Non-central Catadioptric Camera Calibration Toolbox

Author:
Zhiyu Xiang, Xing Dai, Xiaojin Gong, Yanbing Zhou
Department of Information Science and Electronic Engineering
Zhejiang University (ZJU), China
Email:
xiangzy@zju.edu.cn

Last update: July 18, 2013

Index of the tutorial

1. Introduction of the Toolbox

The Non-central Catadioptric Camera Toolbox for matlab allows the user to calibrate not only any central omnidirectioanal camera (any panoramic camera having a single effective viewpoint) but also non-central camera due to the misalignment between the reflective mirror and the camera.

The Toolbox implements the procedure described in the paper which proposed a novel model for non-central catadioptric cameras and give out a method for calibration.

The Toolbox permits easy and practical calibration of the omnidirectional camera through two steps. At first, it requires the user to take a few pictures of a checkerboard shown at different position and orientations. Then, the user is asked to click manually on the corner points. After these two steps, calibration is completely automatically performed. After calibration, the Toolbox provides the relation between a given pixel point and 3D vector emanating from the single effective view point. This relation clearly depends on the mirror shape and on the intrinsic parameters of the camera.The novel aspects of the Non-central Catadioptric Camera Toolbox with aspect to other implementations are following:

- The Toolbox does not require a priori knowledge about the mirror shape.
- It does not require calibrating separately the perspective camera: the system camera and mirror is treated as a unique compact system.
- The calibration of the central or non-central systems can be treated with equal simplicity.

The calibration performed by the Non-central Catadioptric Camera Toolbox can lead to an equally simple calibration for both central and non-central systems. In fact, the Toolbox is able to provide an optimal solution even when the misalignment between the reflective and the camera is severe.

The calibration will estimate :

- the extrinsic parameters corresponding to the rotation (quaternion : Qw) and translation (Tw) between the grid and the mirror
- the parameters describing the mirror shape (xi)
- the distortion induced by the lens (eg. telecentric) (kc)
- the intrinsic parameters of the generalised camera : skew (alpha_c), generalised focal lengths (gamma1,gamma2) and principal points (cc)
- the vector (-ξ1, -ξ2, -ξ3): denote the position of the original projection center.

2. Software requirements

The Non-central Catadioptric Camera Toolbox for Matlab has been successfully tested under Matlab, version 2009b and 2010b for Windows.

The Calibration Refinement routine requires the Matlab Optimization Toolbox, in particular the function *lsqnonlin*, which you should have by default.

3. Download, install and run the Toolbox

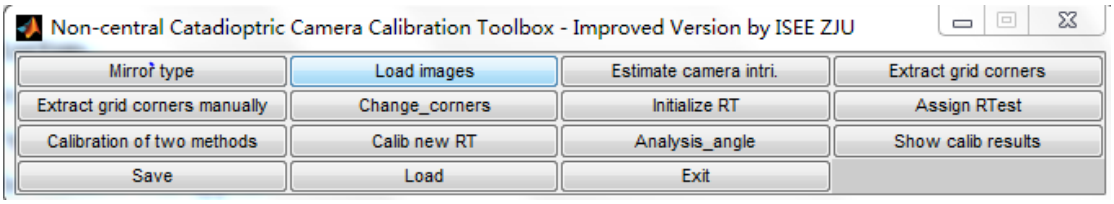
You can download the Non-central Catadioptric Camera Toolbox from here:
https://github.com/zju-isee-cv/new_cvUnzip the file, run Matlab (it's better to add the path of this Toolbox to matlab's set path), and type *ocam_calib_gui*

4. Startup

Please edit the 'SETTINGS.m' file first. Start the toolbox in Matlab:

>>omni_calib_gui

The window will appear:



Go to the directory containing the images. Generally the images are in the 'sample' folder'. In this case ,the images are stored in 'sample/sim_mis15'.

cd sample

cd sim_mis15

5. Mirror type

Click on the "Mirror type":

1 or [] : parabola (xi=1)

2 : catadioptric (hyperbola,ellipse,sphere)

3 : dioptric (fisheye)

Choice :

Camera type : catadioptric.

This step is only to constrain the minimization in the parabolic case (xi=1) and to avoid trying to extract the mirror border in the dioptric.

6. Loading images

"Load images" will ask you for the base name of the images in the current directory and their format:

>>

. **17.bmp** **7.bmp** **hyperbolic.pov**
18.bmp **8.bmp** **hyperbolic.pov.bak**
1.bmp **19.bmp** **9.bmp** **init.mat**
10.bmp **2.bmp** **Calib_Results.mat**

Basename camera calibration images (without number nor suffix):
Image format: ([]='r'='ras', 'b'='bmp', 't'='tif', 'p'='pgm', 'j'='jpg', 'm'='ppm')
b
Loading image 1...2...3...4...5...6...7...8...9...10...11...12...
Extraction of the mirror border from the images to estimate principal point and calculate the region of interest (roi).
Calculating min over all images... 1... 2... 3... 4... 5... 6... 7... 8... 9... 10... 11...

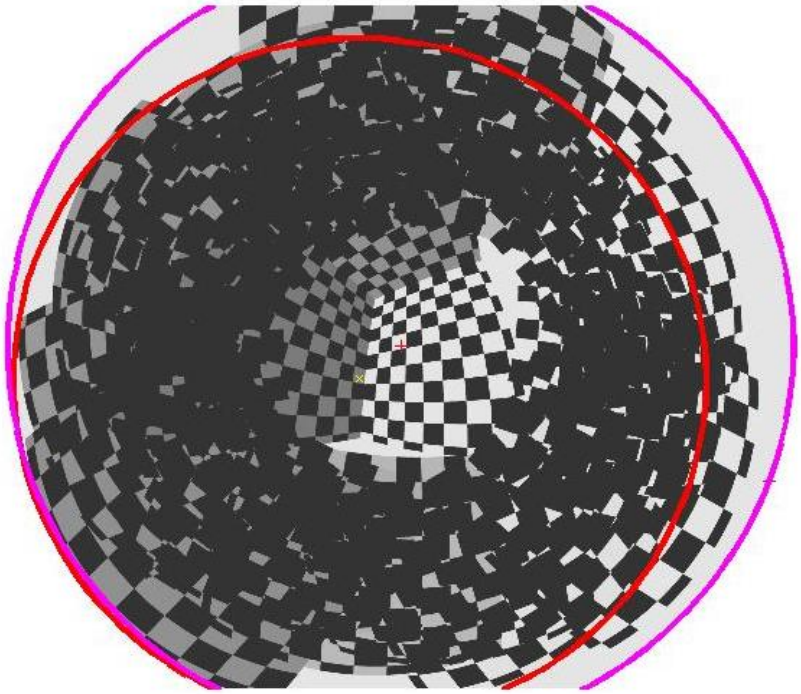
12... done.

The images are now loaded and we are ready to start estimating the intrinsic values and then calibrating the sensor.

7. Estimating the intrinsic parameters with the mirror border

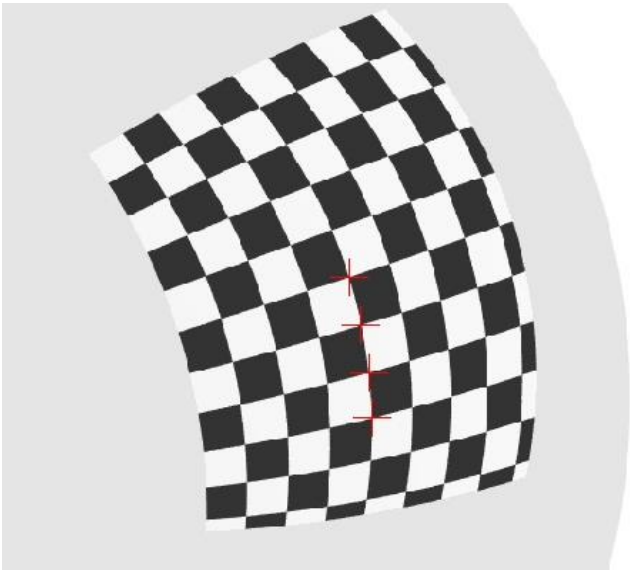
By pressing on "Estimate camera intri.", we will calculate an estimate of the intrinsic parameter of the underlying camera (generalized focal length and center). In the case of a catadioptric sensor, the user is asked to click on the center of the mirror:

Please click on the mirror center and then on the mirror inner border.
Calculating edges... done.
Rejecting inner points... done.
Doing a simplified RANSAC to obtain circle parameters... done.
Was the extraction successful ? ([]=yes, other=no) :



When then estimate the generalized focal length from points on a line image:

We are now going to estimate the generalised focal (gammac) from line images.
Which image shall we use ? ([] = 1) :
Please select at least 4 ALIGNED edge points on a NON-RADIAL line on the grid.
Click with the right button when finished.
done.



8. Extracting grid corners

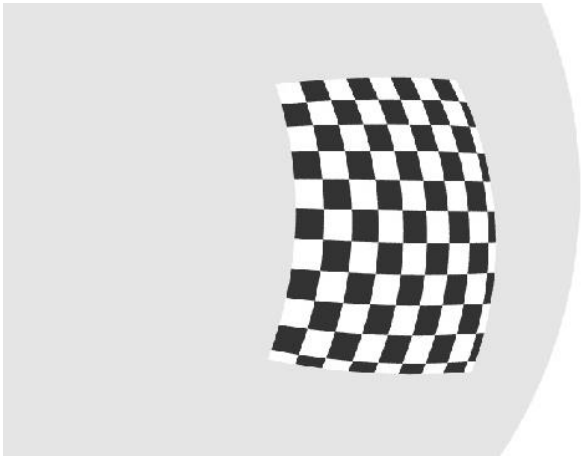
We are now ready to extract the grid corners which will be used during the minimization, press on **"Extract grid corners"**:

Extraction of the grid corners on the images (at each correct grid extraction, the values are save in "calib_data.mat")
Number(s) of image(s) to process ([] = all images) =
Window size for corner finder (wintx and winty):
wintx ([] = 5) =
winty ([] = 5) =

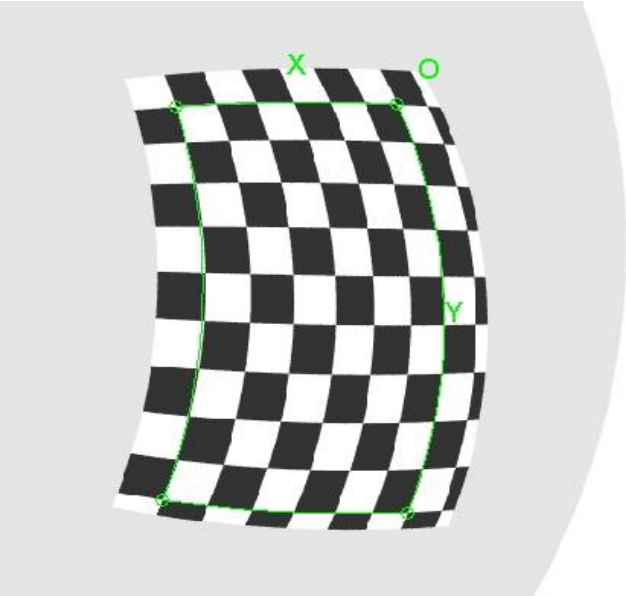
Sub-pixel extraction in the catadioptric case is less tolerant and we should try and keep the window size down.

Window size = 11x11
Processing image 1...
Using (wintx,winty)=(5,5) - Window size = 11x11 (Note: To reset the window size, run script clearwin)
Please press enter after zooming...

You can now zoom in on the calibration grid and press [Enter]:



You now need to click on the four corners of the grid in clockwise order:



The omni-lines should follow the grid.

You will then asked for the size of the grid:

Size of each square along the X direction: dX=50mm

Size of each square along the Y direction: dY=50mm (Note: To reset the size of the squares, clear the variables dX and dY)

Corner extraction...

Number of squares along the X direction ([]=9) = 6

Number of squares along the Y direction ([]=6) = 9

Size dX of each square along the X direction ([]=50mm) =

Size dY of each square along the Y direction ([]=50mm) =

Corner extraction... Was the extraction successful ? ([]=yes, other=no) :

done

Pixel error: err = [0.93266 1.05068]

This extraction is semiautomatic written by C.Mei. It works well under slight misalignment. But under severe misalignment, it work not so good. That why we need manual extraction as is showed in 9.

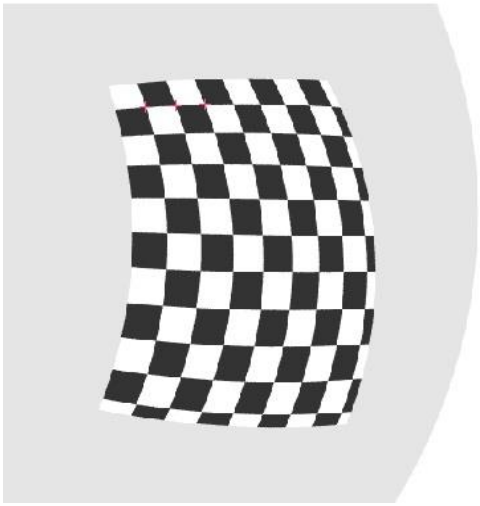
9. Extracting grid corners manually

Generally you can use the grid corners extracted automatically in **Step 8**.But when the misalignment between the reflective mirror and the camera is severe; there is much error in the extraction. So this function allows you to extract the corners manually. Press on '**Extracting grid corners manually**'.

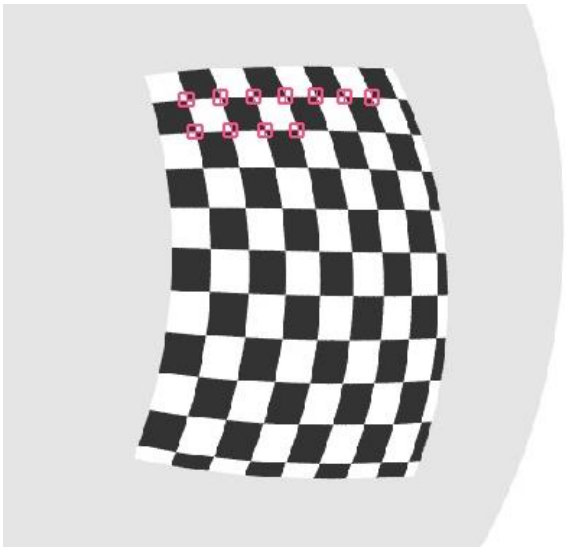
Extraction of the grid corners on the images (at each correct grid extraction, the values are save in "calib_data.mat")

Number(s) of image(s) to process ([] = all images) =

Processing image 1...
Using (wintx,winty)=(4,4) - Window size = 9x9 (Note: To reset the window size, run script clearwin)
Please press enter after zooming...



As the data of the grid corners are stored in**calib_data.mat**sequentially, we have to extract the grid corners in order. We can easily follow the signs lines by lines which are already marked in red.



You need to manually extract every grid corners of the images which you are asked to choose in **Step 8**. This task may be a little bit time-consuming and boring, so you should be more patient and careful.

Tips: You cannot avoid extracting a bad grid corners due to some reasons, so you need to extract the unsatisfactory image again in order to get the precise results. You can press on the '**Extracting grid corners manually**' after you extract all the images even if there are some bad images. Input the numbers of the images which are not extract well in the command window and then you can extract them again instead of extracting all the images again.

Personally it is better to move the cursor with one hand and press '**space**' key to confirm the

corner with another hand.

10. Change_corners

We now have two sets of coordinates of grid corners. By pressing on "**Change_corners**", we can use the grid corners which are extracted automatically or manually for further optimization. In order to verify the type of the grid corners, we set a variable "**manual**" in the "**gridInfo**" struct. It means the grid corners is extracted manually when "**manual**" is 1 and automatically when "**manual**" is 0.

11. InitializeRT

After extracting the corner grids, it is better to obtain an initial estimate for the extrinsic parameters to further stabilize the optimization. The initialization of R and T are both stored in '**paramEst.Qw_est{i}**', '**paramEst.Tw_est{i}**' and '**paramEst3D_Qw_est{i}**', '**paramEst.Tw_est{i}**'.

12. AssignRTest

Pressing on "**AssignRTest**" will assign the initialization of extrinsic parameters which are initialized in **Step 11** to '**paramEst.Qw**' and '**paramEst.Tw**' and we now have the initial values of the optimization.

13. Calibration of two methods

Pressing on "**Calibration of two methods**" will start the minimization with two models, one is C.mei's model and the other is our improved model.

new model with 3xi

Calibration results after optimization (with uncertainties):

Focal Length: gammac = [385.93411, 380.87821] ± [29.91760, 30.62780]
Principal point: cc = [642.08683, 422.08392] ± [3.873051, 9.79880]
Xi1: xi1 = [0.00461] ± [0.00780]
Xi2: xi2 = [0.45758] ± [0.04453]
Xi3: xi3 = [0.93381] ± [0.14447]
Skew: alpha_c = [0.00000] ± [0.00000] => angle of pixel axes = 90.00000 ± 0.00000 degrees
Distortion: kc = [0.00135 -0.00193 0.00000 0.00000 0.00000] ± [0.07142 0.00846 0.00000 0.00000 0.00000]
Pixel error+-std : err = [0.53691 0.33762]+-[1.29643 0.56943]
Note: The numerical errors are approximately three times the standard deviations (for reference).

optimization by cmei model

Main calibration optimization procedure - Number of images : 10
Gradient descent iterations : WARNING: removing singular matrix warning and

managing it internally.

1...2...3...4...(r) 5...6...7...8...(r) 9...10...11...12...(r) 13...14...15...16...(r)
17...18...19...20...(r) 21...22...23...24...(r) 25...26...27...28...(r)
29...30...31...32...(r) 33...34...35...36...(r) 37...38...39...40...(r)
41...42...43...44...(r)
45...46...47...48...49...50...51...52...53...54...55...56...57...58...59...60...done

Estimation of uncertainties...done

Cmei model Calibration results after optimization (with uncertainties):

Focal Length: gammac = [422.39178 463.26976] ± [14.14359 17.01055]

Principal point: cc = [642.99485, 568.67067] ± [3.47961, 6.34000]

Xi: xi = [1.14066] ± [0.05137]

Skew: alpha_c = [0.00000] ± [NaN] => angle of pixel axes = 90.00000 ± NaN degrees

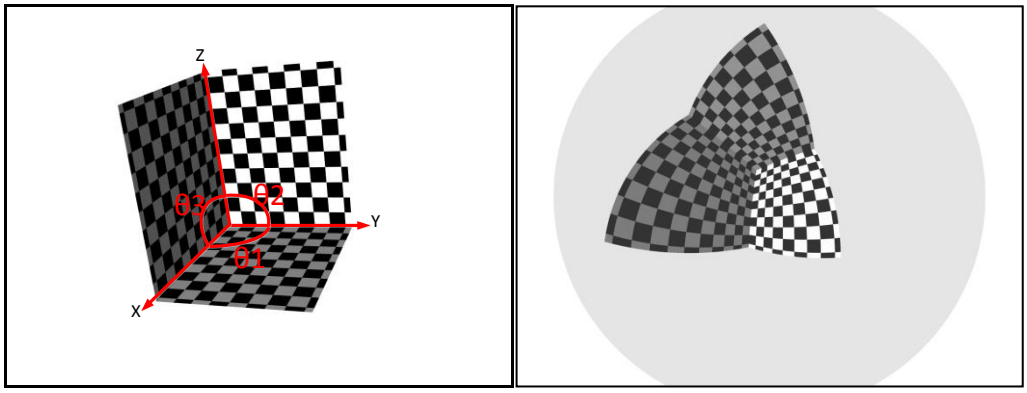
Distortion: kc = [0.13728 0.00686 0.13587 0.00003 0.00000] ± [0.04546 0.01543 0.01005 0.00231 NaN]

Pixel error+-std : err = [0.64934 0.45940]+-[1.33037 0.58411]

Note: The numerical errors are approximately three times the standard deviations (for reference).

14. Analysis_angle

To further verify the performance of our model quantitatively, we employed a trithedral object composed of three orthogonal planes and the angles between the planes are calculated from the calibrated extrinsic parameters **RwQw**.



Image(s) to analysis ([] = last three) :
paramEst angleQ8Q9: 92.147409 +- 0.699372
paramEst3D angleQ8Q9: 91.441807 +- 0.115319

paramEst angleQ9Q10: 91.211297 +- 0.761821
paramEst3D angleQ9Q10: 89.998210 +- 0.200266

paramEst angleQ10Q8: 87.615479 +- 0.562863
paramEst3D angleQ10Q8: 89.023303 +- 0.065071

paramEst_err = 1.914409 +-0.674685
paramEst3D_err = 0.806765 +-0.126885

15. Other buttons

- **"Calib new RT"** computes the extrinsic parameters in the case of known intrinsic parameters.
- **"Show calib results"** show the calibration values and errors
- **"Save"** saves the extrinsic and intrinsic parameters, the grid points, ...
- **"Load"** loads the values obtained during the calibration
- **"Exit"** quits the program

16. References

1. Zhiyu Xiang, Xing Dai and Xiaojin Gong, Noncentral catadioptric camera calibration using a generalized unified model, Optics Letters, Vol.38 (9), 1367-1369, 2013
2. Geyer, Christopher, and Kostas Daniilidis. "A unifying theory for central panoramic systems and practical implications." Computer Vision—ECCV 2000. Springer Berlin Heidelberg, 2000. 445-461.
3. Mei, Christopher, and Patrick Rives. "Single view point omnidirectional camera calibration from planar grids." Robotics and Automation, 2007 IEEE International Conference on. IEEE, 2007.
4. Grossberg, Michael D., and Shree K. Nayar. "The raxel imaging model and ray-based calibration." International Journal of Computer Vision 61.2 (2005): 119-137.
5. Gonçalves, Nuno, and Helder Araújo. "Estimating parameters of non-central catadioptric systems using bundle adjustment." Computer Vision and Image Understanding 113.1 (2009): 11-28.
6. Xiang, Zhiyu, Bo Sun, and Xing Dai. "The camera itself as a calibration pattern: A novel self-calibration method for non-central catadioptric cameras." Sensors 12.6 (2012): 7299-7317.
7. OmnidirectionalCalibration Toolbox,http://homepages.laas.fr/~cmei/index.php/Toolbox

17. Projection Model

We proposed a novel model for non-central catadioptric cameras and give out a method for calibration. The model is called generalized unified model (GUM). We mainly focus on the non-central system caused by the misalignment. The traditional unified model proposed by C.Geyer [1] requires the image plane to be perpendicular with the diameter, a condition which a central camera always meets. However when misalignment happens, an extra rotation and translation deviating from the ideal configuration appears between the camera and the mirror. By realigning the mirror frame and pointing its Zm axis to the normal of the oblique image

plane, the generalized unified model is obtained, as shown in Fig. 1. In our new model the constraint that the image plane has to be perpendicular with the diameter is released and the original projection center and the original projection center C_p,is replaced byC_p.$C_p$can be moved freely within the unit sphere and its position is denoted by a 3-vector $(-\xi1, -\xi2, -\xi3)$. Comparing with the original unified model in [1], our generalized model features only 2 extra parameters, i.e. $-\xi1$ and $-\xi2$ which describe the position of the projection center. This compact description will facilitate the calibration work greatly.

Fig 1 is the new model we proposed. There are two coordinates system: F_p(origin is visual optical center $\mathbf{C_p}$) and F_m (origin is center of sphere $\mathbf{C_m}$). The coordinate of $\mathbf{C_p}$ is $(-\xi1, -\xi2, -\xi3)$in F_m . The projection of 3D points can be done in the following steps.

1. The world points $\mathbf{X_w}(x_w, y_w, z_w)$ are first transformed into a mirror frame $F_m(C_m\text{-}X_m Y_m Z_m)$ by a standard rigid body transformation, as shown in equation:

$$(\mathbf{X_w})_{F_m} = \mathbf{R_m}(\mathbf{X_w})_w + \mathbf{T_m}$$

   Where $\mathbf{R_m}$ and $\mathbf{T_m}$are rotation and translation between the world coordinate and the mirror coordinate respectively.

2. $(\mathbf{X_m})_{F_m}$ is projected to point $(\mathbf{X_s})_{F_m}$ on the unit sphere with

$$(\mathbf{X_s})_{F_m} = \frac{(\mathbf{X_m})_{F_m}}{|(\mathbf{X}_\mathbf{m})_{F_m}|} = (x_s, y_s, z_s)$$

3. The points $(\mathbf{X_s})_{F_m}$ are then changed to a new reference frame F_p centered at $\mathbf{C_p} = (-\xi1, -\xi2, -\xi3)$ with coordinates

$$(\mathbf{X_s})_{F_p} = (x_s + \xi1, y_s + \xi2, z_s + \xi3)$$

4. The following step is a projection of these points onto the normalized plane Π_{m_u} with

$$\mathbf{m_u} = \left(\frac{x_s + \xi1}{z_s + \xi3}, \frac{y_s + \xi2}{z_s + \xi3}, 1\right) = h((\mathbf{X_s})_{F_m})$$

   Finally, with a generalized perspective projection matrix$\mathbf{K}$, $\mathbf{m_u}$is projective onto the image plane Π_p by

$$\mathbf{p} = \mathbf{K}\mathbf{m_u} = \begin{bmatrix} f_1\eta & f_1\eta\alpha & u_0 \\ 0 & f_2\eta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{m_u} = k(\mathbf{m_u})$$

   Where $\gamma_1 = f_1\eta$, $\gamma_2 = f_2\eta$ are focal length of the generalized perspective projection with f_1, f_2 the focal length of the real perspective camera and η a ratio depends on the mirror shape respectively. The mirror values are detailed in Table1.

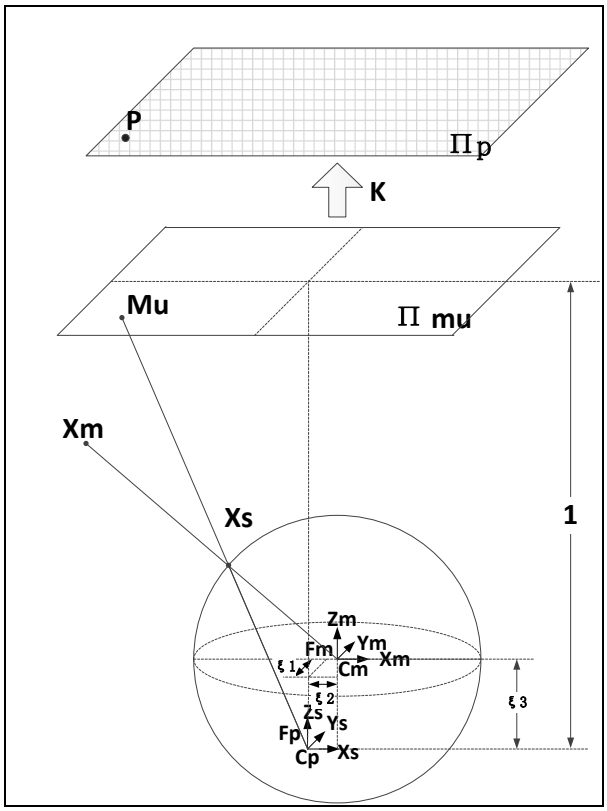


Fig.1.The proposed generalized unified model

Considering the distortion that real camera exists, the radical distortion should be compensated by using:

$$L(\rho) = 1 + k_1\rho^2 + k_2\rho^4$$

Where $\rho = \sqrt{(x^2 + y^2)}$. Unlike in [2], tangential distortion is no longer needed in our model since the effect has been well represented by $\xi 1$and $\xi 2$.

	ξ	η
Parabola	1	$-2p$
Hyperbola	$\frac{d}{\sqrt{d^2 + 4p^2}}$	$\frac{-2p}{\sqrt{d^2 + 4p^2}}$
Ellipse	$\frac{d}{\sqrt{d^2 + 4p^2}}$	$\frac{2p}{\sqrt{d^2 + 4p^2}}$
Planar	0	-1
d:distance between focal points 4p:latus rectum		