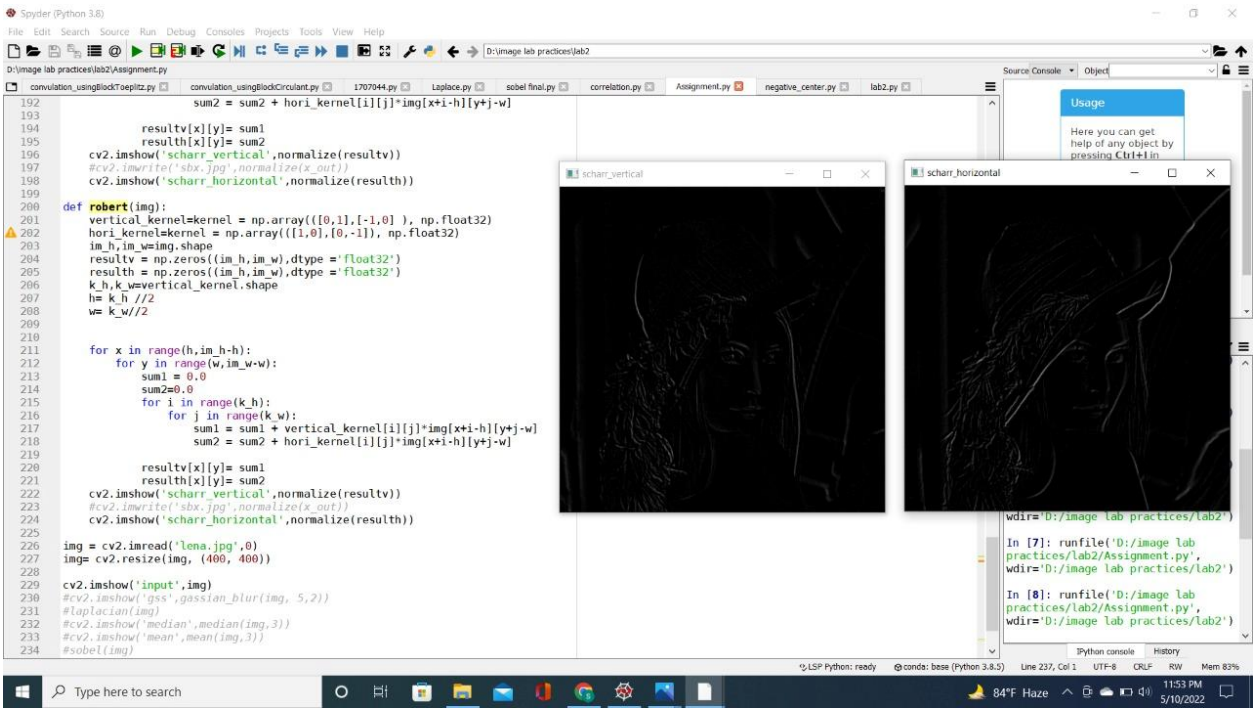
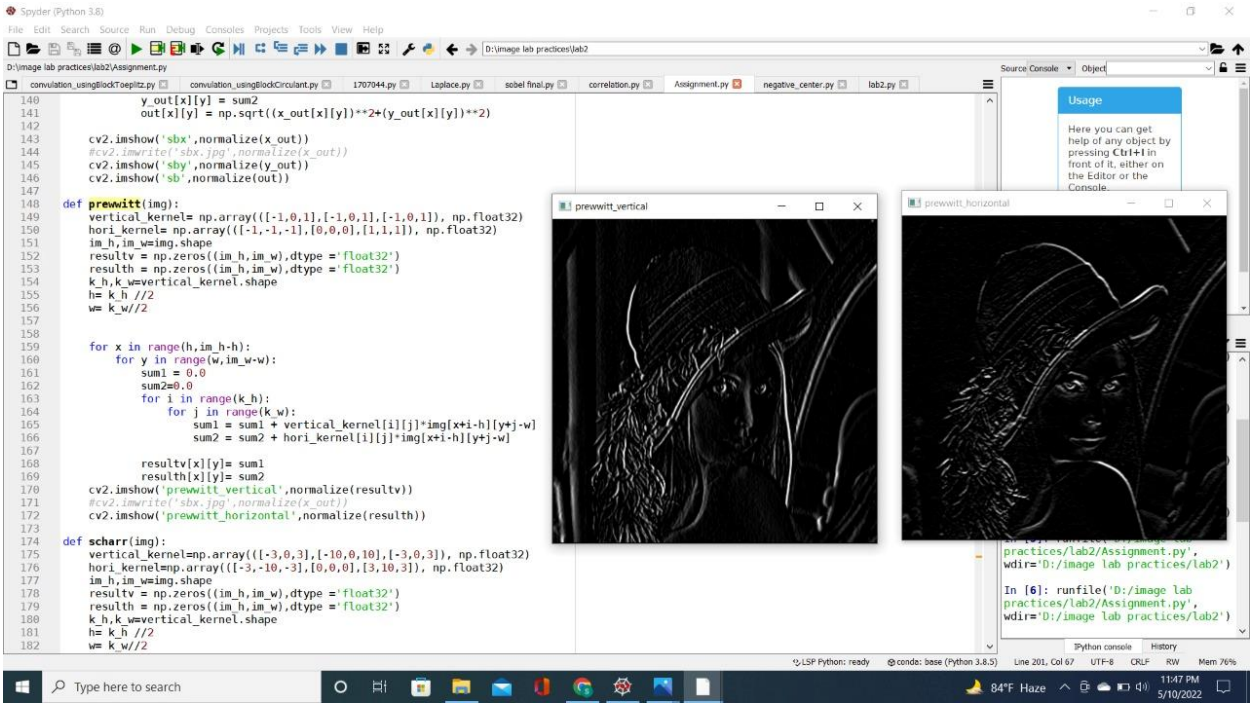
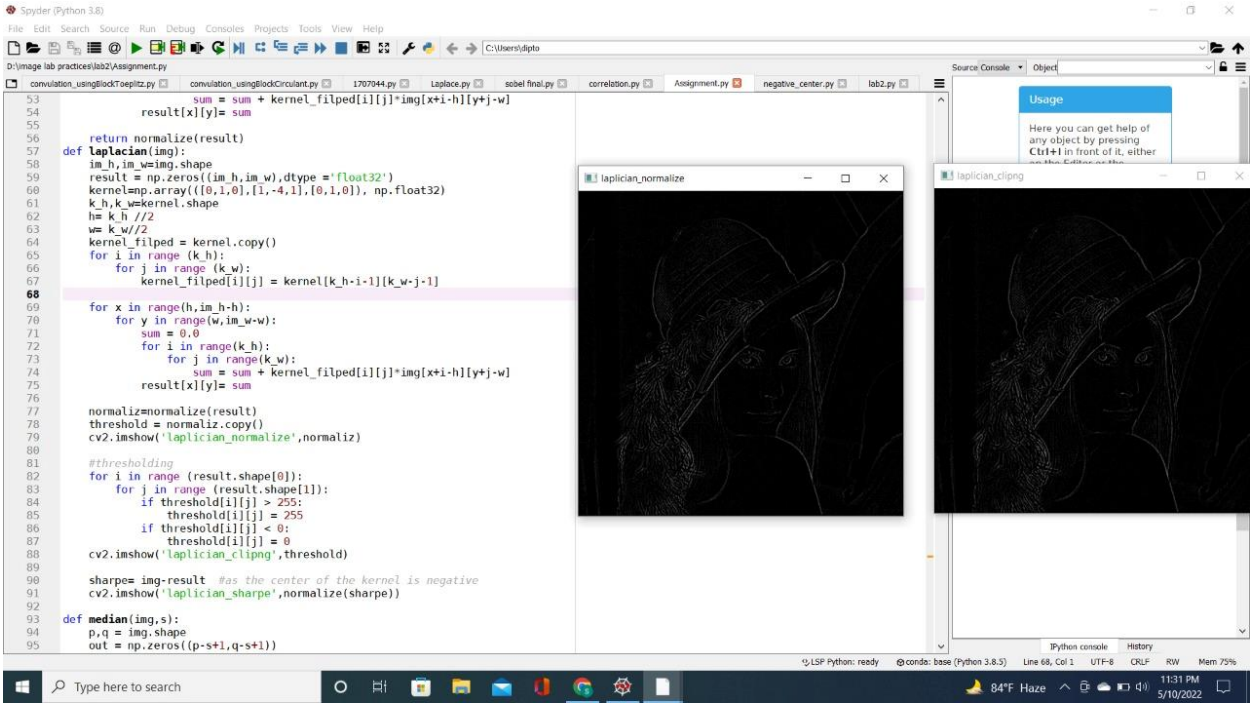
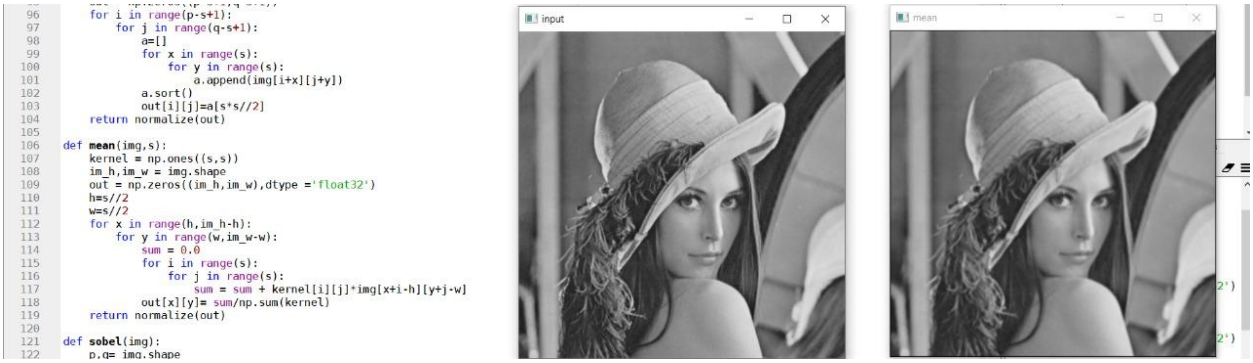

scharr_vertical
scharr_horizontal
Usage
Here you can get help of any object by pressing Ctrl+I in
wdir='D:/image lab practices/lab2')
In [7]: runfile('D:/image lab practices/lab2/Assignment.py', wdir='D:/image lab practices/lab2')
In [8]: runfile('D:/image lab practices/lab2/Assignment.py', wdir='D:/image lab practices/lab2')
IPython console
History
LSP Python: ready
conda: base (Python 3.8.5)
Line 237, Col 1
UTF-8
CRLF
RW
Mem 83%
Type here to search
84°F Haze
11:53 PM
5/10/2022


prewwitt_vertical
prewwitt_horizontal
Usage
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.
practices/lab2/Assignment.py', wdir='D:/image lab practices/lab2')
In [6]: runfile('D:/image lab practices/lab2/Assignment.py', wdir='D:/image lab practices/lab2')
IPython console
History
LSP Python: ready
conda: base (Python 3.8.5)
Line 201, Col 67
UTF-8
CRLF
RW
Mem 76%
Type here to search
84°F Haze
11:47 PM
5/10/2022

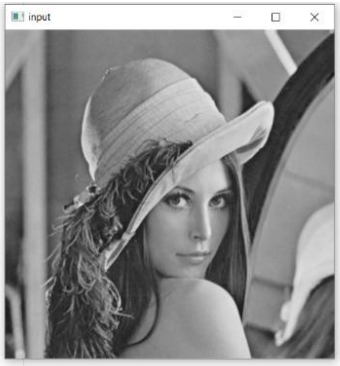


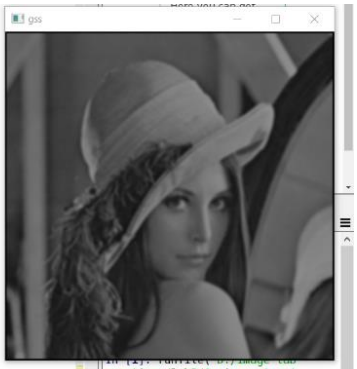

```
                    sum = sum + kernel_filped[i][j]*img[x+i-h][y+j-w]
            result[x][y]= sum

    return normalize(result)
def laplacian(img):
    im_h,im_w=img.shape
    result = np.zeros((im_h,im_w),dtype ='float32')
    kernel=np.array(([0,1,0],[1,-4,1],[0,1,0]), np.float32)
    k_h,k_w=kernel.shape
    h= k_h //2
    w= k_w//2
    kernel_filped = kernel.copy()
    for i in range (k_h):
        for j in range (k_w):
            kernel_filped[i][j] = kernel[k_h-i-1][k_w-j-1]

    for x in range(h,im_h-h):
        for y in range(w,im_w-w):
            sum = 0.0
            for i in range(k_h):
                for j in range(k_w):
                    sum = sum + kernel_filped[i][j]*img[x+i-h][y+j-w]
            result[x][y]= sum

    normaliz=normalize(result)
    threshold = normaliz.copy()
    cv2.imshow('laplician_normalize',normaliz)

    #thresholding
    for i in range (result.shape[0]):
        for j in range (result.shape[1]):
            if threshold[i][j] > 255:
                threshold[i][j] = 255
            if threshold[i][j] < 0:
                threshold[i][j] = 0
    cv2.imshow('laplician_clipng',threshold)

    sharpe= img-result  #as the center of the kernel is negative
    cv2.imshow('laplician_sharpe',normalize(sharpe))

def median(img,s):
    p,q = img.shape
    out = np.zeros((p-s+1,q-s+1))
```



```
    for i in range(p-s+1):
        for j in range(q-s+1):
            a=[]
            for x in range(s):
                for y in range(s):
                    a.append(img[i+x][j+y])
            a.sort()
            out[i][j]=a[s*s//2]
    return normalize(out)

def mean(img,s):
    kernel = np.ones((s,s))
    im_h,im_w = img.shape
    out = np.zeros((im_h,im_w),dtype ='float32')
    h=s//2
    w=s//2
    for x in range(h,im_h-h):
        for y in range(w,im_w-w):
            sum = 0.0
            for i in range(s):
                for j in range(s):
                    sum = sum + kernel[i][j]*img[x+i-h][y+j-w]
            out[x][y]= sum/np.sum(kernel)
    return normalize(out)

def sobel(img):
    p,q= img.shape
```

```
def gauss(size,sigma):
    g = np.zeros((size,size))
    ssq = 2*sigma**2
    tps = math.pi*ssq
    a=size//2
    for x in range(-a,a+1):
        for y in range(-a,a+1):
            g[x+a][y+a]=np.exp(-(x*x+y*y)/ssq)/tps
    return g

def gassian_blur(img,s,sigma):
    im_h,im_w=img.shape
    result = np.zeros((im_h,im_w),dtype ='float32')
    k_h = s
    k_w = s
    kernel=gauss(s, sigma)

    h= k_h //2
    w= k_w//2
    kernel_filped = kernel.copy()
    for i in range (k_h):
        for j in range (k_w):
            kernel_filped[i][j] = kernel[k_h-i-1][k_w-j-1]

    for x in range(h,im_h-h):
        for y in range(w,im_w-w):
            sum = 0.0
            for i in range(k_h):
                for j in range(k_w):
                    sum = sum + kernel_filped[i][j]*img[x+i-h][y+j-w]
            result[x][y]= sum

    return normalize(result)
```






```
practices/lab2/Assignment.py',
wdir='D:/image lab practices/lab2')

In [2]: runfile('D:/image lab
practices/lab2/Assignment.py',
wdir='D:/image lab practices/lab2')
```

Spyder (Python 3.8)

File Edit Search Source Run Debug Consoles Projects Tools View Help

D:\image lab practices\lab2\Assignment.py

convulation_usingBlockToeplitz.py | convulation_usingBlockCirculant.py | 1707044.py | Laplace.py | sobel final.py | correlation.py | Assignment.py | negative_center.py | lab2.py

```
                    sum1 = sum1 + vertical_kernel[i][j]*img[x+i-h][y+j-w]
                    sum2 = sum2 + hori_kernel[i][j]*img[x+i-h][y+j-w]

            resultv[x][y]= sum1
            resulth[x][y]= sum2
    cv2.imshow('prewwitt_vertical',normalize(resultv))
    #cv2.imwrite('sbx.jpg',normalize(x_out))
    cv2.imshow('prewwitt_horizontal',normalize(resulth))

def scharr(img):
    vertical_kernel=np.array(([-3,0,3],[-10,0,10],[-3,0,3]), np.float32)
    hori_kernel=np.array(([-3,-10,-3],[0,0,0],[3,10,3]), np.float32)
    im_h,im_w=img.shape
    resultv = np.zeros((im_h,im_w),dtype ='float32')
    resulth = np.zeros((im_h,im_w),dtype ='float32')
    k_h,k_w=vertical_kernel.shape
    h= k_h //2
    w= k_w//2


    for x in range(h,im_h-h):
        for y in range(w,im_w-w):
            sum1 = 0.0
            sum2=0.0
            for i in range(k_h):
                for j in range(k_w):
                    sum1 = sum1 + vertical_kernel[i][j]*img[x+i-h][y+j-w]
                    sum2 = sum2 + hori_kernel[i][j]*img[x+i-h][y+j-w]

            resultv[x][y]= sum1
            resulth[x][y]= sum2
    cv2.imshow('scharr_vertical',normalize(resultv))
    #cv2.imwrite('sbx.jpg',normalize(x_out))
    cv2.imshow('scharr_horizontal',normalize(resulth))

def robert(img):
    vertical_kernel=kernel = np.array(([0,1],[-1,0] ), np.float32)
    hori_kernel=kernel = np.array(([1,0],[0,-1]), np.float32)
    im_h,im_w=img.shape
    resultv = np.zeros((im_h,im_w),dtype ='float32')
    resulth = np.zeros((im_h,im_w),dtype ='float32')
    k_h,k_w=vertical_kernel.shape
    h= k_h //2
```

Usage

Here you can get help of any object by pressing Ctrl+I in

```
wdir='D:/image lab practices/lab2')

In [6]: runfile('D:/image lab
practices/lab2/Assignment.py',
wdir='D:/image lab practices/lab2')

In [7]: runfile('D:/image lab
practices/lab2/Assignment.py',
wdir='D:/image lab practices/lab2')
```

IPython console | History

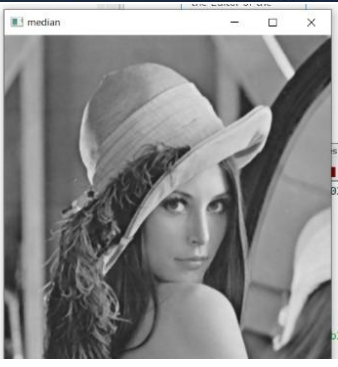
LSP Python: ready | conda: base (Python 3.8.5) | Line 236, Col 1 | UTF-8 | CRLF | RW | Mem 82%

Type here to search

84°F Haze

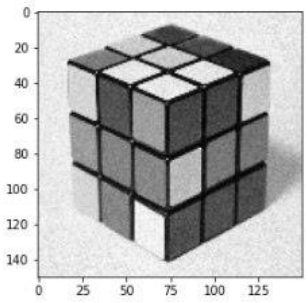
11:49 PM
5/10/2022

```
            if threshold[i][j] < 0:
                threshold[i][j] = 0
    cv2.imshow('laplician_clipng',threshold)

    sharpe= img-result  #as the center of the kernel is negative
    cv2.imshow('laplician_sharpe',normalize(sharpe))

def median(img,s):
    p,q = img.shape
    out = np.zeros((p-s+1,q-s+1))
    for i in range(p-s+1):
        for j in range(q-s+1):
            a=[]
            for x in range(s):
                for y in range(s):
                    a.append(img[i+x][j+y])
            a.sort()
            out[i][j]=a[s*s//2]
    return normalize(out)

def mean(img,s):
    kernel = np.ones((s,s))
    im_h,im_w = img.shape
    out = np.zeros((im_h,im_w),dtype ='float32')
    h=s//2
    w=s//2
    for x in range(h,im_h-h):
```



```
def gauss(x1,x2,sigma):
  dis = np.exp(-((x1-x2)**2)/(2*sigma))
  return dis

n = (kernel_size-1)//2
im_x = img.shape[0]
im_y = img.shape[1]
def bilateral(x,y,sigma):
  n = (kernel_size-1)//2
  center_intensity = img[x][y]
  bilat_kernel = np.zeros((kernel_size,kernel_size), dtype='float32')

  for i in range(-n,n+1):
    for j in range(-n,n+1):
      current_inten = img[x+i-n][y+i-n]
      bilat_kernel[i][j] = gauss(center_intensity, current_inten,sigma)

  w = bilat_kernel.sum()
  bilat_kernel = bilat_kernel/w
  return bilat_kernel

result = np.zeros((kernel_size,kernel_size), dtype='float32')
output = np.zeros((im_x,im_y), dtype='float32')
for i in range(1,img.shape[0]):
  for j in range(1,img.shape[1]):
      result = bilateral(i,j,0.9)
      A = result * gauss_kernel
      for k in range(0,kernel_size):
        for l in range(0,kernel_size):
          output[i][j]+= A[k][l]*img[i-k-1][j-l-1]
```

```
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))

#plt.imshow()
plt.show()
```




```
plt.imshow(cv2.cvtColor(output, cv2.COLOR_BGR2RGB))

#plt.imshow()
plt.show()
```