Digital Image Processing Project 4

Name:彭晨益

Student ID: 310512054

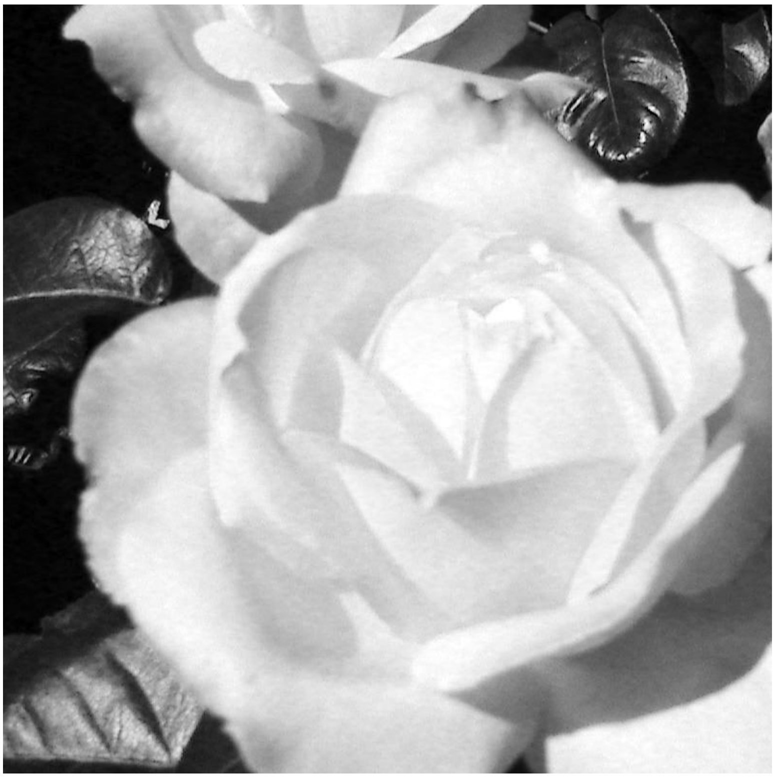
(a)Source Code

```
import os
import sys
import cv2
import copy
import numpy as np
from tqdm import tqdm
from PIL import Image

def check_folder(path):
    if not os.path.exists(path):
        os.makedirs(path)
    return path

def convert_hsi_to_bgr(h, s, i):
    # Convert the HSI values to BGR

    if h >= 0 and h < 120 :
        b = i * (1 - s)
        r = i * (1 + (s * np.cos(np.radians(h)) / np.cos(np.radians(60 - h))))
        g = i * 3 - (r + b)
    elif h >= 120 and h < 240:
        h -= 120
        r = i * (1 - s)
        g = i * (1 + (s * np.cos(np.radians(h)) / np.cos(np.radians(60 - h))))
        b = 3 * i - (r + g)
    elif h >= 240 and h <= 360:
        h -= 240
        g = i * (1 - s)
        b = i * (1 + (s * np.cos(np.radians(h)) / np.cos(np.radians(60 - h))))
        r = i * 3 - (g + b)
    else:
        g = i
        b = i
        r = i
    return b, g, r


def convert_rgb_to_hsi(rgb):
    # Convert the RGB values to HSI
    r, g, b = rgb[0] / 255., rgb[1] / 255., rgb[2] / 255.
    h = 0
    s = 0
    i = (r + g + b) / 3
    if i > 0:
        s = 1 - min(r, g, b) / i
        if s > 0:
            if r == g and g == b and r ==b:
                h = float('nan')
            else:
                h = np.rad2deg(np.arccos((0.5 * ((r - g) + (r - b))) / np.sqrt((r -
g)**2 + (r - b) * (g - b))))
            if b > g:
                h = 360 - h
    return h, s, i
def bgr2rgb(img):
    rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    return rgb

class myImage():

    def __init__(self, img_path, save):
        self.img = cv2.imread(img_path)
        self.rgb = self.bgr2rgb()
        self.hsi = self.rgb2hsi(self.rgb)
        self.save = save

    def bgr2rgb(self):
        image = (self.img)
        rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        self.r, self.g, self.b = rgb[..., 0], rgb[..., 1], rgb[..., 2]
        return rgb

    def hsi2bgr(self, hsi_img):
        hsi_img_ = copy.deepcopy(hsi_img)
        m, n = hsi_img_.shape[:2]
        bgr = np.zeros((m, n, 3), dtype=np.uint8)
        for j in tqdm(range(m)):
            for k in range(n):
                h = hsi_img_[j][k][0]
                s = hsi_img_[j][k][1]
                i = hsi_img_[j][k][2]

                b, g, r = convert_hsi_to_bgr(h, s, i)

                bgr[j][k][0] = np.clip(round(b * 255.), 0, 255)
                bgr[j][k][1] = np.clip(round(g* 255.), 0, 255)
                bgr[j][k][2] = np.clip(round(r * 255.), 0, 255)
        return bgr

    def rgb2hsi(self, image):

        m, n = image.shape[:2]
        hsi = np.zeros((m, n, 3), dtype=np.float32)
        for j in tqdm(range(m)):
            for k in range(n):
                h, s, i = convert_rgb_to_hsi(image[j][k])
                hsi[j][k][0] = h
                hsi[j][k][1] = s
                hsi[j][k][2] = i
        self.h, self.s, self.i = hsi[..., 0], np.uint8(hsi[..., 1] * 255),
np.uint8(hsi[..., 2] * 255)
        return hsi

    def sharpen_img(self, type="rgb"):
        sharpen_kernel = np.array([
            [-1, -1, -1],
            [-1,  9, -1],
```

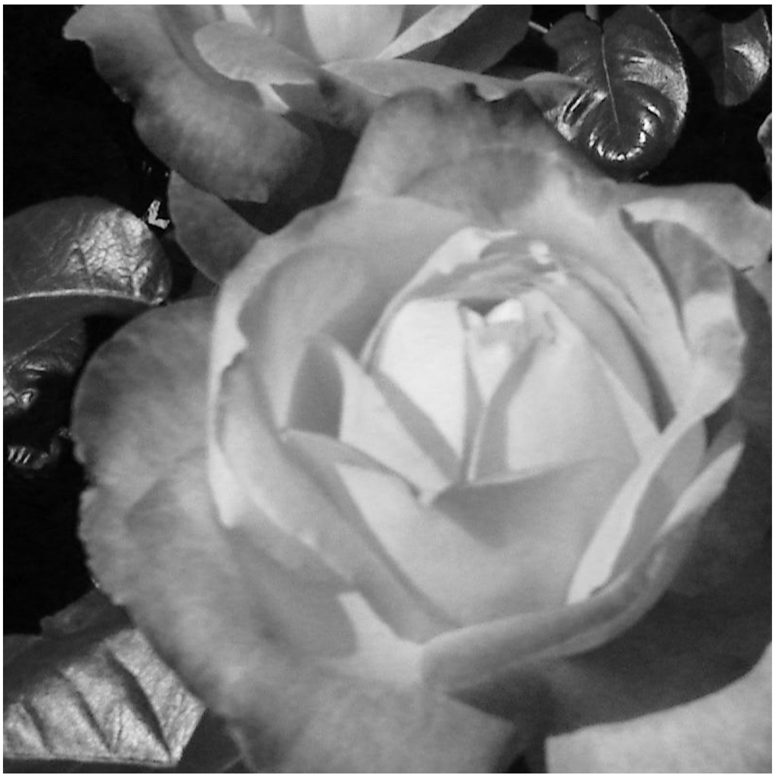
```python
            [-1, -1, -1]], dtype='float32')
        if type == 'rgb':
            img_sharpen = cv2.filter2D(self.img, -1, kernel=sharpen_kernel)
            return img_sharpen
        elif type == 'hsi':
            img_sharpen = self.hsi
            img_sharpen[..., -1] = cv2.filter2D(self.hsi[..., -1], -1, kernel=sharpen_kernel)
            img_sharpen = self.hsi2bgr(img_sharpen)
            return img_sharpen
        else:
            print("[Warn] Wrong sharpen type!!")


    def save_result(self):
        if self.save:
            img_list = [self.r, self.g, self.b, self.h, self.s, self.i]
            title_list = ["r", "g", "b", "h", "s", "i"]
            check_folder("result")
            for i in range(len(img_list)):
                image = Image.fromarray(img_list[i])
                if image.mode == "F":
                    image = image.convert('L')
                image.save("result"+ '/' + title_list[i] + ".jpg", dpi=(200.0, 200.0, 0))

    def save_single_img(self, img, img_title):
        check_folder("result")
        image = Image.fromarray(np.uint8(img))
        image.save("result"+ '/' +img_title + ".jpg", dpi=(200.0, 200.0, 0))

def main():
    # Split the r, g, b and convert to h, s, i in class function
    img = myImage("LovePeace rose.tif", save=True)
    rgb_sharp = img.sharpen_img("rgb")
    hsi_sharp = img.sharpen_img("hsi")

    cv2.imshow("original",img.img)
    cv2.imshow("RGB_sharpen", rgb_sharp)
    cv2.imshow("HSI_sharpen", hsi_sharp)
    cv2.waitKey()
    img.save_result()
    rgb_sharp = bgr2rgb(rgb_sharp)
    hsi_sharp = bgr2rgb(hsi_sharp)
    img.save_single_img(rgb_sharp, "rgb_sharp")
    img.save_single_img(hsi_sharp, "hsi_sharpen")
    img.save_single_img(rgb_sharp - hsi_sharp, "difference")

if __name__ == "__main__":
    main()
```

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

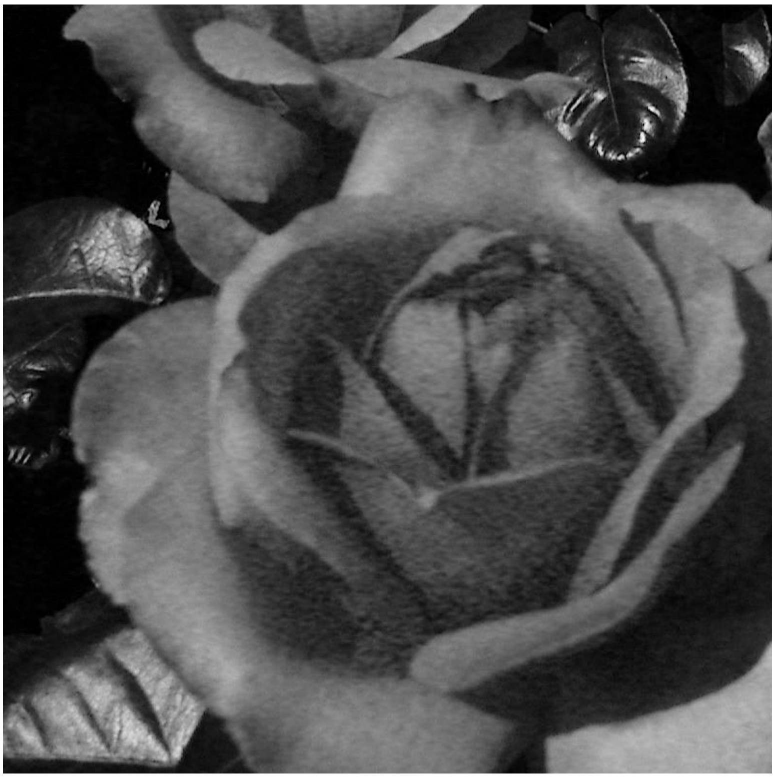
R:



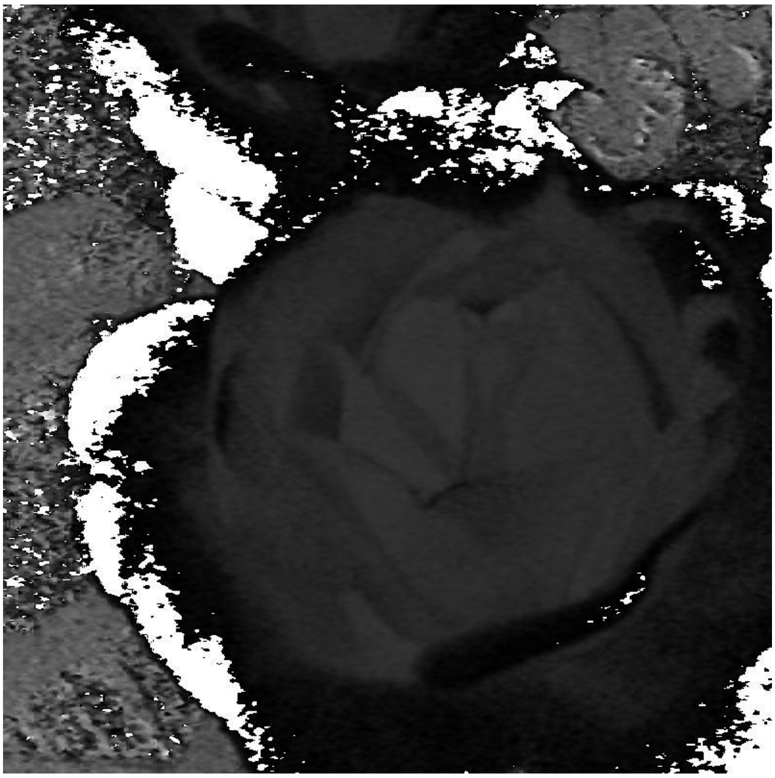
G:



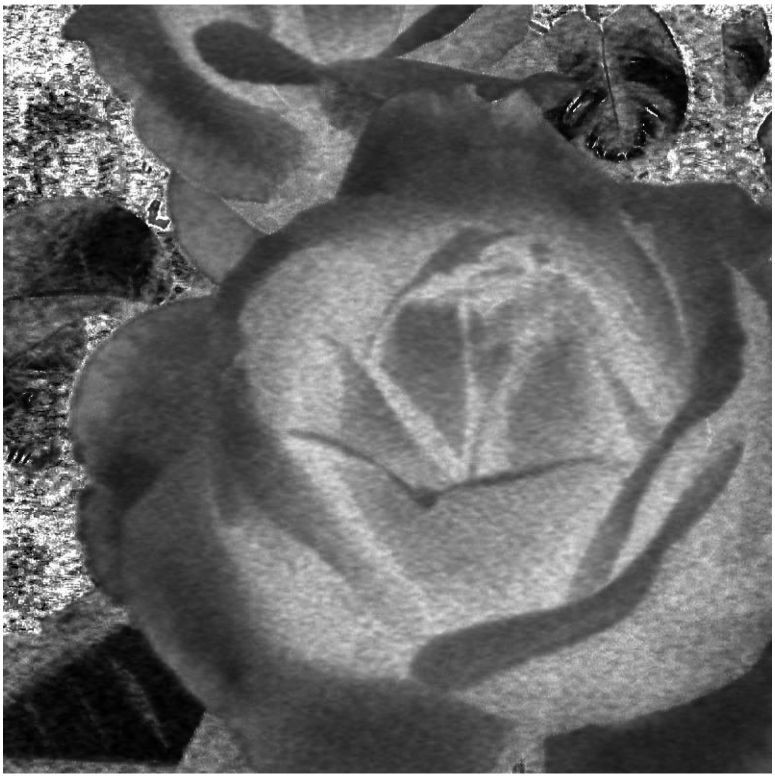
B:



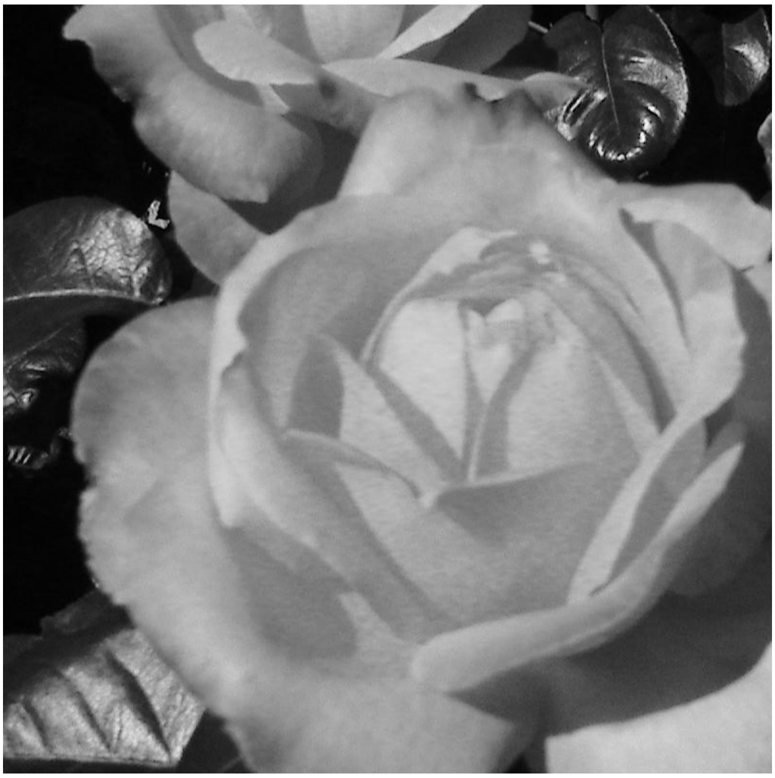
H:



S:

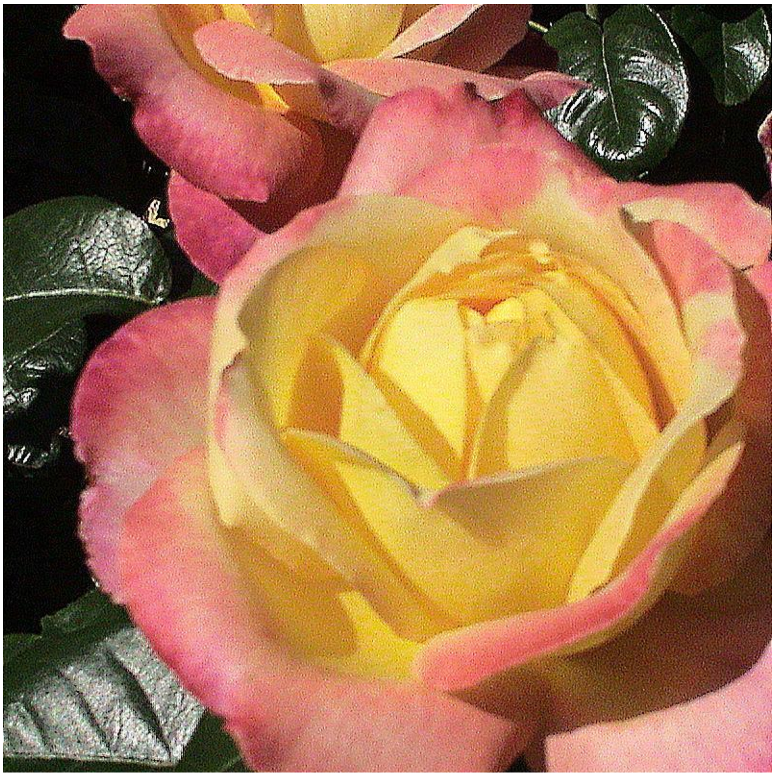


I:

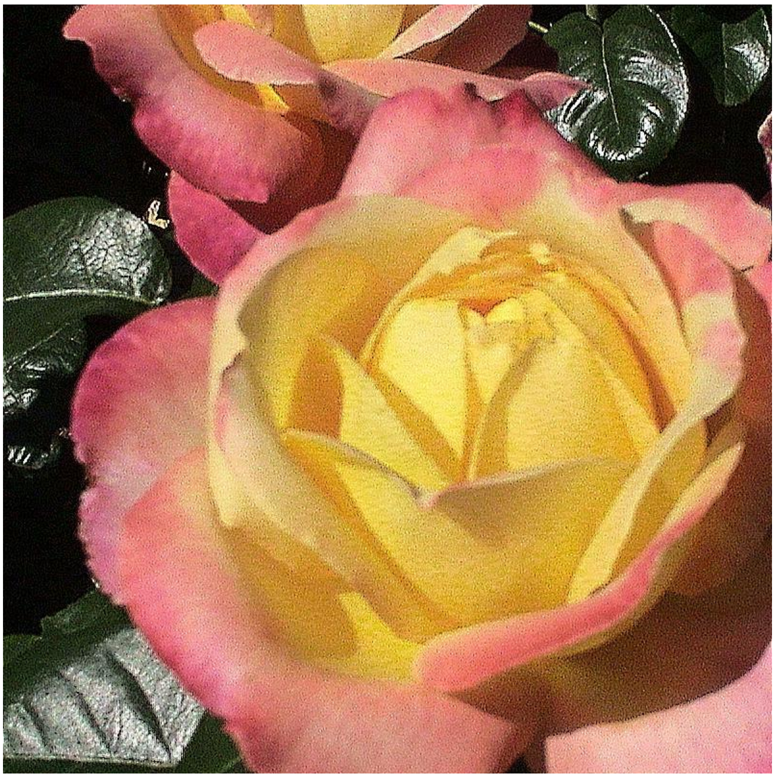


(c) Output images enhanced by RGB-sharpening and HSI-sharpening scheme:

RGB Sharpening:



HSI Sharpening:



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