ET3112 Homework 8 on Alignment

D/ENG/21/0108/EE MWM Shakir

Question 1

In [ ]:
```
# Section 1
#http://ais.informatik.uni-freiburg.de/teaching/ws13/mapping/pdf/slam02-homegenous-4.pdf
import matplotlib.pyplot as plt
import numpy as np

# Points a, b, c, d
a, b, c, d = [0, 0, 1], [0, 1, 1], [1, 1, 1], [1, 0, 1]

X = np.array([a, b, c, d]).T #Transposes
# Y = X + np.array([3, 4, 1]).reshape(3,1) # transfromed version of X


t = np.array([1, 0]).T #starting position of transformed line

for i in range(5):
    i=1+i
    A  = np.array([[0, i, t[0]], [0, i, t[1]], [0, 0, 1]]) # Affine transformaiton
    Y = A @ X
    # Homographic transformation

    x = np.append(X[0, :], X[0, 0])
    y = np.append(X[1, :], X[1, 0])
    fig, ax = plt.subplots(1,1)
    ax.plot(x, y, color='g')

    x = np.append(Y[0, :], Y[0, 0])
    y = np.append(Y[1, :], Y[1, 0])
    ax.plot(x, y, color='r')
    ax.set_aspect('equal')
    plt.show()

# more examples on https://docs.opencv.org/4.x/dd/d52/tutorial_js_geometric_transformations.html
```

Question 2

In [ ]:
```
#  Section 2
# Transfroming the first image onto the second
# Gaffiti images: https://www.robots.ox.ac.uk/~vgg/data/affine/
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt

im1 = cv.imread('graf/img1.ppm', cv.IMREAD_GRAYSCALE)
im2 = cv.imread('graf/img5.ppm', cv.IMREAD_GRAYSCALE)
assert im1 is not None
assert im2 is not None

# H = np.array([[8.7976964e-01,   3.1245438e-01,  -3.9430589e+01],
#   [-1.8389418e-01,   9.3847198e-01,   1.5315784e+02],
#    [1.9641425e-04,  -1.6015275e-05,   1.0000000e+00]])

with open('graf/H1to2p') as f:
   H = np.array([[float(h) for h in line.split()] for line in f])

im1to2 = cv.warpPerspective(im1, H, (1000,1000))

fig, axs = plt.subplots(1, 3, figsize=(10, 5))
axs[0].imshow(im1, cmap='gray')
axs[0].set_title("Image 1")
axs[1].imshow(im2, cmap='gray')
axs[1].set_title("image 2")
axs[2].imshow(im1to2, cmap='gray')
axs[2].set_title("Warped")
plt.show()
```

Question 3

In [ ]:

```
import numpy as np
import cv2 as cv

# Load the images
im1 = cv.imread('graf/img1.ppm', cv.IMREAD_ANYCOLOR)
im2 = cv.imread('graf/img5.ppm', cv.IMREAD_ANYCOLOR)
assert im1 is not None
assert im2 is not None

# Define the number of matching points
N = 5

# Initialize arrays to store the matching points
global n
n = 0
p1 = np.empty((N,2))
p2 = np.empty((N,2))

# Mouse callback function to select matching points
def draw_circle(event, x, y, falgs, param):
    global n
    p = param[0]
    if event == cv.EVENT_LBUTTONDOWN:
        cv.circle(param[1], (x,y), 5, (255, 0, 0), -1)
        p[n] = (x,y)
        n +=1

# Display the first image and select matching points
im1copy = im1.copy()
cv.namedWindow('Image 1', cv.WINDOW_AUTOSIZE)
param = [p1, im1copy]
cv.setMouseCallback('Image 1', draw_circle, param)
while(1):
    cv.imshow('Image 1', im1copy)
    if n == N:
        break
    if cv.waitKey(20) & 0xFF == 27:
        break

# Display the second image and select matching points
n = 0
im2copy = im2.copy()
cv.namedWindow('Image 2', cv.WINDOW_AUTOSIZE)
param = [p2, im2copy]
cv.setMouseCallback('Image 2', draw_circle, param)
while(1):
    cv.imshow('Image 2', im2copy)
    if n == N:
        break
    if cv.waitKey(20) & 0xFF == 27:
        break

# Compute the homography matrix
H, _ = cv.findHomography(p1, p2)

# Transform the second image onto the first using the homography matrix

im2_transformed = cv.warpPerspective(im2, H, (im1.shape[1], im1.shape[0]))
stitched_image = np.zeros((im1.shape[0], im1.shape[1] + im2.shape[1], 3), dtype=np.uint8)
stitched_image[0:im1.shape[0], 0:im1.shape[1]] = im1
stitched_image[0:im2_transformed.shape[0], im1.shape[1]:] = im2_transformed

# Display the results
cv.imshow('Stitched Image', stitched_image)
cv.waitKey(0)
cv.destroyAllWindows()
```

Question 4

In [ ]:

```python
import numpy as np
import cv2 as cv

# Load images
img1 = cv.imread('graf/img1.ppm')
img2 = cv.imread('graf/img2.ppm')

# Convert images to grayscale
gray1 = cv.cvtColor(img1, cv.COLOR_BGR2GRAY)
gray2 = cv.cvtColor(img2, cv.COLOR_BGR2GRAY)

# Initialize SIFT detector
sift = cv.SIFT_create()

# Find keypoints and descriptors using SIFT
kp1, des1 = sift.detectAndCompute(gray1, None)
kp2, des2 = sift.detectAndCompute(gray2, None)

# Initialize brute force matcher
bf = cv.BFMatcher(cv.NORM_L2)

# Match descriptors using K-Nearest Neighbor
matches = bf.knnMatch(des1, des2, k=2)

# Apply ratio test
good_matches = []
for m, n in matches:
    if m.distance < 0.75 * n.distance:
        good_matches.append(m)

# Minimum number of matches
MIN_MATCH_COUNT = 10

# Check if enough good matches are found
if len(good_matches) >= MIN_MATCH_COUNT:
    # Extract matching keypoints
    src_pts = np.float32([kp1[m.queryIdx].pt for m in good_matches]).reshape(-1, 1, 2)
    dst_pts = np.float32([kp2[m.trainIdx].pt for m in good_matches]).reshape(-1, 1, 2)

    # Find homography using RANSAC algorithm
    H, mask = cv.findHomography(src_pts, dst_pts, cv.RANSAC, 5.0)

    # Apply homography to warp images
    warped_img2 = cv.warpPerspective(img2, H, (img1.shape[1] + img2.shape[1], img1.shape[0]))
    warped_img2[:img1.shape[0], :img1.shape[1]] = img1

    # Show stitched image
    cv.imshow('Stitched Image', warped_img2)
    cv.waitKey(0)
    cv.destroyAllWindows()

else:
    print("Not enough matches are found - %d/%d" % (len(good_matches), MIN_MATCH_COUNT))
```