CSC 482
Assignment 3
Hithesh Shanmugam

Template matching and multi-resolution image representation:

Code:

```
import numpy as np
import cv2

# Load the target image and template
img = cv2.imread('C:/Users/sures/Downloads/image.JPG')
template = cv2.imread('C:/Users/sures/Downloads/template.JPG')

# Convert the images to grayscale for processing
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
template_gray = cv2.cvtColor(template, cv2.COLOR_BGR2GRAY)

# Resize the template to match the target image at different scales
scales = [0.5, 1.0, 2.0]
templates = []
for scale in scales:
    resized_template = cv2.resize(template_gray, None, fx=scale, fy=scale, interpolation=cv2.INTER_LINEAR)
    templates.append(resized_template)

# Compute the multi-correlation between the target image and each template
correlations = []
for template in templates:
    correlation = cv2.matchTemplate(img_gray, template, cv2.TM_CCORR_NORMED)
    correlations.append(correlation)

# Pad each correlation to have the same shape as the largest correlation
max_shape = (0, 0)
for correlation in correlations:
    max_shape = np.maximum(max_shape, correlation.shape)

padded_correlations = []
for correlation in correlations:
    padded_correlation = np.zeros(max_shape)
    padded_correlation[:correlation.shape[0], :correlation.shape[1]] = correlation
    padded_correlations.append(padded_correlation)

# Sum the padded correlations from each scale
summed_correlation = np.sum(padded_correlations, axis=0)

# Find the location of the maximum correlation
_, _, _, max_loc = cv2.minMaxLoc(summed_correlation)

# Draw a rectangle on the target image to highlight the matched template
h, w = template_gray.shape
top_left = max_loc
bottom_right = (top_left[0] + w, top_left[1] + h)
cv2.rectangle(img, top_left, bottom_right, (255,255,255), 4)

# Display the result
plt.imshow(img)
```

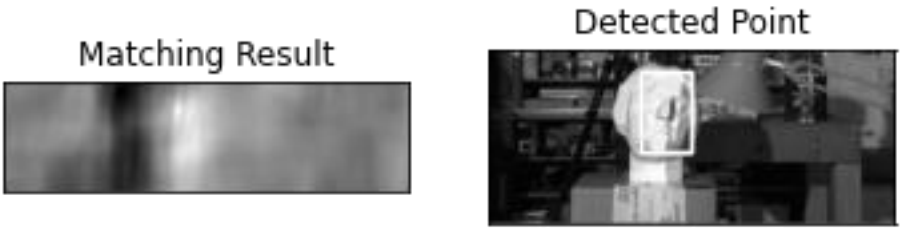
Output:

Code:

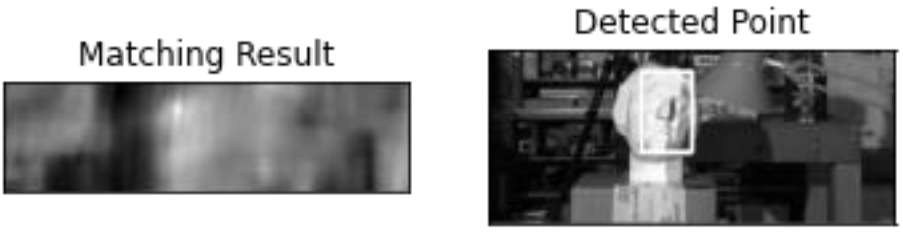
```
methods = ['cv2.TM_CCOEFF', 'cv2.TM_CCOEFF_NORMED', 'cv2.TM_CCORR',
            'cv2.TM_CCORR_NORMED', 'cv2.TM_SQDIFF', 'cv2.TM_SQDIFF_NORMED']

for meth in methods:
    img = img2.copy()
    method = eval(meth)

    # Apply template Matching
    res = cv2.matchTemplate(img,template,method)
    min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)

    # If the method is TM_SQDIFF or TM_SQDIFF_NORMED, take minimum
    if method in [cv2.TM_SQDIFF, cv2.TM_SQDIFF_NORMED]:
        top_left = min_loc
    else:
        top_left = max_loc
    bottom_right = (top_left[0] + w, top_left[1] + h)

    cv2.rectangle(img,top_left, bottom_right,  255, 4)

    plt.subplot(121),plt.imshow(res,cmap = 'gray')
    plt.title('Matching Result'), plt.xticks([]), plt.yticks([])
    plt.subplot(122),plt.imshow(img,cmap = 'gray')
    plt.title('Detected Point'), plt.xticks([]), plt.yticks([])
    plt.suptitle(meth)

    plt.show()

    plt.imshow(img,cmap='gray')

    # PRINT The Coordinates
    print(min_val,max_val,min_loc,max_loc)
```

Output:

cv2.TM_CCOEFF



cv2.TM_CCOEFF_NORMED



cv2.TM_CCORR

cv2.TM_CCORR_NORMED

Matching Result

Detected Point

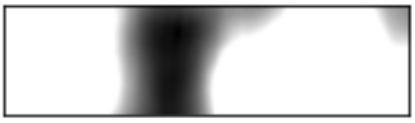
cv2.TM_SQDIFF

Matching Result

Detected Point

cv2.TM_SQDIFF_NORMED

Matching Result



Detected Point

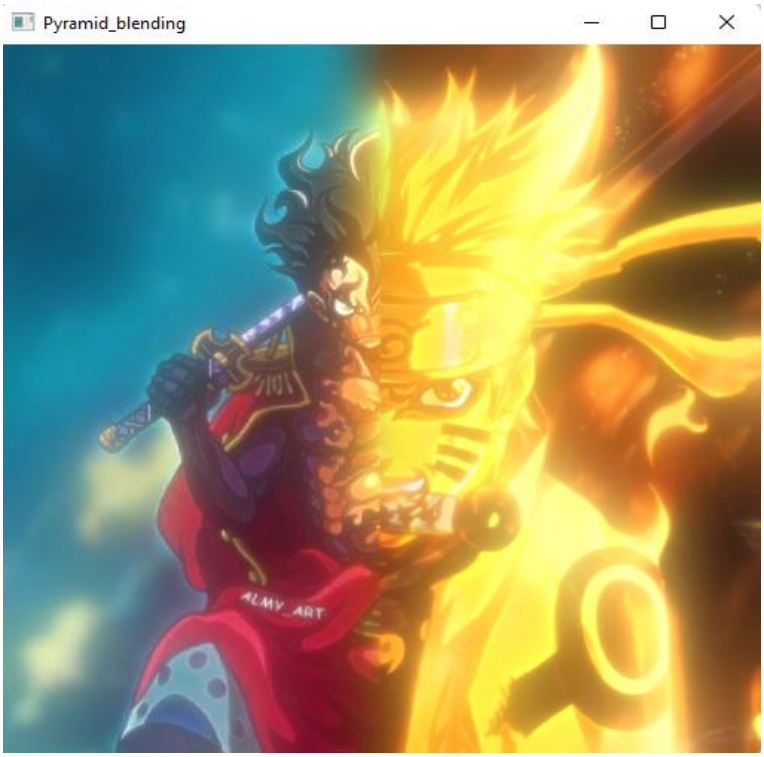
Multi-resolution image blending:

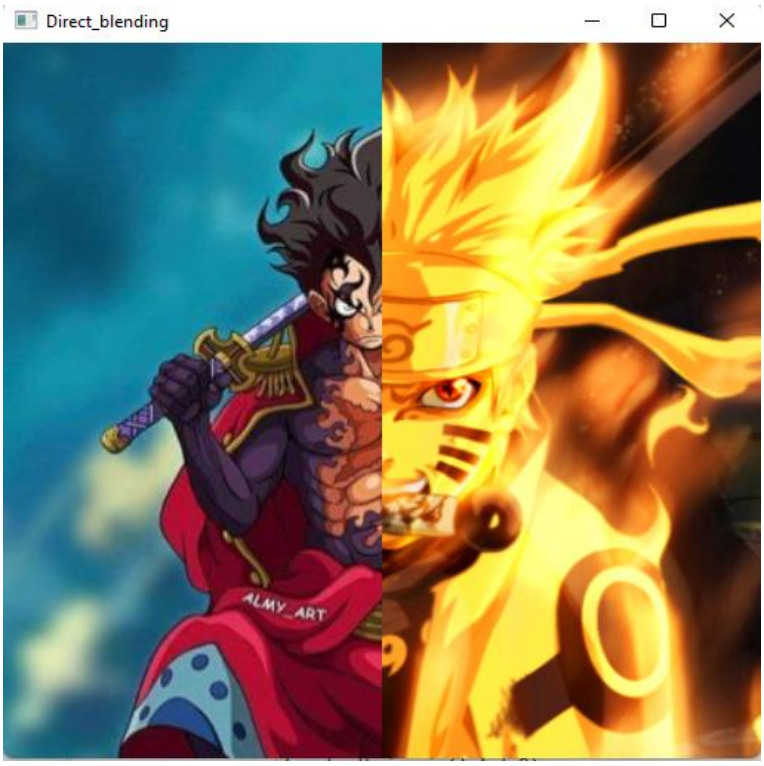
Code:

```
# Read the two input images
img1 = cv2.imread("C:/Users/sures/Downloads/luffy.jpg")
img2 = cv2.imread("C:/Users/sures/Downloads/naruto.jpg")
img1 = cv2.resize(img1, (512, 512))
img2 = cv2.resize(img2, (512, 512))

# generate Gaussian pyramid for img1
gaussian1 = img1.copy()
gaussian_pyramid1 = [gaussian1]
for i in range(6):
    gaussian1 = cv2.pyrDown(gaussian1)
    gaussian_pyramid1.append(gaussian1)

 # generate Gaussian pyramid for img2
gaussian2 = img2.copy()
gaussian_pyramid2 = [gaussian2]
for i in range(6):
    gaussian2 = cv2.pyrDown(gaussian1)
    gaussian_pyramid2.append(gaussian2)

 # generate Laplacian Pyramid for img1
laplacian_pyramid1 = [gaussian_pyramid1[5]]
for i in range(5,0,-1):
    generate = cv2.pyrUp(gaussian_pyramid1[i])
    L = cv2.subtract(gaussian_pyramid1[i-1],generate)
    laplacian_pyramid1.append(L)

# generate Laplacian Pyramid for img2
laplacian_pyramid2 = [gaussian_pyramid2[5]]
for i in range(5,0,-1):
    generate = cv2.pyrUp(gaussian_pyramid2[i])
    L = cv2.subtract(gaussian_pyramid1[i-1],generate)
    laplacian_pyramid2.append(L)

# Now add left and right halves of images in each level
LS = []
for la,lb in zip(laplacian_pyramid1,laplacian_pyramid2):
    rows,cols,dpt = la.shape
    ls = np.hstack((la[:,0:cols//2], lb[:,cols//2:]))
    LS.append(ls)

# now reconstruct
reconstruct = LS[0]
for i in range(1,6):
    reconstruct = cv2.pyrUp(reconstruct)
    reconstruct = cv2.add(reconstruct, LS[i])

# image with direct connecting each half
originial = np.hstack((img1[:,:cols//2],img2[:,cols//2:]))
cv2.imshow('Pyramid_blending',reconstruct)
cv2.waitKey(0)
cv2.imshow('Direct_blending',originial)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input:

Output:




Direct_blending
ALMY_ART