University of Burgundy

MsCV

Autonomous Robotics Module

Camera Auto-calibration Lab

by

Gopikrishna Erabati

Supervisor: Dr. Adlane Habed

1. Introduction

Camera calibration is the process of estimating parameters of the camera using images of a special calibration pattern. The parameters include camera intrinsics, distortion coefficients, and camera extrinsics. Camera auto-calibration is the process of determining internal camera parameters directly from multiple uncalibrated images of unstructured scenes. In contrast to classic camera calibration, auto-calibration does not require any special calibration objects in the scene[1].

Why it's important?

For instance, in 3D reconstruction, the recovery of the calibration parameters of the cameras is significant since it provides metric information about the observed scene, e.g., measures of angles and ratios of distances. Auto-calibration enables the estimation of the camera parameters without using a calibration device (e.g., checkerboard), but by enforcing simple constraints on the camera parameters, such as constant intrinsic parameters in multiple images, known principal point, known pixel shape, etc.

2. Algorithms for camera auto-calibration [2]:

1. The simplified and classical Kruppa's equations.

2. The Mendonça-Cipolla auto-calibration method.

3. Using the absolute dual quadric and its projection, the dual image of the absolute conic.

***Note**: The code is well commented for further reference. There is a README.txt file which explains how to run the code.*

2.1 The classical Kruppa's method

In order to apply Kruppa's method to auto-calibration, we assume that the images are taken from same moving camera with same parameters.

The solution to camera intrinsic parameters for Kruppa's method can be computed by solving the optimization problem for the cost function given below,

$$\mathcal{C}(\alpha_u, \alpha_v, \gamma, u_0, v_0) = \sum_{i,j} || \frac{\mathbf{F}_{ij}\omega^{-1}\mathbf{F}_{ij}^T}{||\mathbf{F}_{ij}\omega^{-1}\mathbf{F}_{ij}^T||_{fro}} - \frac{[\ \mathbf{e}_{ji}\ ]_\times \omega^{-1} [\ \mathbf{e}_{ji}\ ]_\times^T}{||[\ \mathbf{e}_{ji}\ ]_\times \omega^{-1} [\ \mathbf{e}_{ji}\ ]_\times^T||_{fro}} ||_{fro}$$

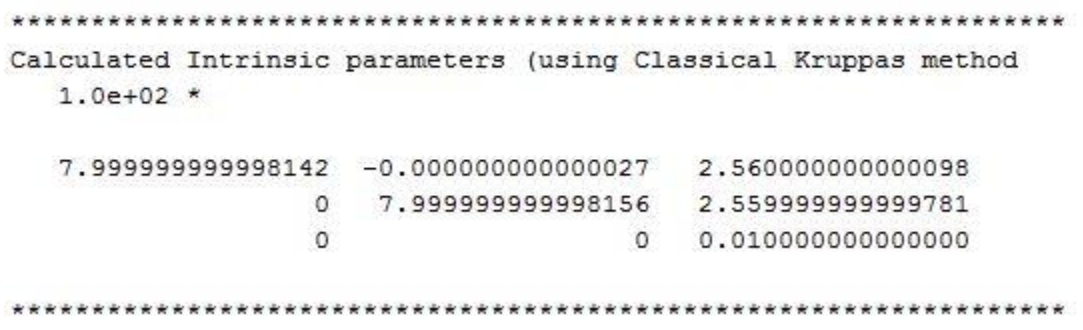
where $||.||_{fro}$ refers to the Frobenius norm.

This is implemented by function,

error = costFunctionKClassical(Fs, x)

The problem I faced with the optimization of cost function is with the tolerance values of the optimizer, so I solved this problem by setting low tolerance value to find minimum.

The result is as shown in Fig. 1



```
*****************************************************************
Calculated Intrinsic parameters (using Classical Kruppas method
   1.0e+02 *

   7.999999999998142  -0.000000000000027   2.560000000000098
                   0   7.999999999998156   2.559999999999781
                   0                   0   0.010000000000000

*****************************************************************
```

Fig. 1 Result of Intrinsic parameter for Classical Kruppas method

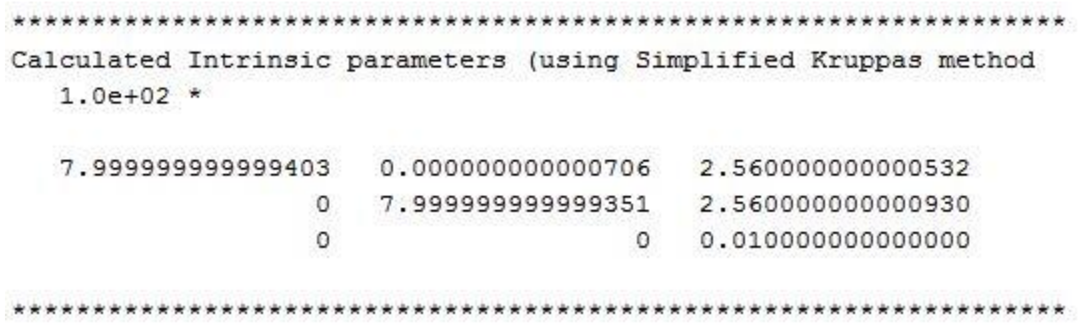
2.2. The Simplified Kruppas method

This method is done by SVD of fundamental matrix.

This is implemented by function,

error = costFunctionKSimplified(Fs, x)

The result for this method is as shown in Fig. 2



```
*****************************************************************
Calculated Intrinsic parameters (using Simplified Kruppas method
   1.0e+02 *

   7.999999999999403   0.000000000000706   2.560000000000532
                   0   7.999999999999351   2.560000000000930
                   0                   0   0.010000000000000

*****************************************************************
```

Fig. 2 Result of Intrinsic parameter for Simplified Kruppas method

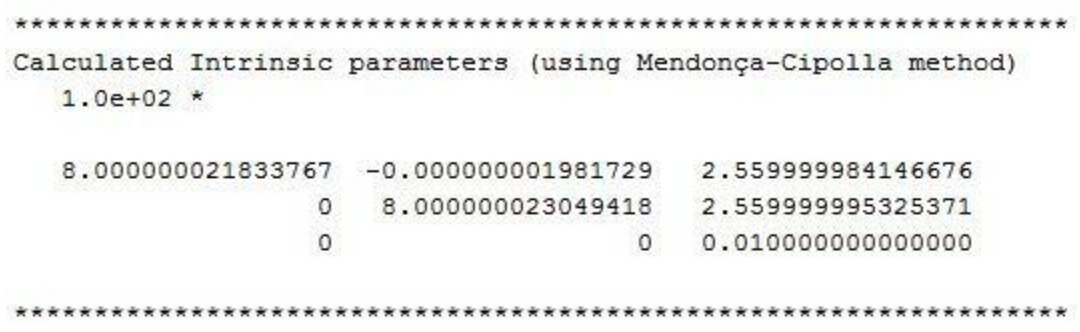
2.3 . The Mendonça-Cipolla auto-calibration method

This method exploits the property of essential matrix having two identical singular values (third is zero). the problem is solved by nonlinear least squares optimization of the cost function.

this function is implemented by,

error = costFunctionMenCipolla(Fs, x)

The result for this method is as given in Fig.3



```
****************************************************************
Calculated Intrinsic parameters (using Mendonça-Cipolla method)
   1.0e+02 *

   8.000000021833767  -0.000000001981729   2.559999984146676
                   0   8.000000023049418   2.559999995325371
                   0                   0   0.010000000000000

****************************************************************
```

Fig. 3 Result of Intrinsic parameter for Mendonça-Cipolla method

2.4 Using Dual Absolute Quadric

Here, the idea is to find the plane at infinity using Dual Absolute Quadric (DAQ) and initial guess of intrinsic parameters.

This is done solving the equation,

$$\mathsf{M}_j \mathcal{Q} \mathsf{M}_j^T \simeq \omega^{-1} \quad \text{where} \quad \mathcal{Q} = \begin{pmatrix} \omega^{-1} & \omega^{-1}\mathbf{n}_\Pi \\ \mathbf{n}_\Pi^T \omega^{-1} & \mathbf{n}_\Pi^T \omega^{-1} \mathbf{n}_\Pi \end{pmatrix}.$$

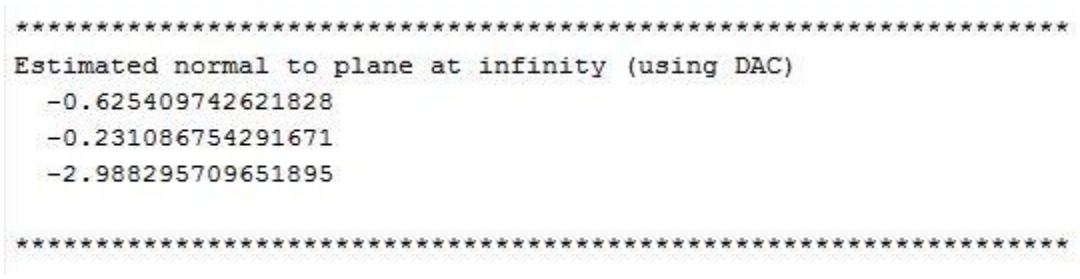
where, M is camera matrix, Q is DAQ, w = AA^T and 'n' is normal to plane at infinity.

So by solving this equations by some scale factor 'lambda' and initial guess we can find the plane at infinity and also 'lambda'.

This is implemented by,

normal = getNormalToPlaneAtInfinity(PPM, A)

The result is as shown in Fig.4



```
****************************************************************
Estimated normal to plane at infinity (using DAC)
  -0.625409742621828
  -0.231086754291671
  -2.988295709651895

****************************************************************
```

Fig. 4 Normal to plane at infinity using DAQ

After we find normal to plane at infinity, the idea is to find Homography at plane at infinity and to find 'w' from homography at plane at infinity and there by finding intrinsic parameters 'K' using Cholesky factorization[3].

The equation to find plane at infinity would be:

$$H_\infty = [e_{21}]_x F + e_{21} n^T$$

where, F is fundamental matrix, e_{21} is epipole and 'n' is normal to plane at inifnity.

After, we find plane at infinity, we can optimize for 'w' using equation given by,

$$H_\infty w H_\infty^T = w$$

I tried to optimize this function to find better value of intrinsic parameters, but my optimization is struck at minimum at initial guess and could not continue even if I vary the optimization settings like 'Function tolerance' , 'max iterations' etc.

3. Conclusion

These methods provide a powerful tool to find the intrinsic parameters of camera for auto calibration even if we don't have a pattern for calibration of camera as we generally have for calibration.

References

[1] Auto-calibration , Wikipedia

[2] Lecture notes of Dr. Adlane Habed.

[3] "Multiple View Geometry in Computer Vision", book by Richard Hartley and Andrew Zisserman