# EN2550 2018: Assignment 02

February 20, 2021

Please note that you must implement the key Python functions and scripts on your own. If you copy from the Internet or others, you will not get the learning experience intended through this assignment. Based on the Jupyter notebook file and the Graffiti images carry out the following:

1. Using he given code (item no. 1), experiment with various types of 2-D transformations. [1 mark]
2. Transform Graffiti (`https://www.robots.ox.ac.uk/~vgg/data/affine/`) img1.ppm onto img5.ppm using code in item no. 2. [2 marks]
3. Item no. 3 is for mouse clicking and selecting matching points in the two images to be stitched. Compute the homography using the relevant OpenCV function and carry out stitching. [3 marks]
4. Item no. 4 is similar to item no. 3. Here, compute the homography using your own code and stitch the two images. [4 marks]
5. BONUS: Stitch images using SuperGlue `https://github.com/magicleap/SuperGluePretrainedNetwork` features instead of mouse-clicked points. Compute the homography though RANSAC or MSAC. Stitch more than two images using mouse-clicked points. Handle the seams. See the original paper here `http://matthewalunbrown.com/papers/iccv2003.pdf`. [4 marks]

Upload a five-page report named as your_index_a02.pdf. Type out the index number within the file as well. Include important parts of code, results, and interpretations in the file.

# en2550_lec12_alignment_assignment

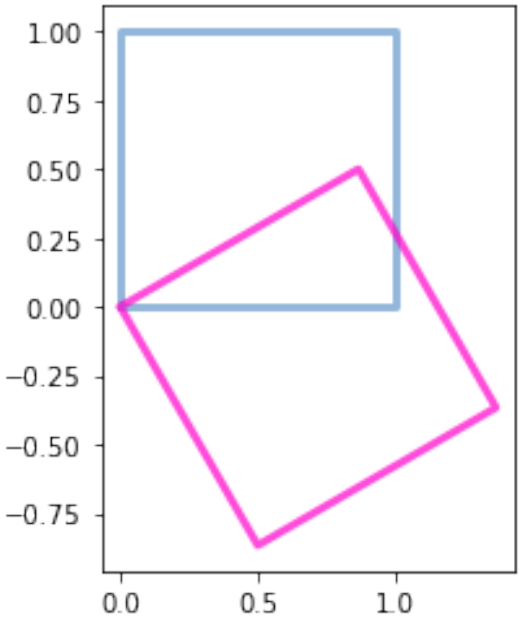
February 20, 2021

## 1 1. 2-D Transforms

[1]:
```
import matplotlib.pyplot as plt
import numpy as np
import string

# points a, b and, c
a, b, c, d = (0, 0, 1), (0, 1, 1), (1, 1, 1), (1, 0, 1)

# matrix with row vectors of points
P = np.array([a, b, c, d]).T
print(P)

# H = np.identity(3)
# H[0,0] = 2
# H[1,1] = 2
# H[2, 0] = 0.3
# H[2, 1] = 0.5

t = np.pi/3

H = [[np.cos(t), np.sin(t), 0.], [-np.sin(t), np.cos(t), 0.], [0., 0., 1.]]
print(H)

Pt = np.matmul(H, P)


P = P/P[-1, :]
P = np.insert(P,4,P[:,0],axis=1)
x = P[0, :]
y = P[1, :]


Pt = Pt/Pt[-1, :]
Pt = np.insert(Pt,4,Pt[:,0],axis=1)
xt = Pt[0, :]
yt = Pt[1, :]
```

```
print(Pt)

fig, ax = plt.subplots(1,1, sharex=True, sharey=True)
ax.plot(x, y, color='#6699cc', alpha=0.7,
    linewidth=3, solid_capstyle='round', zorder=2)
ax.set_aspect('equal')


ax.plot(xt, yt, color='#ff00cc', alpha=0.7,
    linewidth=3, solid_capstyle='round', zorder=2)
ax.set_aspect('equal')
```

```
[[0 0 1 1]
 [0 1 1 0]
 [1 1 1 1]]
[[0.5000000000000001, 0.8660254037844386, 0.0], [-0.8660254037844386,
0.5000000000000001, 0.0], [0.0, 0.0, 1.0]]
[[ 0.          0.8660254   1.3660254   0.5         0.        ]
 [ 0.          0.5        -0.3660254  -0.8660254   0.        ]
 [ 1.          1.          1.          1.          1.        ]]
```



## 2 2. Warping Using a Given Homography

[2]:
```
import cv2 as cv
import numpy as np

im1 = cv.imread('../images/graf/img1.ppm', cv.IMREAD_ANYCOLOR)
im4 = cv.imread('../images/graf/img4.ppm', cv.IMREAD_ANYCOLOR)


with open('../images/graf/H1to4p') as f:
    H = [[float(x) for x in line.split()] for line in f]
print(H)
H = np.array(H)



im4_warped = cv.warpPerspective(im4, np.linalg.inv(H), (1000,1000))
im4_warped[0:im1.shape[0], 0:im1.shape[1]] = im1

cv.namedWindow("Image 1", cv.WINDOW_AUTOSIZE)
cv.imshow("Image 1", im1)
cv.waitKey(0)
cv.namedWindow("Image 4", cv.WINDOW_AUTOSIZE)
cv.imshow("Image 4", im4)
cv.waitKey(0)
cv.namedWindow("Image 4 Warped", cv.WINDOW_AUTOSIZE)
cv.imshow("Image 4 Warped", im4_warped)
cv.waitKey(0)
cv.destroyAllWindows()
```

```
[[0.66378505, 0.68003334, -31.230335], [-0.144955, 0.97128304, 148.7742],
[0.00042518504, -1.3930359e-05, 1.0]]
```

## 3 3. Computing the Homogrpahy Using Mouse-Clicked Points and Warping

[3]:
```
import cv2 as cv
import numpy as np


N = 5
global n
n = 0
p1 = np.empty((N,2))
p2 = np.empty((N,2))
```

```
# mouse callback function
def draw_circle(event,x,y,flags,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


im1 = cv.imread('../images/graf/img1.ppm', cv.IMREAD_ANYCOLOR)
im4 = cv.imread('../images/graf/img4.ppm', cv.IMREAD_ANYCOLOR)

im1copy = im1.copy()
im4copy = im4.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


param = [p2, im4copy]
n = 0
cv.namedWindow("Image 4", cv.WINDOW_AUTOSIZE)
cv.setMouseCallback('Image 4',draw_circle, param)

while(1):
    cv.imshow("Image 4", im4copy)
    if n == N:
        break
    if cv.waitKey(20) & 0xFF == 27:
        break


print(p1)
print(p2)
```

```
cv.destroyAllWindows()
```

```
[[273. 139.]
 [525. 222.]
 [643. 298.]
 [591. 487.]
 [378. 468.]]
[[219. 217.]
 [385. 236.]
 [468. 271.]
 [560. 433.]
 [469. 474.]]
```

# 4 4. Computing the Homogrpahy Using Mouse-Clicked Points without OpenCV

[1]:
```
import cv2 as cv
import numpy as np


N = 5
global n
n = 0
p1 = np.empty((N,2))
p2 = np.empty((N,2))

# mouse callback function
def draw_circle(event,x,y,flags,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


im1 = cv.imread('../images/graf/img1.ppm', cv.IMREAD_ANYCOLOR)
im4 = cv.imread('../images/graf/img4.ppm', cv.IMREAD_ANYCOLOR)

im1copy = im1.copy()
im4copy = im4.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


param = [p2, im4copy]
n = 0
cv.namedWindow("Image 4", cv.WINDOW_AUTOSIZE)
cv.setMouseCallback('Image 4',draw_circle, param)

while(1):
    cv.imshow("Image 4", im4copy)
    if n == N:
        break
    if cv.waitKey(20) & 0xFF == 27:
        break


print(p1)
print(p2)

A = np.empty((2*N, 9))



cv.destroyAllWindows()
```

```
[[272. 196.]
 [101. 322.]
 [182. 142.]
 [527. 170.]
 [434. 305.]]
[[475. 150.]
 [480. 350.]
 [272. 255.]
 [282. 113.]
 [414. 156.]]
```

[ ]: