Project 4

ID:0810981 name:安實克

(a) Source codes:

https://github.com/Nurassyl-lab/NYCU-Image-Processing-2022 you can also find code in my GitHub repo

```
'include packages'
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal
import cv2
import matplotlib

'define class image'
class image:
    def __init__(self, image):
        self.im = image

    def conv_to_ndarray(self):
        return np.array(self.im)

    def get_RGB(self):
        x = self.conv_to_ndarray()
        R, G, B = x[:,:,0], x[:,:,1], x[:,:,2]
        self.rgb_channels = np.array([R,G,B])

    def rgb_to_hsi(self):
        R,G,B = self.rgb_channels[0]/255.0, self.rgb_channels[1]/255.0, self.rgb_channels[2]/255.0
        H,S,I = np.zeros((R.shape[1], R.shape[1])),np.zeros((R.shape[1], R.shape[1])),np.zeros((R.shape[1], R.shape[1]))

        for i in range(R.shape[0]):
            for j in range(R.shape[0]):
                r = R[i,j]
                g = G[i,j]
                b = B[i,j]
                if np.sqrt(((r - g) ** 2) + (r - b) * (g - b)) == 0.0:
                    H[i,j] = 0.0
                else:
                    theta = np.arccos(0.5 * ((r - g) + (r - b)) / np.sqrt(((r - g) ** 2) + (r - b) * (g - b)))

                    if b<=g:
                        H[i,j] = theta
                    elif b > g:
                        H[i,j] = 2 * np.pi - theta

                S[i,j] = 1. - 3. * np.min([r, g, b]) / (r + g + b)
                I[i,j] = (r+g+b)/3

        self.hsi_channels = np.array([H,S,I])

    def hsi_to_rgb(self):
        H,S,I = self.hsi_channels[0], self.hsi_channels[1], self.hsi_channels[2]
        R,G,B = np.zeros((H.shape[1], H.shape[1])),np.zeros((H.shape[1], H.shape[1])),np.zeros((H.shape[1], H.shape[1]))

        for m in range(R.shape[0]):
            for n in range(R.shape[0]):
                h = H[m,n]
                s = S[m,n]
                i = I[m,n]

                h = np.degrees(h)

                if 0 <= h <= 120:
                    R[m,n] = i*(1 + ((s*np.cos(np.radians(h)))/np.cos(np.radians(60) - np.radians(h))))
                    B[m,n] = i*(1-s)
                    G[m,n] = 3*i - (R[m,n]+B[m,n])
                elif 120 <= h <= 240:
                    h = h - 120
                    G[m,n] = i*(1 + ((s*np.cos(np.radians(h)))/np.cos(np.radians(60) - np.radians(h))))
```

```
            R[m,n] = i*(1-s)
            B[m,n] = 3*i - (R[m,n]+G[m,n])
        elif 240 <= h <= 360:
            h = h - 240
            B[m,n] = i*(1 + ((s*np.cos(np.radians(h)))/np.cos(np.radians(60) - np.radians(h))))
            G[m,n] = i*(1-s)
            R[m,n] = 3*i - (B[m,n]+G[m,n])

        R[m,n] = np.clip(round(R[m,n] * 255.), 0, 255)
        G[m,n] = np.clip(round(G[m,n] * 255.), 0, 255)
        B[m,n] = np.clip(round(B[m,n] * 255.), 0, 255)
    return np.array([R,G,B])

'define some functions for convenience'
def show_im(im, save = False, cm = '', name = ''):
    if save:
        matplotlib.use('Agg')
    else:
        matplotlib.use('qt5agg')

    fig = plt.figure(dpi = 200)
    fig.set_size_inches(im.shape[0]/200, im.shape[1]/200, forward=False)
    ax = plt.Axes(fig, [0., 0., 1., 1.])
    ax.set_axis_off()
    fig.add_axes(ax)
    ax.imshow(im, cmap = cm)

    if save:
        plt.savefig(name+'.png', dpi = 200)
        plt.close()
        matplotlib.use('qt5agg')
    return

'load image and convert it to numpy array for convenience'
rose = image(Image.open('LovePeace rose.tif'))

'show image'
# show_im(np.array(rose.im), cm = 'viridis')

'get R, G, B components from image'
rose.get_RGB()

'plot/save and make sure that every channel was selected properly'
show_im(rose.rgb_channels[0],cm='gray', save = True, name = 'red_channel')
show_im(rose.rgb_channels[1],cm='gray', save = True, name = 'green_channel')
show_im(rose.rgb_channels[2],cm='gray', save = True, name = 'blue_channel')

'convert image from R,G,B to H,S,I'
rose.rgb_to_hsi()

'plot/save and make sure that every channel was selected properly'
show_im(rose.hsi_channels[0],cm='gray', save = True, name = 'hue_channel')
show_im(rose.hsi_channels[1],cm='gray', save = True, name = 'sat_channel')
show_im(rose.hsi_channels[2],cm='gray', save = True, name = 'int_channel')

'check if it was right, uncomments lines below'
'for some reason, plt.inshow() does not plot image in a coorect way'
'because of this problem, cv2 imwrite is used'
'NOTE, do not use cv2.imshow, use only imwrite'
# rgb = rose.hsi_to_rgb()
# x = cv2.merge([rgb[2],rgb[1],rgb[0]])
# cv2.imwrite('check.png',x)

'lets sharpen an image'
'first is RGB'
kernel = np.array([[-1,-1,-1],[-1,8,-1],[-1,-1,-1]])

lap_r = signal.convolve2d(rose.rgb_channels[0], kernel, 'full',boundary='symm')[:800,:800]
lap_g = signal.convolve2d(rose.rgb_channels[1], kernel, 'full',boundary='symm')[:800,:800]
lap_b = signal.convolve2d(rose.rgb_channels[2], kernel, 'full', boundary='symm')[:800,:800]
```

```
sh_r = rose.rgb_channels[0] + lap_r
sh_g = rose.rgb_channels[1] + lap_g
sh_b = rose.rgb_channels[2] + lap_b

rgb = cv2.merge([sh_r,sh_g,sh_b])# you need to save it, if you plot it the image will not be correct
show_im(rgb, cm = 'viridis', save = True, name = 'rgb_sharp')

'second is HSI'
lap_h = signal.convolve2d(rose.hsi_channels[0], kernel, 'full',boundary='symm')[:800,:800]
lap_s = signal.convolve2d(rose.hsi_channels[1], kernel, 'full',boundary='symm')[:800,:800]
lap_i = signal.convolve2d(rose.hsi_channels[2], kernel, 'full', boundary='symm')[:800,:800]

sh_i = rose.hsi_channels[2] + lap_i

rose.hsi_channels = np.array([rose.hsi_channels[0],rose.hsi_channels[1],sh_i])
x = rose.hsi_to_rgb()
hsi = cv2.merge([x[2],x[1],x[0]])
cv2.imwrite('hsi_sharp.png',hsi)

'load 2 images again and find difference'
X = cv2.imread('rgb_sharp.png')
X = cv2.cvtColor(X, cv2.COLOR_RGB2BGR)
Y = cv2.imread('hsi_sharp.png')
Y = cv2.cvtColor(Y, cv2.COLOR_RGB2BGR)
DIFF = X - Y
show_im(DIFF, save = True, cm = 'viridis', name = 'DIFF')
```

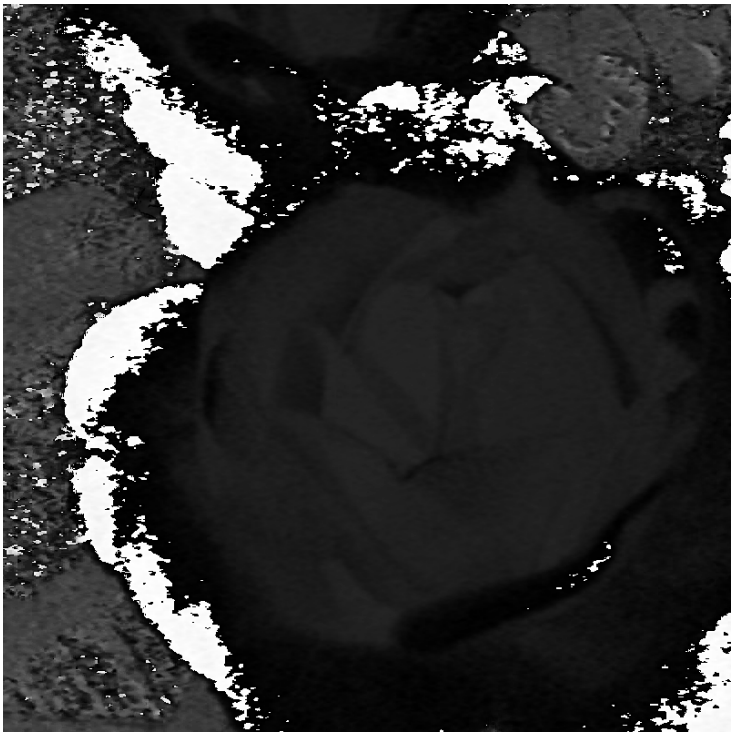
(b) Images of R, G, B, H, S and I component images:

R

G

B

H

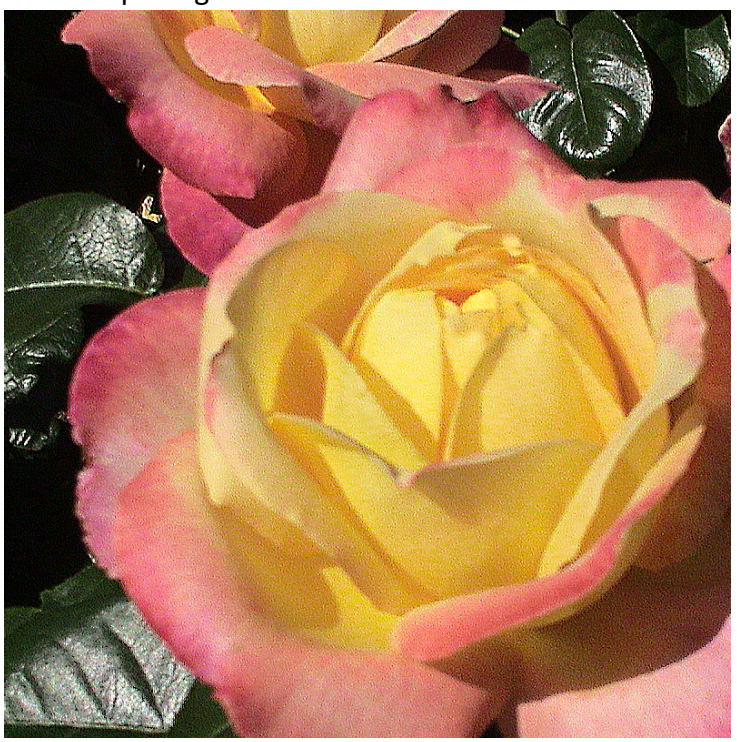


S

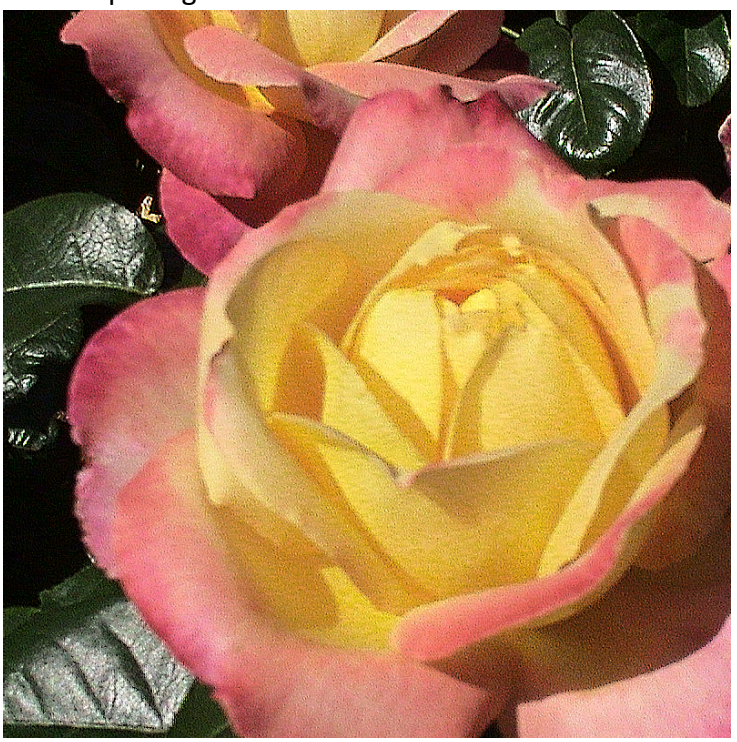
I

(c) Output images enhanced by RGB-sharpening and HSI-sharpening scheme:

RGB-sharpening



HSI-sharpening



(d) Difference image of two images obtained in (c):

IT IS ALMOST DARK BUT YOU CAN SEE DETAILS