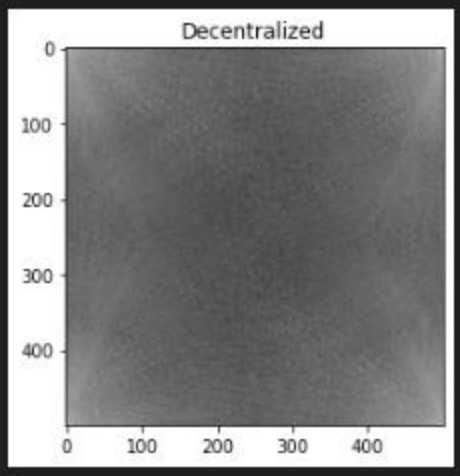
```
(<matplotlib.image.AxesImage at 0x1a5fe276070>,
 Text(0.5, 1.0, 'Centered Spectrum'))
```



```
image_c4 = np.fft.ifftshift(image_c3)
plt.imshow(np.log(1+np.abs(image_c4)),'gray'), plt.title('Decentralized')
```

[10]

```
(<matplotlib.image.AxesImage at 0x1a5fdac3c10>,
 Text(0.5, 1.0, 'Decentralized'))
```



*** Low pass and High pass filters**

Task 2.ipynb U

Lab 6 > Lab6_CSE457 > Task 2.ipynb > ...

+ Code + Markdown | Run All Clear All Outputs | Outline ...

```
[1]
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

```
[2]
image = cv2.imread('pic.jpg',0)
original = np.fft.fft2(image)
center = np.fft.fftshift(original)
```

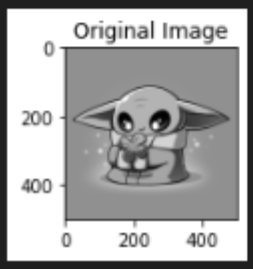
```
[3]
plt.figure(figsize=(6.4*5, 4.8*5), constrained_layout=False)
```

```
<Figure size 2304x1728 with 0 Axes>
```

```
<Figure size 2304x1728 with 0 Axes>
```

```
[4]
plt.subplot(131), plt.imshow(image, 'gray'), plt.title("Original Image")
```

```
(<AxesSubplot:title={'center':'Original Image'}>,
 <matplotlib.image.AxesImage at 0x21c66ed5760>,
 Text(0.5, 1.0, 'Original Image'))
```



Task 2.ipynb U

Lab 6 > Lab6_CSE457 > Task 2.ipynb > ...

+ Code + Markdown | Run All | Clear All Outputs | Outline ...

```
def distance(point1,point2):
    return np.sqrt((point1[0]-point2[0])**2 + (point1[1]-point2[1])**2)

def gaussianLP(D0,imgShape):
    base = np.zeros(imgShape[:2])
    rows, cols = imgShape[:2]
    center = (rows/2,cols/2)
    for x in range(cols):
        for y in range(rows):
            base[y,x] = np.exp(((-distance((y,x),center)**2)/(2*(D0**2))))
    return base

def gaussianHP(D0,imgShape):
    base = np.zeros(imgShape[:2])
    rows, cols = imgShape[:2]
    center = (rows/2,cols/2)
    for x in range(cols):
        for y in range(rows):
            base[y,x] = 1 - np.exp(((-distance((y,x),center)**2)/(2*(D0**2))))
    return base
```

[5]

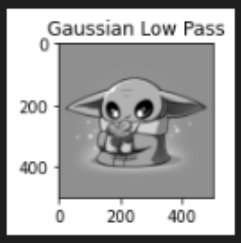
```
lowPassCenter = center*gaussianLP(50,image.shape)
lowPass = np.fft.ifftshift(lowPassCenter)
inverseLowPass = np.fft.ifft2(lowPass)
```

[6]

```
plt.subplot(132), plt.imshow(np.abs(inverseLowPass),'gray'), plt.title("Gaussian Low Pass")
```

[41]

```
(<AxesSubplot:title={'center':'Gaussian Low Pass'}>,
 <matplotlib.image.AxesImage at 0x194fbc5f970>,
 Text(0.5, 1.0, 'Gaussian Low Pass'))
```



Task 2.ipynb U

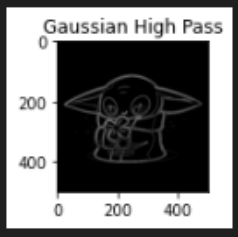
Lab 6 > Lab6_CSE457 > Task 2.ipynb > ...

+ Code + Markdown | Run All Clear All Outputs | Outline ...

[31]
```
highPassCenter = center*gaussianHP(50,image.shape)
highPass = np.fft.ifftshift(highPassCenter)
inverseHighPass = np.fft.ifft2(highPass)
```

[32]
```
plt.subplot(133), plt.imshow(np.abs(inverseHighPass),'gray'), plt.title("Gaussian High Pass")
```

```
(<AxesSubplot:title={'center':'Gaussian High Pass'}>,
 <matplotlib.image.AxesImage at 0x194fb558af0>,
 Text(0.5, 1.0, 'Gaussian High Pass'))
```

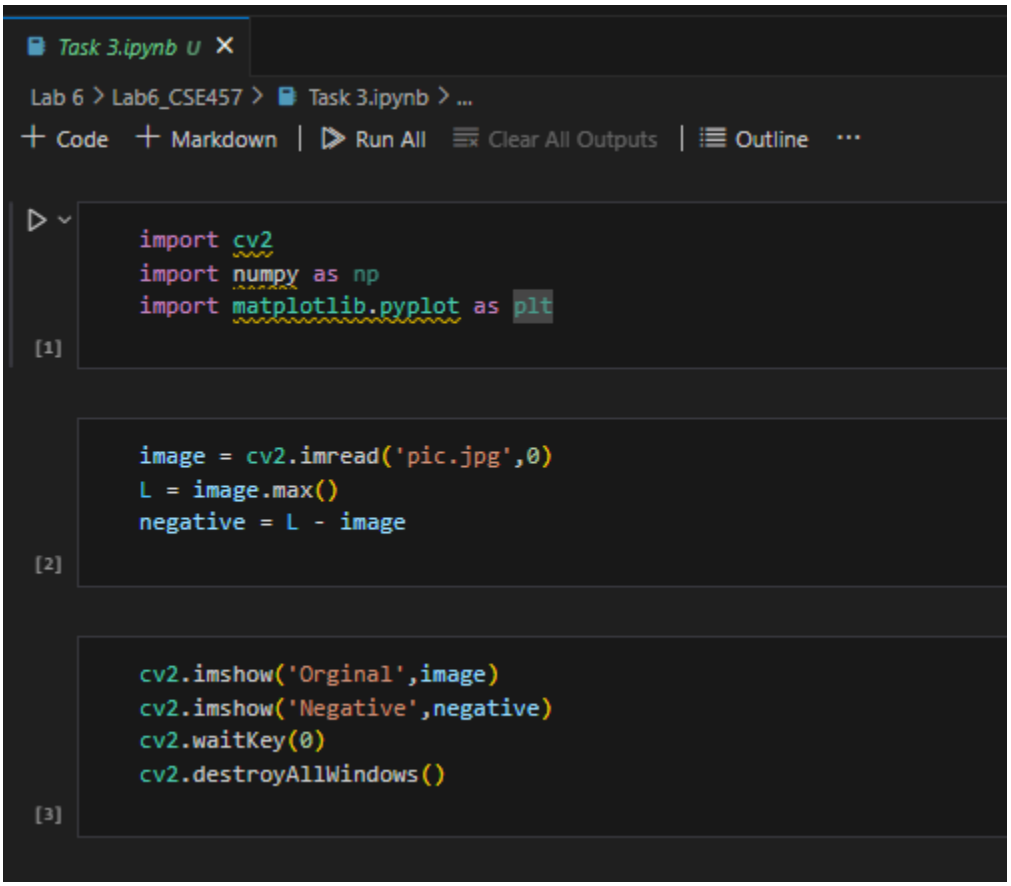


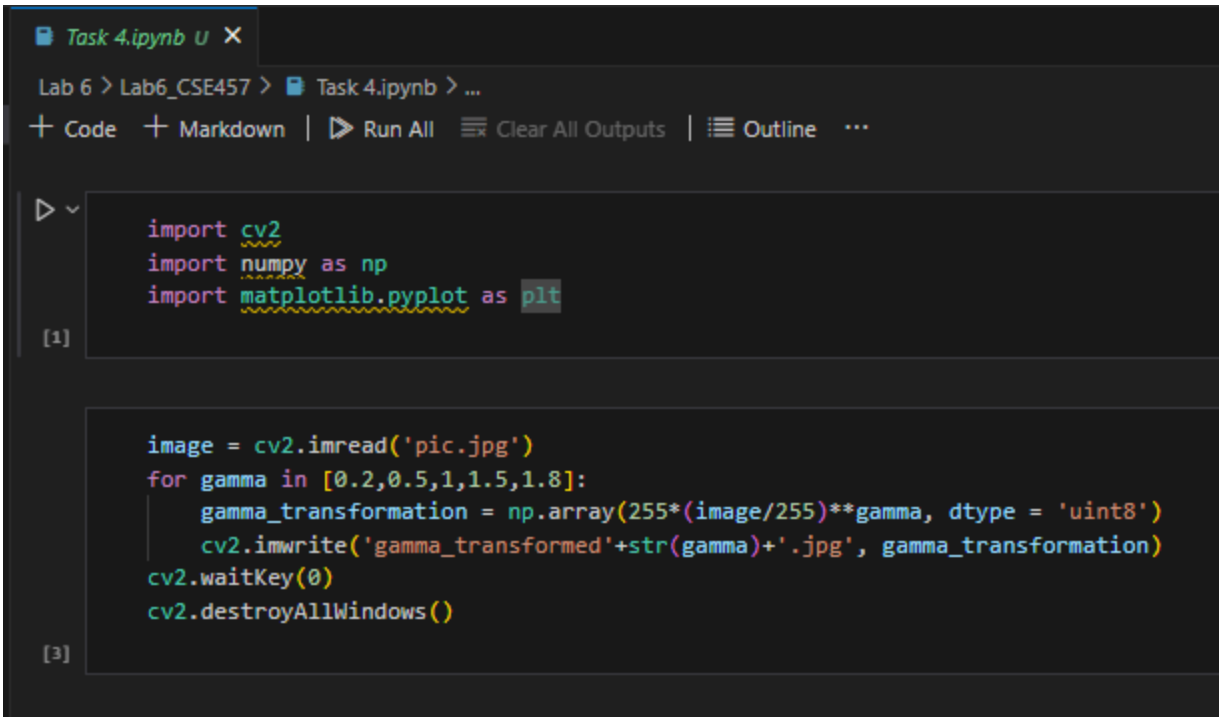
[33]
```
plt.show()
```

[34]
```
print(image.shape)
```

```
(500, 500)
```

*** Point operations**




Task 4.ipynb U
Lab 6 > Lab6_CSE457 > Task 4.ipynb > ...
+ Code
+ Markdown
Run All
Clear All Outputs
Outline
import cv2
import numpy as np
import matplotlib.pyplot as plt
[1]
image = cv2.imread('pic.jpg')
for gamma in [0.2,0.5,1,1.5,1.8]:
gamma_transformation = np.array(255*(image/255)**gamma, dtype = 'uint8')
cv2.imwrite('gamma_transformed'+str(gamma)+'.jpg', gamma_transformation)
cv2.waitKey(0)
cv2.destroyAllWindows()
[3]

* Image Contours

Task 5.ipynb U

Lab 6 > Lab6_CSE457 > Task 5.ipynb > ...

+ Code + Markdown | Run All Clear All Outputs | Outline ···

[1]
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

[2]
```
image = cv2.imread('pic.jpg')
image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

[3]
```
ret, thresh = cv2.threshold(image_gray, 150, 255, cv2.THRESH_BINARY)
```

[5]
```
cv2.imshow('Binary Image', thresh)
cv2.waitKey(0)
cv2.imwrite('image_thresh1.jpg',thresh)
cv2.destroyAllWindows()
```

[7]
```
contours, hierarchy = cv2.findContours(image=thresh, mode=cv2.RETR_TREE, method=cv2.CHAIN_APPROX_NONE)
```

[9]
```
image_copy = image.copy()
cv2.drawContours(image=image_copy, contours=contours, contourIdx=-1, color=(0,255,0), thickness=2,lineType=cv2.LINE_AA)
```

Task 5.ipynb U

Lab 6 > Lab6_CSE457 > Task 5.ipynb > ...

+ Code + Markdown | Run All Clear All Outputs | Outline ···

[9]

```
    image_copy = image.copy()
    cv2.drawContours(image=image_copy, contours=contours, contourIdx=-1, color=(0,255,0), thickness=2,lineType=cv2.LINE_AA)
```

```
array([[[  0, 255,   0],
        [  0, 255,   0],
        [  0, 255,   0],
        ...,
        [  0, 255,   0],
        [  0, 255,   0],
        [  0, 255,   0]],

       [[  0, 255,   0],
        [  0, 255,   0],
        [  0, 255,   0],
        ...,
        [  0, 255,   0],
        [  0, 255,   0],
        [  0, 255,   0]],

       [[  0, 255,   0],
        [  0, 255,   0],
        [ 32, 228,  45],
        ...,
        [ 34, 226,  49],
        [  0, 255,   0],
        [  0, 255,   0]],

       ...,
...
        [  0, 255,   0],
        ...,
        [  0, 255,   0],
        [  0, 255,   0],
        [  0, 255,   0]]], dtype=uint8)
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

[10]

```
    cv2.imshow('None Approximation',image_copy)
    cv2.waitKey(0)
    cv2.imwrite('Contours_None_Image1.jpg', image_copy)
    cv2.destroyAllWindows()
```