Handson 2

CIE5141 Computer Vision in Construction

Student_ID Student_Name

1.1 Using OpenCV convert original image to grayscale image

```
grayImage = cv2.cvtColor(originalImage, cv2.COLOR_BGR2GRAY )
plt.imshow(grayImage, cmap='gray')
plt.show()
```

1.2 Converting Gray to Black and White Image

```
# Convert the gray image to black and white image
thresh, blackAndWhiteImage  = cv2.threshold(grayImage, 127, 255, cv2.THRESH_BINARY)
plt.imshow(blackAndWhiteImage, cmap='gray')
plt.show()
```

1.3 Using Pillow Library convert original image to grayscale image

```
from PIL import Image

# Open the image using PIL
org_img = Image.open(path)
display(org_img)

# Convert the image to grayscale
gray_img = org_img.convert('L')
display(gray_img)
```

2.1 Use OpenCV for color channel manipulation

```
# Resize the Image
org_img = cv2.imread(path)
scale= 0.5
height, width, channels =org_img.shape

dim = (int(width/2), int(height/2))
resized_img = cv2.resize(org_img, dim, interpolation=cv2.INTER_AREA)

# Show the resized image
image = cv2.cvtColor(resized_img, cv2.COLOR_BGR2RGB)
plt.imshow(image)
plt.show()
```

```
# Red channel manipulation
b,g,r=cv2.split(resized_img)
r = np.uint8(r*0.1)
resized_img=cv2.merge([b,g,r])
# Show the manipulated images
image = cv2.cvtColor(resized_img, cv2.COLOR_BGR2RGB)
plt.imshow(image)
plt.show()
```

2.2 Creating color mask using OpenCV

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

# Open the safety vest image using OpenCV
path = './safetyvest.jpg'
safetyvest = cv2.imread(path)

# Show the image
plt.imshow(safetyvest)
plt.show()
```

```
# Are the colors correct in this image?
# Open CV represents color channel as BRG instead of RGB

# Convert BGR to RGB
safetyvest = cv2.cvtColor(safetyvest, cv2.COLOR_BGR2RGB)
# Show the image
plt.imshow(safetyvest)
plt.show()
```

```
# Convert the image in HSV space
hsv_sv = cv2.cvtColor(safetyvest, cv2.COLOR_RGB2HSV)

# Show the image
plt.imshow(hsv_sv)
plt.show()
```

```
# Define the range of orange.
light_orange = (1, 190, 200)
dark_orange = (18, 255, 255)

# Create a binary mask within the range.
# If the pixel value is within the range it will return 1 or else 0.

mask = cv2.inRange(hsv_sv , light_orange,dark_orange)

# To impose the mask on top of the original image,
# you can use cv2.bitwise_and(), which keeps every pixel in the given image if the corresponding value in the mask i

result = cv2.bitwise_and(safetyvest, safetyvest, mask = mask)
#use cv2.bitwise_and() to create a result image



# Use sub plot for showing both the images side by side
plt.subplot(121)
plt.imshow(mask, cmap="gray")
plt.subplot(1, 2, 2)
plt.imshow(result)
plt.show()
```

3.1 Bonus

```
# Code for creating 9 manipulated images
new_img = cv2.resize(originalImage, (originalImage.shape[1]//2, originalImage.shape[0]//2))

b,g,r=cv2.split(new_img)

new_r = np.uint8(r*0.1)
resized_img00=cv2.merge([b,g,new_r])
new_r = np.uint8(r*0.5)
resized_img01=cv2.merge([b,g,new_r])
new_r = np.uint8(r*0.9)
resized_img02=cv2.merge([b,g,new_r])

new_g = np.uint8(g*0.1)
resized_img10=cv2.merge([b,new_g,r])
new_g = np.uint8(g*0.5)
resized_img11=cv2.merge([b,new_g,r])
new_g = np.uint8(g*0.9)
resized_img12=cv2.merge([b,new_g,r])

new_b = np.uint8(b*0.1)
resized_img20=cv2.merge([new_b,g,r])
new_b = np.uint8(b*0.5)
resized_img21=cv2.merge([new_b,g,r])
new_b = np.uint8(b*0.9)
resized_img22=cv2.merge([new_b,g,r])
```

```
result0=np.hstack((resized_img00,resized_img01,resized_img02))
result1=np.hstack((resized_img10,resized_img11,resized_img12))
result2=np.hstack((resized_img20,resized_img21,resized_img22))
result=np.vstack((result0,result1,result2))

result = cv2.cvtColor(result, cv2.COLOR_BGR2RGB)
result = cv2.resize(result , (result .shape[1]//2, result .shape[0]//2))
plt.imshow(result)
plt.show()
```