Camera Calibration

Camera calibration is the process of estimating two sets of parameters: the intrinsic parameters and the extrinsic parameters. This laboratory is based on Shireen Elhabian: A Tutorial on Camera Calibration, CVIP Lab – 2007.

- ***The intrinsic parameters*** matrix *K* defined as:

$$K = \begin{bmatrix} \alpha & -\alpha\cot\theta & u_0 \\ 0 & \beta/\sin\theta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

- ***The extrinsic parameters*** are a 3-D translation vector, t = [t_x t_y t_z]^T and a rotation matrix R:

$${}^{C}_{W}R = \begin{pmatrix} \cos\theta_y\cos\theta_z & \cos\theta_z\sin\theta_x\sin\theta_y - \cos\theta_x\sin\theta_z & \sin\theta_x\sin\theta_z + \cos\theta_x\cos\theta_z\sin\theta_y \\ \cos\theta_y\sin\theta_z & \sin\theta_x\sin\theta_y\sin\theta_z + \cos\theta_x\cos\theta_z & \cos\theta_x\sin\theta_y\sin\theta_z - \cos\theta_z\sin\theta_x \\ -\sin\theta_y & \cos\theta_y\sin\theta_x & \cos\theta_x\cos\theta_y \end{pmatrix} \tag{2}$$

- ***The projection matrix*** is the transformation between the 3-D world and the 2-D image frame:

$$\begin{bmatrix} p \\ 1 \end{bmatrix} = \frac{1}{z_c} M \begin{bmatrix} {}^{w}P \\ 1 \end{bmatrix} \tag{3}$$

- Let *M* = (*A b*), with a_1^T, a_2^T, a_3^T are the rows of *A,* and b is the 4th column of M, then

$$\rho(Ab) = K(R\,\mathrm{t}), \text{ and } R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{4}$$

Hence,

$$\rho\begin{pmatrix} a_1^T \\ a_2^T \\ a_3^T \end{pmatrix} = \begin{pmatrix} \alpha r_1^T - \alpha\cot\theta\, r_2^T + u_0 r_3^T \\ \dfrac{\beta}{\sin\theta} r_2^T + v_0 r_3^T \\ r_3^T \end{pmatrix} \tag{5}$$

$$\rho = \frac{\varepsilon}{|a_3|}, \text{where } \varepsilon = \pm 1 \tag{6}$$

$$r_3 = \rho a_3 \tag{7}$$

- Let

$a_1 = [m_{11}\ m_{12}\ m_{13}]^T$

$a_2 = [m_{21}\ m_{22}\ m_{23}]^T$

$a_3 = [m_{31}\ m_{32}\ m_{33}]^T$

$b = [m_{14}\ m_{24}\ m_{34}]^T$,

Therefore,

- $u_0 = \rho^2(a_1 \cdot a_3)$ and $v_0 = \rho^2(a_2 \cdot a_3)$

$$\cos\theta = -\frac{(a_1 \times a_3)\cdot(a_2 \times a_3)}{|a_1 \times a_3||a_2 \times a_3|} \tag{8}$$

- Then α and β can be recovered as:

$$\alpha = \rho^2|a_1 \times a_3|\sin\theta \text{ and } \beta = \rho^2|a_2 \times a_3|\sin\theta \tag{9}$$

- The remaining parameters can be computed as;

$$r_1 = \frac{a_2 \times a_3}{|a_2 \times a_3|} \tag{10}$$

$$r_2 = r_3 \times r_1 \tag{11}$$

- $\theta_y = \sin^{-1} r_{13}$, $\theta_x = \cos^{-1}(r_{33}/\cos\theta_y)$, and $\theta_z = \cos^{-1}(r_{11}/\cos\theta_y)$. (12)
- $t = \rho K^{-1} b$ (13)

Task 1: Simulation with Synthetic Data

	translation			*rotation*			f_x	f_y	*Image plane info*		
Parameter	t_x*(mm)*	t_y	t_z	θ_x*(rad)*	θ_y	θ_z	α *(pixel)*	β	u_0	v_0	θ
Ground Truth	-27	-28	701	0.09	0.8	-0.03	556	549	172	121	$\pi/2$

1. For the given camera calibration parameters, obtain the ground truth perspective projection matrix M_g.
2. Generate a set of 3D points (x_w, y_w, z_w).

3. Generate the corresponding projected 2-D points using M_g.
4. Use the singular value decomposition (SVD) method to estimate the projection matrix *M*.
5. Compare between the obtained projection matrix *M* and the M_g, comment on your result.
6. Decompose the obtained projection matrix into its corresponding camera parameters.
7. Compare between the obtained camera parameters and the corresponding ground truth parameters. Comment on your result.
8. Modify your programs to handle the radial distortion case. Provide your results and comments in case of radial distortion.

Task 2: Using Real Data

1. Find calibration pattern images in "your_OpenCV_Dir/samples/data/left01.jpg -- left14.jpg" we will use these files as input for our camera calibration. Copy them to your code folder
2. Find the corner in each chessboard image using cv2.findChessboardCorners function:

```
isfound, corners = cv2.findChessboardCorners(gray, (7,6), None)
```

3. Increase the corner accuracy using cv2.cornerSubPix function

```
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)
corners2 = cv2.cornerSubPix(gray,corners,(11,11),(-1,-1),criteria)
```

4. Draw the corner on image using cv2.drawChessboardCorners

```
IMG = cv2.drawChessboardCorners(IMG, (7,6), corners2, isfound)
```

5. Assuming that the 3D origin point (zero point for 3D coordinate) is on the top left corner of the chessboard), estimate the 3D coordinate for the chessboard corner.
6. Find camera calibration using cv2.calibrateCamera function

```
ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints, imgpoints, gray.shape[::-1],None,None)
```

Note that this function returns also the distortion parameter.

7. Refine the camera matrix based on a free scaling parameter using cv2.getOptimalNewCameraMatrix function
8. Get the undistorted image using the cv2. undistortfunction

```
newcameramtx, roi=cv2.getOptimalNewCameraMatrix (mtx,dist,(w,h),1,(w,h))
```

```
dst = cv2.undistort(img, mtx, dist, None, newcameramtx)
```

```
x,y,w,h = roi
dst = dst[y:y+h, x:x+w]
cv2.imwrite('calibresult.jpg',dst)
```

9. Crop the image using the region of interest obtained from the new camera matrix and show the result and save it.

10. Use one of the calibration patterns at CVIP Lab. Estimate the intrinsic parameters of your cellphone camera.
11. Duplicate the algorithms in the OpenCv tutorial https://docs.opencv.org/trunk/dc/d2c/tutorial_real_time_pose.html, using an object and a camera from the Lab.

Notes: 1) Submit a comprehensive report explaining all steps and label the figures.

2) Upload your report of blackboard AND submit the original in print form by due the date.

References: Shireen Elhabian**:** *A Tutorial on Camera Calibration* – CVIP Lab 2007.