Project #5

DIGITAL IMAGE PROCESSING

Student: 黎文雄 (Le Van Hung)

Student ID: 0860831

Requirement:

a. Determine and plot the H, S and I component images

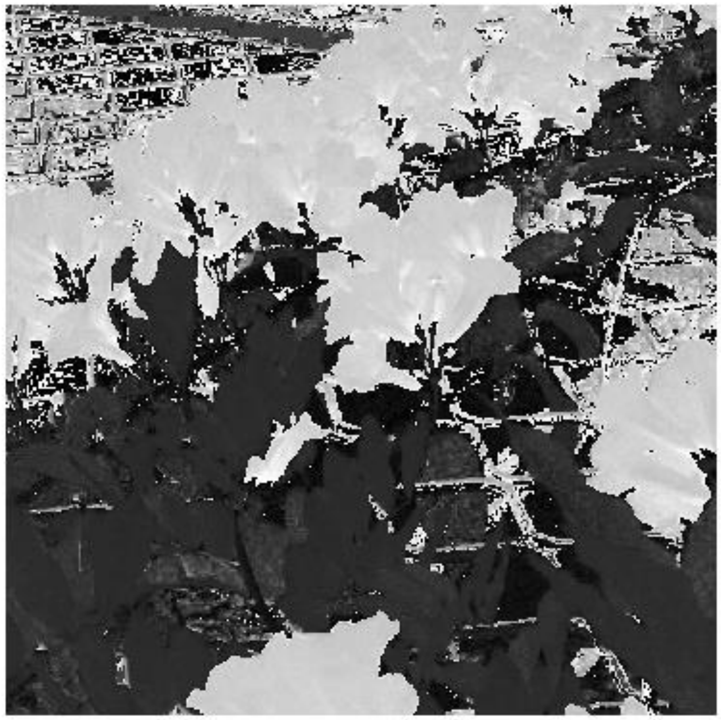
b. Apply sphere-based color slicing to the image, using the prototypical color (i) a1 = (134, 51, 143) and (ii) a2 = (131, 132, 4), and the same radius of the sphere, Ro = 30

- *Your report should contain:*
- Source codes
- Figures of H, S and I component images
- Figure of color – slicing image using a1
- Figure of color – slicing image using a2

Solution

RGB image

1. Figures of H, I and I component images



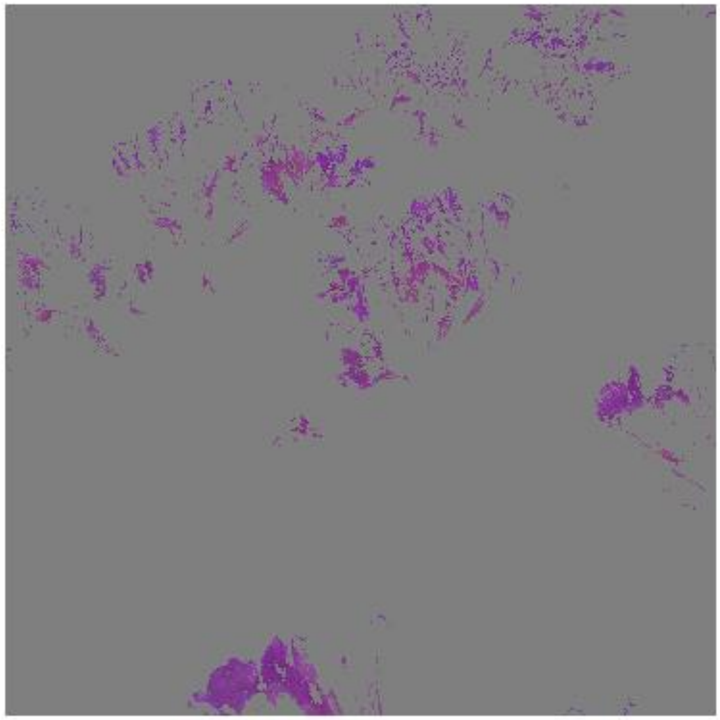
H component

S component

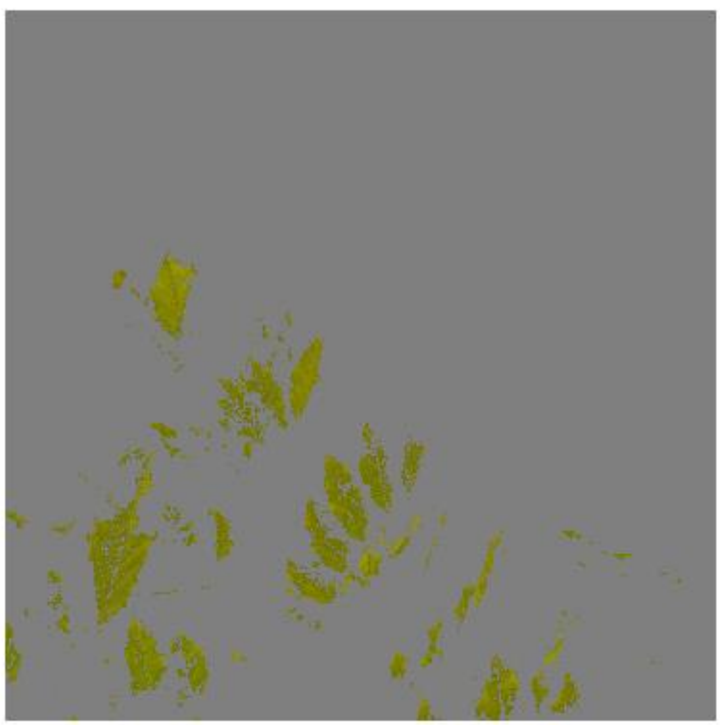
I component

HSI picture

2. Figure of color – slicing image using a1



2. Figure of color – slicing image using a2



3. Source code (python language)

```
4. # import library
from PIL import Image
import cv2
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing

# import image
image = plt.imread("violet (color).tif")  #RGB order
f = np.array(image, dtype='float')
shape_img = np.shape(f)
N = shape_img[0]
f = f/255

fig = plt.figure()
plt.axis("off")
plt.imshow(image)
fig.savefig("RGB image",bbox_inches = 'tight')

# get R,G,B component
R = f[:,:,0]
G = f[:,:,1]
B = f[:,:,2]

# calculate H, S, I component
H = np.zeros_like(R)
for i in range(N):
    for j in range(N):
        temp1 = 0.5*(2*R[i][j]-G[i][j]-B[i][j])
        temp2 = np.sqrt((R[i][j]-G[i][j])**2+(R[i][j]-
B[i][j])*(G[i][j]-B[i][j]))
        if temp2 ==0:
            theta = np.pi/2
        else:
            cos_theta = temp1 / temp2
            theta = np.arccos(cos_theta)
        if B[i][j]<=G[i][j]:
            H[i][j] = theta
        else:
            H[i][j] = 2*np.pi - theta
H = H/(2*np.pi)


fig = plt.figure()
```

```
plt.axis("off")
plt.imshow(H,cmap='gray')
fig.savefig("H component",bbox_inches = 'tight')

# calculate S component
S = np.zeros_like(R)
for i in range(N):
    for j in range(N):
        if (R[i][j] == G[i][j] and R[i][j] == B[i][j]):
            S[i][j] = 0
        else:
            minRGB =
np.minimum(R[i][j],np.minimum(G[i][j],B[i][j]))
            S[i][j] = 1 - (3/(R[i][j]+G[i][j]+B[i][j]))*minRGB
fig = plt.figure()
plt.axis("off")
plt.imshow(S,cmap='gray')
fig.savefig("S component",bbox_inches = 'tight')

#calculate I component
I = np.divide(R+G+B,3)

fig = plt.figure()
plt.axis("off")
plt.imshow(I,cmap='gray')
fig.savefig("I component",bbox_inches = 'tight')

# plot HSI image
gHSI = np.zeros_like(f)
gHSI[:,:,0] = H
gHSI[:,:,1] = S
gHSI[:,:,2] = I

fig = plt.figure()
plt.axis("off")
plt.imshow(gHSI)
fig.savefig("HSI picture", bbox_inches = 'tight')

###############################################################
##############
#Apply sphere-based color slicing to the image
def color_slice(image,a,R0):
    a = np.array(a)
    a = a/255
    R0 = R0/255
    out_image = np.zeros_like(image)
```

```python
    shape_img = np.shape(image)
    N = shape_img[0]
    for i in range(N):
        for j in range(N):
            RR = R[i][j]
            GG = G[i][j]
            BB = B[i][j]
            distance_sqr = (RR-a[0])**2+(GG-a[1])**2+(BB-
a[2])**2
            if distance_sqr > R0**2:
                out_image[i][j][0] = 0.5
                out_image[i][j][1] = 0.5
                out_image[i][j][2] = 0.5
            else:
                out_image[i][j][0] = RR
                out_image[i][j][1] = GG
                out_image[i][j][2] = BB
    return  out_image

# color slicing with a1 = (134,51,143)
a1 = [134,51,143]
R0 = 30
g1 = color_slice(f,a1,R0)

# color slicing with a1 = (131,132,4)
a2 = [131,132,4]
R0 = 30
g2 = color_slice(f,a2,R0)

fig = plt.figure()
plt.axis("off")
plt.imshow(g1)
fig.savefig("using prototypical color a1", bbox_inches =
'tight')

fig = plt.figure()
plt.axis("off")
plt.imshow(g2)
fig.savefig("using prototypical color a2", bbox_inches =
'tight')
```