Projective Geometry HW3
Fall 2019

Name: 楊仲萱 **Department:** 電子所 ICS 組 博一 **Student ID:** F07943023

Part 1: Estimating Homography (4%)

I use solution 2 with solving the SVD of A for estimating homography, and the main function solve_homography(u, v) in main.py is down below.

```
def solve_homography(u, v):
    N = u.shape[0]
    if v.shape[0] is not N:
        print('u and v should have the same size')
        return None
    if N < 4:
        print('At least 4 points should be given')

    #A = np.zeros((2*N, 8))
    # if you take solution 2:
    A = np.zeros((2*N, 9))
    b = np.zeros((2*N, 1))
    H = np.zeros((3, 3))
    # TODO: compute H from A and b
    # consturct A matrix and there are N points
    for i in range(N):
        A[2*i:(2*i+2),:] = np.array([[u[i,0],u[i,1],1,0,0,0,-u[i,0]*v[i,0],-u[i,1]*v[i,0],-v[i,0]],[0,0,0,u[i,0],u[i,1]

    # Using SVD
    U, S, Vt = np.linalg.svd(A)

    # Construct H matrix
    v = np.transpose(Vt)
    h = v[:,-1]
    H = h.reshape((3,3))
    return H
```

After using the five different images to paste on the photo of akihabara, the final result is in the following image.

Part 2: Unwarp the screen (3%)

The unwrap region I choose is [[1237,1978], [1209,2041], [1396,2023], [1364,2085]] and calculate the pixel value at sub-pixel location with the help of bilinear interpolation. After unwrapping the original QR code on the screen, the below image is the output result.

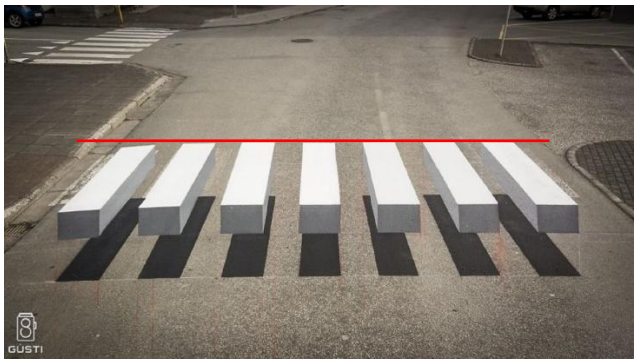
Detectable QR code

The decoded link is http://media.ee.ntu.edu.tw/courses/cv/19F/

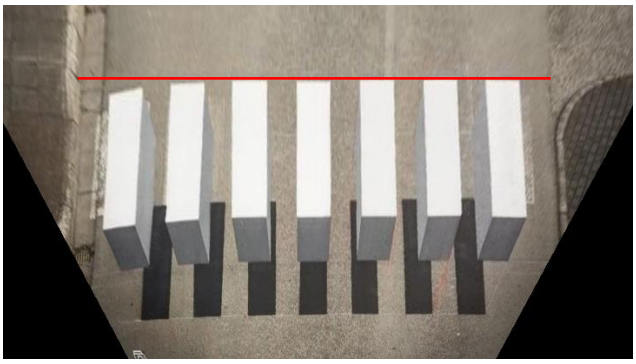
Part 3: Unwarp the 3D Illusion (3%)

After I unwrapped the input image to the top view of it, the output result is shown below. It is obvious that I can't get the parallel bars from the output image.

Input Image



Output Image



In the input image, the red line indicating the extension of top side of the middle bar showed that not every top side of bars is on the red line, especially for the leftmost bar. Therefore, if we change the white part of the bar into a rectangular, those bars will definitely not be parallel like the red line in output image not being able to include all seven bars.

Part 4: Simple AR (3%)

The total run time of producing the output video is less than 15 minutes. One of the result frames is shown below.

The algorithm to the simple AR is described below.

1. Using the SIFT Detector to find the keypoints and descriptors

```
# Create the sift
sift = cv2.xfeatures2d.SIFT_create()

# Find the keypoints and descriptors
keypoint, descriptor = sift.detectAndCompute(template,None)
```

```
# gray descriptor (1331,128)
gray_keypoint, gray_descriptor = sift.detectAndCompute(gray_frame,None)
```

2. Brute-Force Matching and ratio test for finding match points between marker image and frame image

```
# BFMatcher
bf = cv2.BFMatcher()
matches = bf.knnMatch(descriptor, gray_descriptor, k=2)

# Apply Ratio test
good = []
for m,n in matches:
    if m.distance < 0.7 * n.distance:
        good.append(m)

#(225,1,2)
marker_points = np.float32([(keypoint[m.queryIdx].pt) for m in good]).reshape(-1,1,2)
frame_points = np.float32([(gray_keypoint[m.trainIdx].pt) for m in good]).reshape(-1,1,2)
```

3. Applying RANSAC to select better points to construct the homography matrix

```
# Find Homography (source, destination)
H, status = cv2.findHomography(marker_points, frame_points, cv2.RANSAC, 5.0)
```

4. Using the calculated matrix to get the points in the video frame

```
# Do the transform
target_point = cv2.perspectiveTransform(ref_point,H)    #(4,1,2)
```

5. Paste the reference image on the calculated position in every frame

Environment Settings

Package	Version
absl-py	0.7.1
backcall	0.1.0
blockdiag	1.5.4
cycler	0.10.0
decorator	4.4.0
funcparserlib	0.3.6
ipdb	0.12.2
ipython	7.8.0
ipython-genutils	0.2.0
jedi	0.15.1
joblib	0.14.0
kiwisolver	1.0.1
matplotlib	3.0.2
numpy	1.15.4
opencv-contrib-python	3.4.2.16
opencv-python	3.4.2.16
parso	0.5.1
pexpect	4.7.0
pickleshare	0.7.5
pip	19.3.1
prompt-toolkit	2.0.10
ptyprocess	0.6.0
Pygments	2.4.2
pyparsing	2.3.0
python-dateutil	2.7.5
scikit-learn	0.21.3
scipy	1.3.1
setuptools	39.0.1
six	1.11.0
torch	1.1.0
torchsummary	1.5.1
torchvision	0.3.0
tqdm	4.31.1
traitlets	4.3.3
wcwidth	0.1.7
webcolors	1.8.1
wheel	0.32.3