ENGN6528 Computer Vision – S2, 2020
Computer Lab-3 (CLab3)

Objectives:

This is CLab-3 for ENGN6528. The goal of this lab is to help you become familiar with, and practice basic multi-view camera geometry, camera calibration, and DLT method for two-view homography estimation. These are the basic building blocks for 3D visual reconstruction systems.

Note, in this lab, you are free to choose Matlab or Python and lab task descriptions are provided with both languages. If you have not used Matlab or Python before, this lab is an opportunity to get you quickly familiar with basic language usages and relevant libraries for image processing and computer vision. Please note that Python is now increasingly used in computer vision, and we encourage you to practise it in this course.

Special Notes:

1. Each Computer Lab task lasts for three weeks and has two lab sessions: session-A and session-B in the first two weeks. Tutors/Lab instructors will provide basic supervision to both sessions. The third week has no lab, which is for you to complete and submit the lab report.
2. Your lab will be marked based on the overall quality of your lab report. The report is to be uploaded to Wattle site before the due time, which is usually on the Sunday evening of the third week after the announcement of computer lab tasks. (e.g. Sunday of Week4 for Clab1).
3. Your submission includes the lab report in PDF format as well as the lab code that generates the experimental result.
4. It is normal if you cannot finish all the tasks within the two 2-hour sessions -- these tasks are designed so that you will have to spend about 9 hours to finish all the tasks including finishing your lab report. This suggests that, before attending the second lab session (e.g. the lab in Week3 for Clab1), you must make sure that you have almost completed 80%.

Academic Integrity

You are expected to comply with the University Policy on Academic Integrity and Plagiarism. You are allowed to talk with / work with other students on lab and project assignments. You can share ideas but not code, you should submit your own work. Your course instructors reserve the right to determine an appropriate penalty based on the violation of academic integrity that occurs. Violations of the university policy can result in severe penalties.

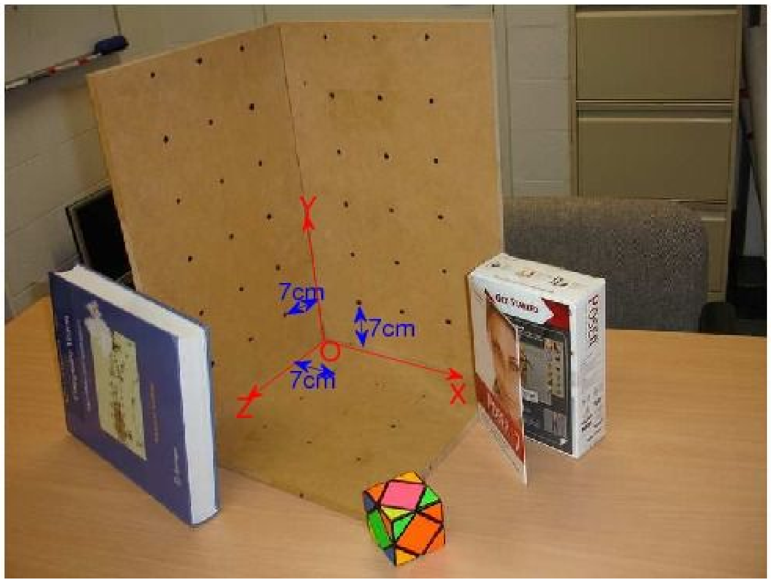
C-Lab-3 Tasks

Task-1: 3D-2D Camera Calibration (6 marks)

(Acknowledgement: Lab material courtesy of Professor Du Huynh of UWA).

Camera calibration involves finding the geometric relationship between 3D world coordinates and their 2D projected positions in the image.

Four images, *stereo2012a.jpg, stereo2012b.jpg, stereo2012c.jpg, and stereo2012d.jpg* are given for this CLab-3. These images are different views of a calibration target and some objects. For example, the diagram below is *stereo2012a.jpg* with some text superimposed onto it:



(Do not directly use the above image for your camera calibration work
as it has been scaled for illustration. Use the original (unlabelled) image files provided.)

On the calibration target there are 3 mutually orthogonal faces. The points marked on each face form a regular grid. They are all 7cm apart.

Write a function to the following specification:

MATLAB user:

```
function C = calibrate(im, XYZ, uv)
```

Python user:

```
def calibrate(im, XYZ, uv)
    return C
```

Input:

im: is the image of the calibration target.

XYZ: is a *N* x 3 array of *XYZ* coordinates of the calibration target points.

uv: is a *N* x 2 array of the image coordinates of the calibration target points.

Output:

C: is the 3 x 4 camera calibration matrix.

- The variable *N* should be an integer greater than or equal to 6.
- This function plots the *uv* coordinates onto the image of the calibration target. It also projects the *XYZ* coordinates back into image coordinates using the calibration matrix and plots these points too as a visual check on the accuracy of the calibration process. Lines from the origin to the vanishing points (namely, the world coordinate system) in the *X*, *Y*, and *Z* directions are overlaid on the image.
- The mean squared error between the positions of the *uv* coordinates and the projected *XYZ* coordinates is also reported.

From the four supplied images (*stereo2012a.jpg, stereo2012b.jpg, stereo2012c.jpg, and stereo2012d.jpg*), choose any image to work on. Use the suggested right-hand coordinate system shown in the diagram above and choose a sufficient number of calibration points on the calibration target.

Store the *XYZ* coordinates in a file so that you can load the data into Matlab and Python (you can choose your preferred data type, for instance, mat in MATLAB and numpy array in Python) and use them again and again. Note that each image can be calibrated independently, so you can choose different calibration points to calibrate each image. Neither do the numbers of calibration points need to be the same for your chosen images.

The *uv* coordinates can be obtained using the MATLAB function *ginput*. If one invokes *ginput* as follows:
```
>> uv = ginput(12)    % e.g., to digitise 12 points in the image
```
and digitises a series of points by clicking with the left mouse button, then *uv* will be a matrix containing the column and row coordinates of the points that you digitised.

(As for Python users, you can get *matplotlib.pyplot.ginput* to get *uv* coordinates. The operation is similar with that in MATLAB)
https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.ginput.html

After the above operation, the variable *uv* should be a 12 × 2 matrix, each row of which should contain the coordinates of one image point.

Note: You need to ensure that, for each image, the numbers of 3D and 2D calibration points are the same. Thus, if your *uv* variable is a 12 × 2 matrix, then your *XYZ* variable should be a 12 × 3 matrix. Also, the points should appear in the same order in the two matrices.

Use the DLT algorithm to solve the unknown camera calibration matrix of size 3x4. Refer to lecture notes, and/or the textbooks: *Multiple View Geometry in Computer Vision Section 7 (page 178)*, *Computer Vision: Algorithms and Applications (Section 6.2)*, or any online resource to set up the DLT linear matrix equation that needs to be solved.

You will end up setting up an equation of the form:

```
    A q = b
```

where:
`A` is a $2N \times 11$ matrix of constraint coefficients (eg, $N = 12$).
`q` is an 11 element column vector of calibration matrix coefficients to be solved.
`b` is a $2N$ column vector of the image coordinates of the target points. This is the vector that is composed of the coordinates stored in the *uv* matrix.

This over-constrained set of equations can be solved (in a least squares sense) using the expression

```
    q = A \ b
```

In this case where the matrix equation is over-constrained, you need to solve the least square solution. A good attempt starts from searching how to use *mldivide* in MATLAB or *np.linalg.lstsq* in Python.

Hints

1. In writing your code you will want to 'reshape' a 2D vector into a 1D vector. You can use the function *reshape* or *np.reshape* (in MATLAB and Python respectively) to reshape a matrix to arbitrary dimensions.
2. You can save your calibration matrices using Matlab's save command. For example, the command
   ```
   >> save mydata im1 im2 calibPts uv1 uv2 C1 C2   % (do not use
   commas between variable names)
   ```
   will save your variables *im1 im2, calibPts, uv1, uv2, C1,* and *C2* in a file called *mydata.mat* . At a later date you can reload this data with the command:
   ```
   >> load mydata
   ```
   The variables *im1 im2, calibPts, uv1, uv2, C1,* and *C2* will then be restored in your workspace.
3. Python users would handle the saving and loading problem by *np.save* and *np.load*.
   https://docs.scipy.org/doc/numpy/reference/generated/numpy.save.html
   https://docs.scipy.org/doc/numpy/reference/generated/numpy.load.html

For Task-1, you should include the followings in your Lab-Report PDF file:

1. List the *calibrate* function in your PDF file. (2 marks)
2. List which image you have chosen for your experiment and display the annotated image in your PDF file. (1 mark)
3. List the 3x4 camera calibration matrix *C* that you have calculated. (1 mark)
4. Decompose the *C* matrix into *K, R, t*, such that $C = K[R|t]$, by using the following provided code (*vgg_KR_from_P.m* or *vgg_KR_from_P.py*). List the results in your PDF file. (0.5 marks)

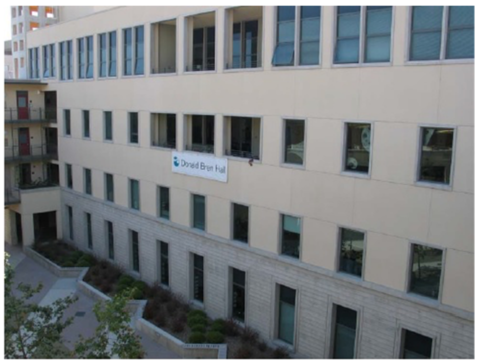
5. Answer the following questions:
    a. What is the focal length of the camera? (0.75 marks)
    b. What is the pitch angle of the camera with respect to the X-Z plane in the world coordinate system? (Assuming the X-Z plane is the ground plane, then the pitch angle is the angle between the camera's optical axis and the ground-plane) (0.75 marks)

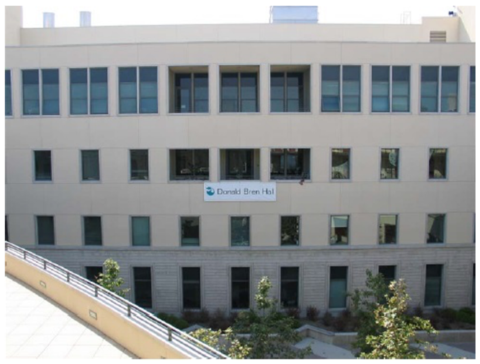
Task-2: Two-View DLT based homography estimation (4 marks)

A transformation from the projective space P^3 to itself is called homography. A homography is represented by a 3x3 matrix with 8 degree of freedom (scale, as usual, does not matter)

$$\begin{bmatrix} x^C w \\ y^C w \\ w \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{bmatrix} x^R \\ y^R \\ 1 \end{bmatrix}$$

The goal of this task is to use the DLT algorithm to estimate a 3x3 homography matrix.


(a) Left


(b) Right

Pick any 6 corresponding coplanar points in the images *left.jpg* and *right.jpg* and get their image coordinates.

In doing this step you may find it useful to check the Matlab function *ginput*.

Calculate the 3x3 homography matrix between the two images, from the above 6 pairs of corresponding points, using the DLT algorithm. You are required to implement your function in the following syntax.

MATLAB user:
```
    function H = homography(u2Trans, v2Trans, uBase, vBase)
```
Python user:
```
    def homography(u2Trans, v2Trans, uBase, vBase)
          return H
```

Input:

u2Trans, v2Trans: vectors with coordinates u and v of the transformed image point (p')

uBase, vBase: vectors with coordinates u and v of the original base image point p

Output

H: a 3x3 Homography matrix

- Computes the homography H applying the Direct Linear Transformation

---

In doing this lab task, should include the followings in your lab report:

1. List your source code for homography estimation and display the two images and the location of six pairs of selected points. (2 marks)
2. List the 3x3 camera homography matrix H that you have calculated. (1 mark)
3. Warp the left image according to the calculated homography. Study the factors that affect the rectified results, e.g., the distance between selected points. (1 mark)

============= END of CLab-3 ============