Lab 1

Camera Calibration

Objective

In this experiment, students will use stereo cameras, an image acquisition program and camera calibration algorithms to achieve the following goals:

1. Develop a procedure for camera calibration.
2. Measure the error in calibration using two approaches:
   i. A simple Least Square method
   ii. A non-linear method using a well-designed calibration toolbox.

Reference Materials

- Lecture Notes, i.e. camera calibration using least square approach
- Zhang's paper: "A Flexible New Technique for Camera Calibration"
- Caltech Camera Calibration Toolbox.

Pre-lab

Please read the following materials:

- Lecture notes about "Reference (Coordinate) Frames", "Homogeneous Transformation Matrix (HTM)", "Camera Model", "Linear Method Camera Calibration", "Non-Linear Method Camera Calibration (specially Zhang's Method)
- Zhang's paper: "A Flexible New Technique for Camera Calibration" "Getting Started" section for "Camera Calibration Toolbox for Matlab": http://www.vision.caltech.edu/bouguetj/calib_doc/#start

Briefly describe functions required from calibration toolbox.

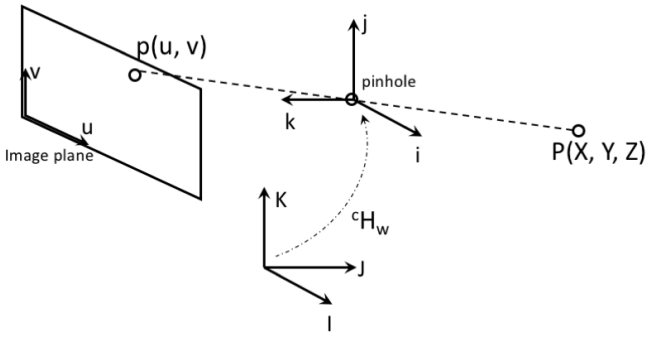
Background

Camera calibration is a procedure to determine a mathematical model (i.e. pinhole) for the camera in which relation between points outside world and image is formulized. The intrinsic and extrinsic parameters relate the camera's coordinate system to idealized and world coordinate systems, respectively. In this lab, we use

two cameras (left and right) next to each puma robot. The pinhole camera model can be expressed as following:

$$z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K[R \quad T] \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

$$K = \begin{bmatrix} \alpha_x & \gamma & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$



where R is the rotation matrix and T is the translation vector. u and v are pixel coordinates and z_c is the scaling factor. x_w, y_w, and z_w are coordinate of the point with respect to world coordinate frame.

Lab Procedure

This lab consists of the following three parts:

Part 1: Least Square method for camera calibration

1. Take five snapshots of the calibration pattern using both left and right cameras. Each snapshot should be taken with the pattern at a known XY position and at different heights, parallel to the table. You can use the green foams in the lab to raise the pattern. The pattern must be moved only vertically for each of the three heights. That is, there is no translation in the XY direction when moving up for each pattern. To capture left and right images, you can use `puma#_save_left` and `puma#_save_right` where # is the robot number. To see stream video of cameras you can use `mvid_stream -i port -d IP -2` where `port` and `IP` are:

robot	port	IP
puma1	1500	10.14.1.100
puma2	1500	10.14.1.101
puma3	1600	10.14.1.100

2. Use corner extraction tool in the Camera Calibration Toolbox to determine the pixel coordinate of each corner in the pattern and create a file with XYZ-UV information for the calibration of each camera. You can find the toolbox under `/usr/share/TOOLBOX_calib`. Add this directory to Matlab search path.
3. Write a simple Matlab program to find the calibration matrix (intrinsic and extrinsic parameters) using the least square approach. The suggestion is to use the method in your lecture notes. That is, re-rewriting the equations to construct one single matrix, A, and minimizing $Aq = 0$ under the constraint $\|q\|=1$, where q contains the calibration parameters in a column vector.

NOTE:
In this case the calibration toolbox won't be able to determine the Z coordinate of the pattern. Therefore, make sure you record the different heights at which you placed the pattern and correct the Z coordinate in your code accoringly.

Part 2: Using a non-linear method for camera calibration

In this case you will use the Calibration toolobox.

1. Position the pattern this time at random poses with respect to the camera and take five snapshots with each camera. You may pick any random position, but make sure each one is reasonably different from the others (rotation and translation) and that at all poses, the pattern can be clearly seen by both cameras.

Also make sure that one such pose of the pattern is well known with respect to the robot world coordinate frame – this will be the reference frame to establish the camera to robot/world transformation. (Hint: use the same initial position used for part 1)

2. Run the Matlab programs provided in the Calibration Toolbox to obtain the calibration matrix (intrinsic and extrinsic parameters) and distortion parameters – the program may use Zhang's method or any other approach with distortion parameters modeling. You should also save the distortion parameters and image corner points.

Part 3: Error Measurement

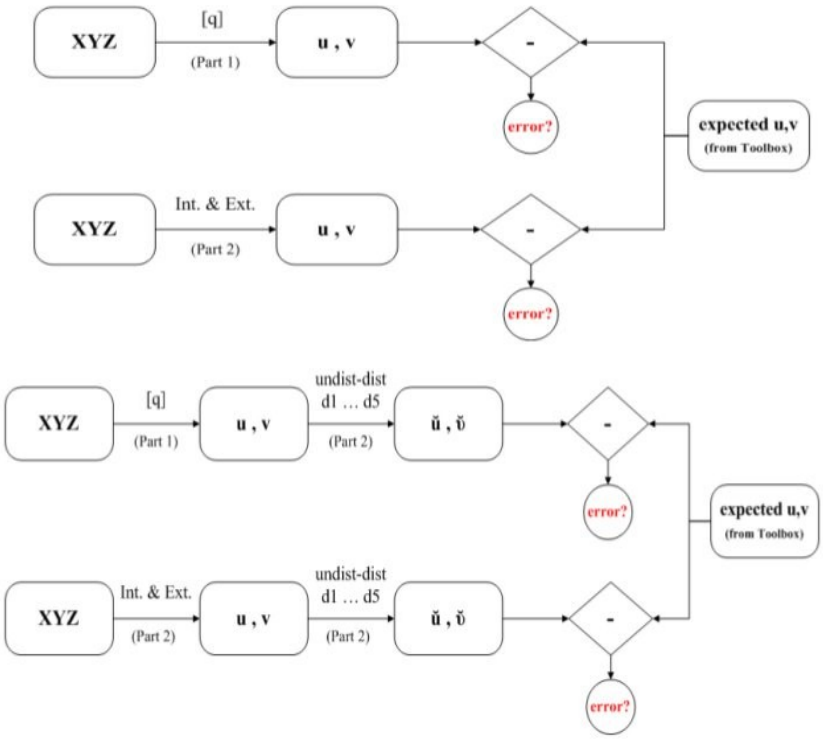
At this point you have collected training images in order to learn the parameters for camera calibration using two different methods. Next step is to test and compare results of the calibration.

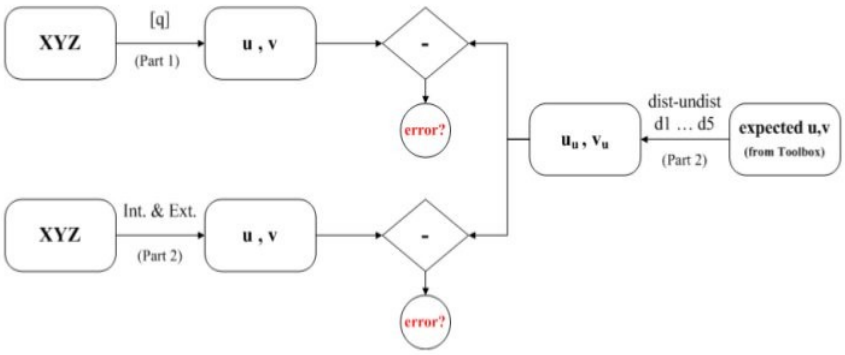
1. Take another snapshot of the pattern with known XYZ position for testing purpose. (Hint: record the XYZ position with respect to the world coordinate frame). Make sure that the image of this pattern is different from all the images used for the calibration.

2. Extract the corners of the test image using Camera Calibration Toolbox and save the XYZ-UV coordinate of the pattern. You will use these coordinates as ground truth to compare with results obtained from the calibration parameters learnt in parts 1 and 2.
3. Calculate the error. That is a) observed UV coordinates in the image vs the estimated UV coordinates using the calibration matrix derived in Part 1. b) observed UV coordinates in the image vs the estimated UV coordinates using the calibration matrix derived in Part 2.
4. In step 3 above, you are comparing normalized UV coordinates to distorted coordinates. Therefore, for this step you should take distortions into acount – i.e. Repeat step (3) for undistorted and distorted UV coordinates.

Hints:

- Use Weng's or Zhang's error measurement technique for converting from normalized UV coordinates to undistorted normalized UV coordinates and vice versa.
- Use the distortion parameters saved in Part 2.
- Flowcharts bellow give a visual explanation for what you need to do.





Post-Lab Questions:

Comment your results by answering to below questions:

- what you expected the error to be and why?
- what error did you get and why?

Include all your results, and describe steps you did in your implementation.