School of Computing
National University of Singapore
CS4243 Computer Vision and Pattern Recognition
Semester 1, AY 2016/17

Lab 6,7
(due on 21st Oct 2016 Friday before noon)

Objectives:

- To revise the camera projection models that we had learned in class
- To practice how to represent rotation using quaternions
- To practice how to compute a homography matrix

Part 1. Defining the scene points and camera position and orientations

Write python codes to do the following:

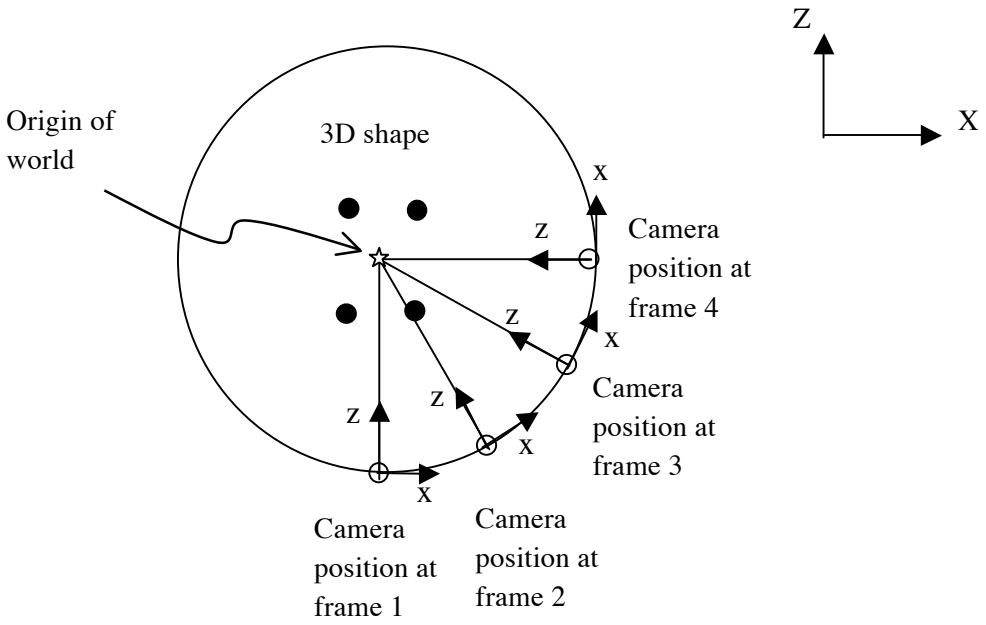
1.1 Define the Shape

Define a 11x3 array to store the following shape points:

```
pts       = np.zeros([11, 3])
pts[0, :] = [-1, -1, -1]
pts[1, :] = [1,  -1, -1]
pts[2, :] = [1,   1, -1]
pts[3, :] = [-1,  1, -1]
pts[4, :] = [-1, -1,  1]
pts[5, :] = [1,  -1,  1]
pts[6, :] = [1,   1,  1]
pts[7, :] = [-1,  1,  1]
pts[8, :] = [-0.5, -0.5, -1]
pts[9, :] = [0.5,  -0.5, -1]
pts[10,:] = [0, 0.5, -1]
```

1.2 Define the Camera Translation

Continuing from the codes written so far, define the positions of camera for 4 frames. The camera is assumed to start from the position (0, 0, -5). It is supposed to move along a circular arc, on the XZ plane, at 30 degrees intervals from frame to frame.



Define a function to perform quaternion multiplication, given two input quaternions.

The python code is:

```
def quatmult(q1, q2):
    # quaternion multiplication
    out = [0, 0, 0, 0]      # output array to hold the result
    ---- you need to fill in the code here ----
    return out
```

Call the quarternion multiplication function to rotate the camera initial position (i.e. (0, 0, -5)) with respect to the Y-axis (direction of Y-axis is “into” paper) to obtain the camera locations for frames 2, 3 and 4. Note that the axis of rotation is Y-axis, angle of rotation is -30 degrees. Be careful to convert angles to radians because python cosine and sine functions take input in radians.

1.3 Define the Camera Orientation

Define a function to return a 3x3 rotation matrix parameterized by the elements of a given input quaternion.

The python code is:

```
def quat2rot(q):
    ---- you need to fill in the code here ----
    return np.matrix( -- the 3x3 array that you had formed -- )
```

Let the orientation of the camera at the first frame be *I*, the identity matrix. For the next three frames, we will rotate the camera by an increment of -30 degrees at each step. The axis of rotation is the Y-axis. Note that by doing so, the camera Z-axis will always point towards the origin of the world coordinate system *(we are doing this for ease of lab simulation. In the real world, the camera can have arbitrary movement)*.

Important: In this case, you will need to set the angle of rotation to be +30 degrees instead of –30 degrees. Think about why this is so. You need not submit the answer to this question, but you should think about it.

We want to obtain the 3x3 matrix representing the orientation of the camera at each of the other three frames. To be specific, for each of the 3x3 matrix, the first row represents the X-axis direction, the second row represents the Y-axis direction, and the third row represents the Z-axis direction, all with respect to the world coordinate system.

Call the function quat2rot that you defined above to return you the 3x3 rotation matrix. Obtain the 3x3 matrix representing the orientation of the camera. Call this matrix ***quatmat_i***, where ***i*** is the frame number

Part 2. Projecting 3D shape points onto camera image planes

Using the 3D shape points defined in 1.1, and the camera translation and orientation for the four frames defined in 1.2 and 1.3, project the 3D shape points onto the image planes for all the four frames. You need to do both the projection models that we have studied in class: orthographic and perspective.

Use the following values for the various parameters when applicable:

$u_0 = 0$
$v_0 = 0$
$\beta_u = 1$
$\beta_v = 1$
$k_u = 1$
$k_v = 1$
$focal\_length = 1$

Display your results in 2 figures – one figure for each projection model. Since there are four frames (images) for each projection models, you need to use the subplot command to squeeze 4 plots into one single figure. *Hint:* you may use the add_subplot method of plt.figure().

Part 3. Homography

Using the points defined in 1.1, specifically pts[0], pts[1], pts[2], pts[3], pts[8], compute a homography matrix mapping these points to their image points on camera frame-3.

Normalize your homography matrix such that the element $h_{33} = 1$.

Submission Instruction

Submit the following to IVLE by the due date (before noon on 21st Oct 2016):

1. The softcopy of your Python program.
2. The softcopy of the 2 figures (orthographic projection and perspective projection)
3. A softcopy document showing the 3x3 homography matrix, with h_{33} element normalized to 1.

Please put your files in a folder and submit the folder. Use the following convention to name your folder:

StudentNumber_yourName_Lab#. For example, if your student number is A1234567B, and your name is Chow Yuen Fatt, for this assignment, your file name should be A1234567B_ChowYuenFatt_Lab6n7.