School of Computing
National University of Singapore
CS4243 Computer Vision and Pattern Recognition
Semester 1, AY 2018/19

Lab 6,7
(due on 21st Oct 2018 Sunday 2359hrs)

Objectives:
- To revise the camera projection models that we had learned in class
- To practice how to represent rotation in various ways, including the use of quaternions
- To practice how to compute a homography matrix

Write a Matlab program to do the following:

Q1 Define the 3D Shape

Define a 8x3 array (matrix) to store the following shape points:

(-1 -1 -1)
(1 -1 -1)
(1 1 -1)
(-1 1 -1)
(-1 -1 1)
(1 -1 1)
(1 1 1)
(-1 1 1)

program statements:

pts = zeros(8, 3);

pts(1, :) = [-1 -1 -1];

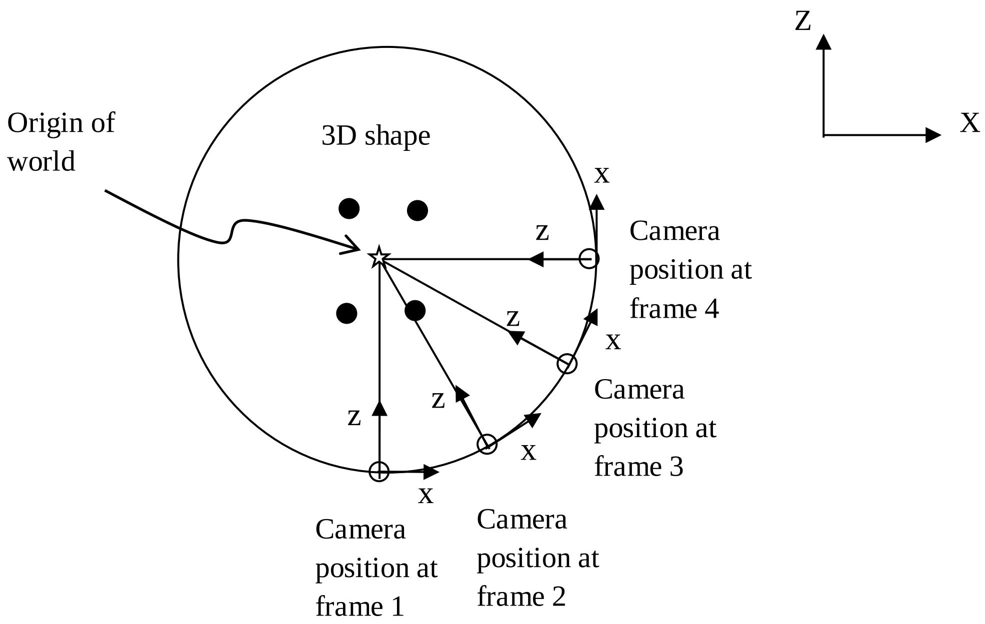
pts(2,:) = [1 -1 -1];

and so on.

Run the codes and print out the coordinates of the points to make sure you have defined them correctly.

Q2 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.



Use quaternion multiplication 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 matlab cosine and sine functions take input in radians. In matlab, π is “pi”.

Print the locations of the camera in all 4 frames, using the following statements:

cam_pos = zeros(4, 3);

cam_pos(1, :) = [0 0 -5];

%--- use quaternion multiplication to rotate the first position to obtain the positions for

%-- frames 2, 3, and 4. Store the results in the array cam_pos. Then use the

%-- following statements to print out the resultant positions

disp(‘camera location 1’);

cam_pos(1,:)

and so on.

Q3 Define the Camera Orientation

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, we also want the camera Z-axis to 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)*.

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.

Using the roll, pitch and yaw representation, obtain the 3x3 matrix representing the orientation of the camera. Call this matrix ***rpymat_i***, where ***i*** is the frame number .

Print the results using the following statements:

disp('using roll, pitch and yaw representation for rotation, the computed matrices are : ');

rpymat_1

rpymat_2

rpymat_3

rpymat_4

Using a 3x3 rotation matrix with its elements given by the entries of a quaternion, obtain the 3x3 matrix representing the orientation of the camera. Call this matrix ***quatmat_i***, where ***i*** is the frame number. Print the results using the following statements:

disp('using quaternion representation for rotation, the computed matrices are : ');

quatmat_1

quatmat_2

quatmat_3

quatmat_4

Verify that ***rpymat_i*** is equal to ***quatmat_i*** by computing ***rpymat_i - quatmat_i.***

Q4 Projecting the 3D Shape onto Camera Image Planes

Using the 3D shape points defined in Q1, and the camera translation and orientation for the four frames defined in Q2 and Q3, project the 3D shape points onto the image planes for all the four frames. You need to do both the projection models that we had studied in class i.e. orthographic and perspective.
Use the following values for the various parameters:

$$u_0 = 0, \ \ v_0 = 0$$

$$(\beta_u = 1, \qquad \beta_v = 1) \ \ i.e. \ (k_u = 1, \qquad k_v = 1)$$

$$foal\ length = 1$$

Save the projected points into the matrices *U* and *V* :

```
nframes = 4;                    % number of frames
npts    = size(pts,1);          % number of 3D points
U = zeros(nframes, npts);       % array holding the image coordinates (horizontal)
V = zeros(nframes, npts);       % array holding the image coordinates (vertical)
```

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 to squeeze 4 plots into one single figure. Use the following program statements to plot the results:

```
% repeat the following for each of the 2 projection models figure;
for fr = 1 : nframes
  subplot(2,2,fr), plot(U(fr,:), V(fr,:), '*');
  for p = 1 : npts
     text(U(fr,p)+0.02, V(fr,p)+0.02, num2str(p));
  end
end
```

Q5 Homography

Using the points defined in Q1, specifically the first 4 points, 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:

1. The softcopy of your Matlab 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.