

ENGN4528Computer Vision - 2020 S1

Computer Lab 3 (CLab-3)

Objectives:

This is CLab-3 for ENGN4528/6528. The goal of this lab is to help you familiar with, and practice:

- Basic multi-view camera geometry, camera calibration. They are the basic building blocks for 3D visual reconstruction systems.
- DLT method for two-view homography estimation.

Note, however, in all the lab task descriptions given below, we by default use Matlab as the preferred language. You are however free to choose Python if you are more comfortable with Python. If you have not used Matlab or Python before, this lab is an opportunity to get you to quickly familiar with basic language usages of Matlab/Python for image processing and computer vision.

Special Notes:

1. Each computer lab has three weeks: session-A and session-B. Tutors/Lab instructor will provide basic supervision to both sessions.
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 Week-3 session of your lab.
3. 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 third lab session (in Week-2 of each CLab), you must make sure that you have almost complete 80.

Academic Integrity

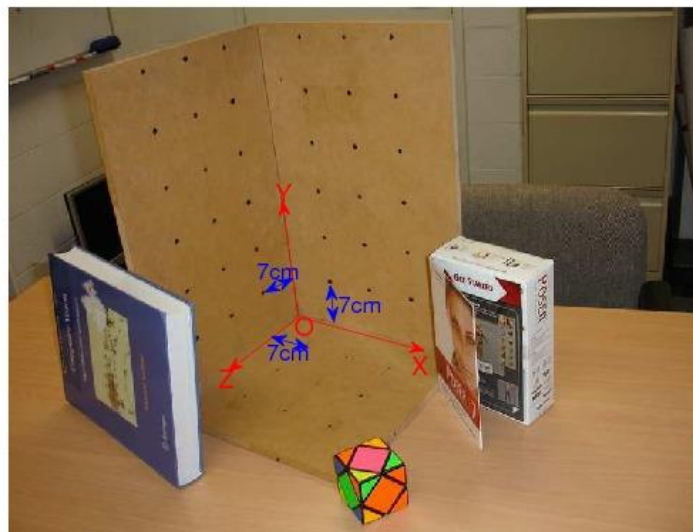
You are expected to comply with the University Policy on Academic Integrity and Plagiarism. You are allowed to talk with / work with other students on lab and project assignments. You can share ideas but not code, you should submit your own work. Your course instructors reserve the right to determine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

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 Matlab function to the following specification

Function to perform camera calibration

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

Input: `im`: is the image of the calibration target.
 `XYZ`: is a $N \times 3$ array of XYZ coordinates of the calibration target points.
 `uv`: is a $N \times 2$ array of the image coordinates of the calibration target points.

Outputs: `C`: is the 3×4 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 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.

Generally, we ask you to implement a function:

MATLAB user:

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

Python user:

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

From the 4 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 datatype, for instances, **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 3×4 . Refer to lecture notes (Week7 3D vision), textbook,

Multiple View Geometry in Computer Vision, Section 7, page 178., or

Computer Vision: Algorithms and Applications Section 6.2, or online resources to set up the DLT linear equation matrix that needs to be solved.)

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

$$\mathbf{A} \mathbf{q} = \mathbf{b}$$

where:

\mathbf{A} is a $2N \times 11$ matrix of constraint coefficients (in this case, $N=12$).

\mathbf{q} is an 11 element column vector of calibration matrix coefficients to be solved.

\mathbf{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 \mathbf{uv} matrix.

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

$$\mathbf{q} = \mathbf{A} \setminus \mathbf{b}$$

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.

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

5. Answer the following questions:

- what is the focal length of the camera?

- what is the pitch angle of the camera with respect to the X-Z plane in the world coordinate system? (Assuming the X-Z plan is the ground plane, then the pitch angle is the angle between the camera's optical axis and the ground-plane.)

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.



Fig. 1. The left images shows one of the almost planar facades of the Bren School with a large perspective distortion. The goal is to construct a projective transformation that rectifies the image producing a result similar to the right image.

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 DLT algorithm. You are required to implement your function in the following syntax.

Function to calculate homography matrix

H = homography (u2Trans, v2Trans, uBase, vBase)

Usage: Computes the homography H applying the Direct Linear Transformation
Inputs: u_{2Trans} , are vectors with coordinates u and v of the transformed image
 v_{2Trans} : point (p')
 u_{Base} , are vectors with coordinates u and v of the original base
 v_{Base} : image point p
Output: H : is a 3×3 Homography matrix
Name and date need to be mentioned as a signature.

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. List the 3×3 camera homography matrix H that you have calculated.
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.

===== **End of CLab-3** =====