Calibration Tutorial

Kaori Hagihara

Communications and Remote Sensing Laboratory, Ecole Polytechnique de Louvain,UCL, Belgium

This document is aimed to explain our process to calibrate cameras with fish-eye lens attached on each, and especially for people who want to make BOSS video frames distortion free. We remove lens distortion by simply projecting the image from omni-directional camera model to pinhole model, and compute extrinsic parameters by defining the pinhole model. Our process includes three steps on each camera; Intrinsic calibration, Projection map computation and undistortion, and finally extrinsic calibration.

This is just a suggestion and you might find the other better ways.

Note: OpenCV library [4] is required to compile every sample C++ code under src/ directory.

Intrinsic Calibration

Our intrinsic calibration is basically performed by OcamCalib Toolbox [1][2][3]. Refer to the following website:
http://asl.epfl.ch/~scaramuz/research/Davide_Scaramuzza_files/Research/OcamCalib_Tutorial.htm

Set the parameters required in OcamCalib as following:

Number of squares along the X direction = 5

Number of squares along the Y direction = 8

Size dX of each square along the X direction =27mm

Size dY of each square along the X direction =27mm

Note: Don't forget to save the calibration results (**Omni_Calib_Results.mat**).

Projection map Computation and undistortion

Once the calibration results are acquired, you will compute a projection map into pinhole model. If you have restarted Matlab, you have to load **Omni_Calib_Results.mat**. We prepared the following Matlab functions for this computation; **Lookup.m**, **undistort.m**

Here is sample Matlab codes:

>> // Define pinhole model

>>focal=145; // focal length

>>shiftX=-30; // difference between the position of optical center and of image center

>>shiftY=10;

>> distance=600; // distance between the optical center and the end of the projection plane

>>prec = 10; // precision

>>rt=lookup(prec,ocam_model,focale,distance);

>>X_coord = zeros(height, width);

>>Y_coord = zeros(height, width);

>>[im_und,corners_und,X_coord,Y_coord]=undistort(I_1,rt,prec,Xp_abs(:,1,1),Yp_abs(:,1,1),ocam_model,focale,shiftX,shiftY);

>>

>>save Xcoord.csv -ASCII -TABS Y_coord;

>>save Ycoord.csv -ASCII -TABS X_coord;

>>// the folloing lines are an option to display undistorted image

>>figure

>>imshow(im_und,[0 255])

Now, you got files "**Xcoord.csv**" "**Ycoord.csv**" containing pixel map table, where pixel at (Xcoord[x,y], Ycoord[x,y]) on mapped images corresponds to (x,y) in original images.

The map table generated by the above code corresponds to the following camera intrinsic parameters.

Focal length: 145

image center: (360, 288)

This definition need to be written into a text file (e.g. "**intrinsicMat.ini**") and saved for running a program, which computes extrinsic parameters, in a matrix format as following:

$$\begin{matrix} 145 & 0 & 360 \\ 0 & 145 & 288 \\ 0 & 0 & 1 \end{matrix}$$

A sample C++ code "**projectimg.cc**" might be useful to reconstruct distortion free images.

% *./projectimg input_image_filename* **Xcoord.csv Ycoord.csv** *output_image_filename*

Note: The defined intrinsic parameters and computing extrinsic parameters are applicable just for projected images.

Extrinsic Calibration

Extrinsic calibration is performed using a projected image containing a cubic calibration target. Pixel coordinate information of chess-board corners on the projected image and 3D coordinate information at corresponding Chess-board corners are the input to compute extrinsic parameters.

You can use the following information to define the world coordinate system.

Cube dimention: 930*930*930 (mm)

Chess-board dimension on each face:

Number of squares along the X direction = 5

Number of squares along the Y direction = 5

Size dX of each square along the X direction =180mm

Size dY of each square along the Y direction =180mm

An interactive program "**recordCalPnts**" facilitates collecting pixel coordinate information of chess-board corners on the projected image. This program outputs an annotated image "**pointRefImg.jpg**" of original image (Ref: Illustration 1) and a text file "**pointsInfo.csv**" containing the list of point coordinate at pixels which you have clicked (Ref: Insider red frame of Illustration 2).

% *./recordCalPnts image_file_name output_directory*

Mouse control:

Left click: Pick the corner pixel information

The index number is represented at the position where you release the bottom

Yellow point: Clicked position

Red point: Detected corner position

Right click: Remove the last corner pixel information

Keyboard control:

↓ and T: Enlarge the displayed image

← and Q: Reduce the displayed image

ESC: Save the files and quite the program

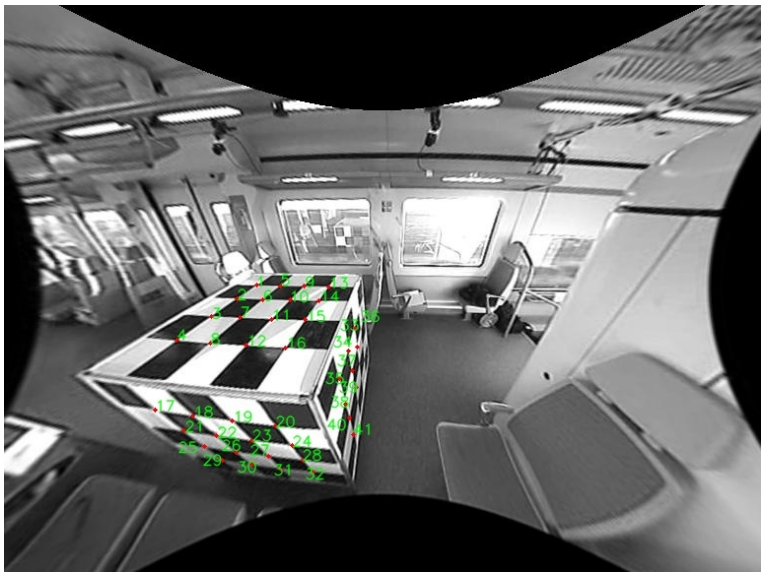
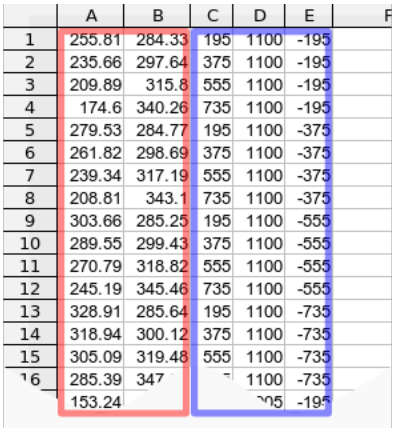


Illustration 1: The image "pointRefImg.jpg" represents all the points you clicked and index of each.

Once you finish the program "**recordCalPnts**", you open the output file "**pointsInfo.csv**" and add 3D coordinate information at each point as Illustration 2.



	A	B	C	D	E	F
1	255.81	284.33	195	1100	-195	
2	235.66	297.64	375	1100	-195	
3	209.89	315.8	555	1100	-195	
4	174.6	340.26	735	1100	-195	
5	279.53	284.77	195	1100	-375	
6	261.82	298.69	375	1100	-375	
7	239.34	317.19	555	1100	-375	
8	208.81	343.1	735	1100	-375	
9	303.66	285.25	195	1100	-555	
10	289.55	299.43	375	1100	-555	
11	270.79	318.82	555	1100	-555	
12	245.19	345.46	735	1100	-555	
13	328.91	285.64	195	1100	-735	
14	318.94	300.12	375	1100	-735	
15	305.09	319.48	555	1100	-735	
16	285.39	347		1100	-735	
	153.24			205	-195	

Illustration 2: Text file "pointsInfo.csv" contain tab-spaced values. Values inside red frame represents the list of pixel coordinate information, and Values inside blue frame represents 3D coordinate information. Each line corresponds to the same point.

Finally, we compute the extrinsic Matrix by executing "**extCalibration**".

>>*./extCalibration* ***pointsInfo.csv*** **intrinsicMat.ini** *projection_image output_directory_path*

The *projection_image* is an input image to project the 3D points on the list in "***pointsInfo.csv***". This program output a file "**extrinsicPars.ini**" containing Translation and Rotation Vector information as following:

Translation Vector		
Rotation Vector		
-1001.64	821.301	2019.86
-1.79213	-0.495429	2.018

References:

[1] D. Scaramuzza, A. Martinelli, and R. Siegwart. A flexible technique for accurate omnidirectional camera calibration and structure from motion. Proceedings of IEEE International Conference on Computer Vision Systems, page 45, 2006.

[2] D. Scaramuzza, A. Martinelli, and R. Siegwart. A toolbox for easily calibrating omnidirectional cameras. Proc. of the IEEE International Conference on Intelligent Systems, IROS06, Beijing, China, 2006.

[3] M. Rufli, D. Scaramuzza, and R. Siegwart. Automatic detection of checkerboards on blurred and distorted images. In /IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008.

[4] Open Computer Vision Library, http://sourceforge.net/projects/opencvlibrary/