



e-ISSN: 2319-8753 | p-ISSN: 2347-6710

IJRSET

International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 11, Issue 5, May 2022

ISSN INTERNATIONAL
STANDARD
SERIAL
NUMBER
INDIA

Impact Factor: 8.118

 9940 572 462

 6381 907 438

 ijrset@gmail.com

 www.ijrset.com

Case Study on Calibration of own camera iPhone 12

Debabrata Paul¹, Piyush Pritam²

Student, Department of Civil Engineering, Alma Mater Studiorum – University of Bologna, viale del Risorgimento 2,
Bologna, Italy¹

Student, Department of Civil Engineering, Alma Mater Studiorum – University of Bologna, viale del Risorgimento 2,
Bologna, Italy²

ABSTRACT: Goal of this case study is to calibrate your own camera, like your own digital single lens reflect (SLR) camera, or the one in your mobile phone. Calibration have been done using a chessboard pattern to estimate its interior parameters. The interior orientation is preferably estimated before you use your camera on location. This case study involves the fundamental principle of photogrammetry: the collinear equations. In this case study, OpenCV in MATLAB is utilized to apply the camera calibration.

KEYWORDS: camera calibration, mobile phone, photogrammetry, collinear equations, OpenCV, MATLAB.

I. INTRODUCTION

Interior parameters of a camera¹:

The list of internal parameters is as per the seriatim, -

- **Focal length:** The focal length in pixels is stored in the 2x1 vector f_c .
- **Principal point:** The principal point coordinates are stored in the 2x1 vector $cc.(x_0, y_0)$: image coordinates of the principal point (i.e., image centre), where x and y denotethe column and row direction, respectively.
- **Skew coefficient:** The skew coefficient defining the angle between the x and y pixel axes is stored in the scalar α_c .
- **Distortions:** The image distortion coefficients (radial and tangential distortions) are stored in the 5x1 vector kc .
- **f_x and f_y :** focal length of a camera along x and y directions.

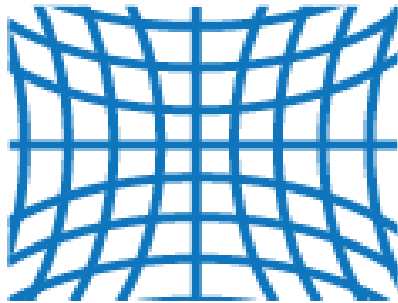
The intrinsic parameters include the focal length, the optical center, also known as the principal point, and the skew coefficient. The camera intrinsic matrix, K , is defined as:

$$\begin{pmatrix} f_x & 0 & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{pmatrix}$$

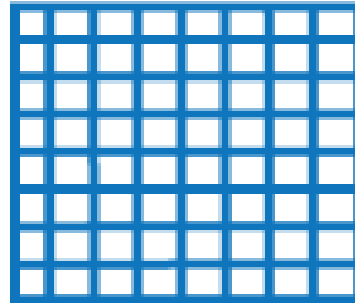
Camera's distortion parameters^{1,2}:

The first, and most common type of camera lens distortion is called **radial distortion**.

There are two types of this distortion, the **positive or barrel radial distortion**, and the **negative or pincushion radial distortion**.



Pincushion distortion
Positive radial displacement



No distortion



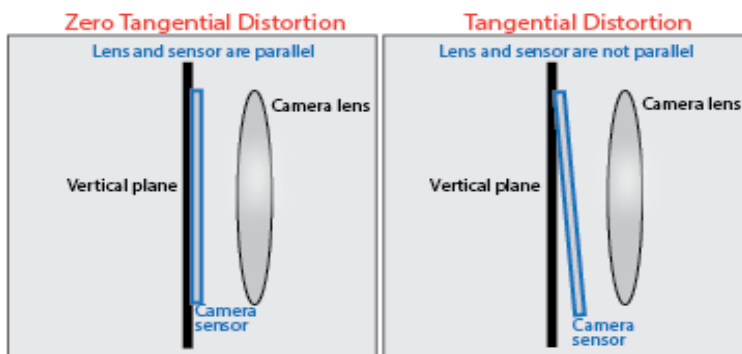
Barrel distortion
Negative radial displacement

Radial distortion can be represented as follows:

$$x_{\text{distorted}} = x(1 + k_1r^2 + k_2r^4 + k_3r^6)$$

$$y_{\text{distorted}} = y(1 + k_1r^2 + k_2r^4 + k_3r^6)$$

Similarly, **tangential distortion** occurs because the image-taking lens is not aligned perfectly parallel to the imaging plane. So, some areas in the image may look nearer than expected.



The amount of **tangential distortion** can be represented as below:

$$x_{\text{distorted}} = x + [2p_1xy + p_2(r^2 + 2x^2)]$$

$$y_{\text{distorted}} = y + [p_1(r^2 + 2y^2) + 2p_2xy]$$

In short, we need to find **five parameters**, known as **distortion coefficients**, given by:

$$\text{Distortion coefficients} = (k_1 \quad k_2 \quad p_1 \quad p_2 \quad k_3)$$

II. RELATED WORK

Chessboards are widely used for camera calibration³:

Chessboards are often used during camera calibration because they are simple to construct, and their planar grid structure defines many natural interest points in an image.

They provide features that can be computed as intersection of lines which are:

- **robust:** if a part of the calibration pattern is occluded, you can still get the coordinates of the intersection points thanks to the line equation.
- **accurate:** line equations can be interpolated from sample points, and this is the case for their intersection too. Thus, you naturally get subpixel precision. This also holds for the old-fashioned pipelines where the corners were clicked by somebody.

III. METHODOLOGY

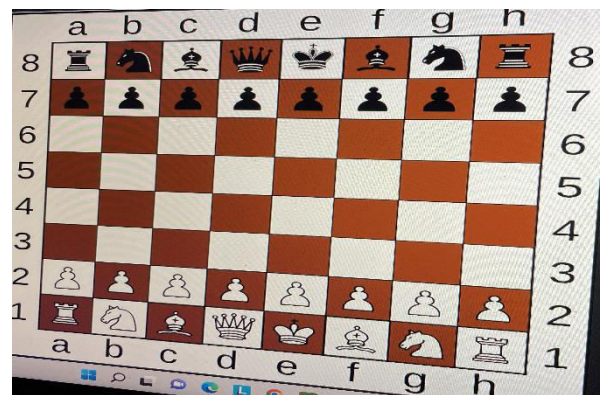
Description of my own camera including its lens including two captured images of a chessboard^{4,5}:

The camera used in this study is the camera from iPhone 12, which has a 12-megapixel rear-facing iSight camera with a 1.4 μm pixel size. The focal length of such iSight camera is 26 mm and the sensor area is 23.9 mm².

The lens of the camera should be fixed, while taking pictures of the chessboard. At that point two images of a chessboard (downloaded from https://upload.wikimedia.org/wikipedia/commons/2/2c/AAA_SVG_Chessboard_and_chess_pieces_02.svg and from this took photos with the iPhone 12) from different viewing angles can be obtained, as shown in Figure (i).



(A) Left image



(B) Right image

Figure (i): Images of a chessboard from different angles

Chessboard designs are distinct and simple to distinguish in an image. Not only that, the corners of squares on the chessboard are perfect for localizing them since they have sharp angles in two directions and these corners are too related by the truth that they are at the intersection of chessboard lines. That's why chessboard should cover the full extent of the images for calibrating precisely. The essential idea is that the distortion is always extraordinary at the edges of a picture. In case there's no chessboard representation at the edges, the distortion arrangement based on the central portion of the picture isn't accurate enough to correct the edges.

Application of the corner detection algorithm on both images and displaying of the found corners^{6,7,8,9}:

Image coordinates of corners on the chessboard can be obtained by a bunch of corner detection algorithms. In MATLAB, the function *detectHarrisFeatures* can help to detect these corners, which can be implemented as per the following MATLAB algorithm:

For Left image:

```
clc
clear all
close all
warning off
a=imread('leftimage.png');
imshow(a);
b=rgb2gray(a);
figure;
imshow(b);
points=detectHarrisFeatures(b);
hold on;
plot(points);
```

For Right image:

```
clc
clear all
close all
warning off
a=imread('rightimage.png');
imshow(a);
b=rgb2gray(a);
figure;
imshow(b);
points=detectHarrisFeatures(b);
hold on;
plot(points);
```

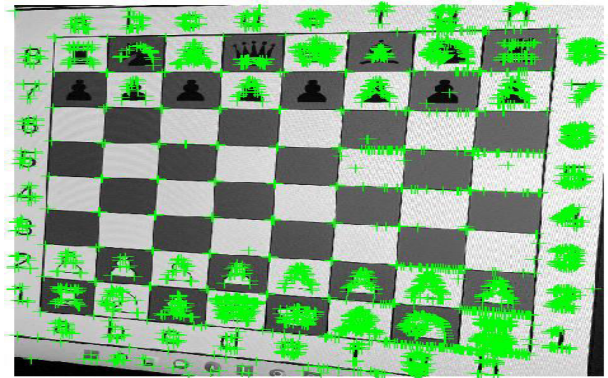
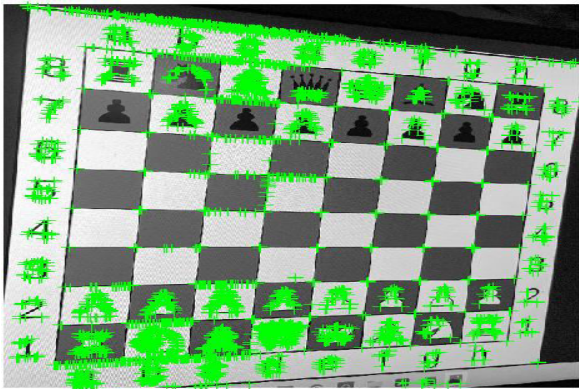


Figure (ii) shows the corner detection results in both images taken for the calibration.

(A) Left image

(B) Right image

Figure (ii): Results of corner detection

Creation of an array or vector to store the 3D coordinates in a world coordinate system²:

To find the projection of a 3D point onto the image plane, we first need to transform the point from **world coordinate system** to the **camera coordinate system** using the **extrinsic parameters** (Rotation **R** and Translation **t**).

Next, using the intrinsic parameters of the camera, we project the point onto the image plane.

The equations that relate 3D point (X_w, Y_w, Z_w) in world coordinates to its projection (u, v) in the image coordinates are shown below, -



$$\begin{bmatrix} u' \\ v' \\ z' \end{bmatrix} = \mathbf{P} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

$$u = \frac{u'}{w'}$$

$$v = \frac{v'}{w'}$$

Where, \mathbf{P} is a 3×4 Projection matrix consisting of two parts — the **intrinsic matrix** (K) that contains the intrinsic parameters and the **extrinsic matrix** ($[\mathbf{R} \mid \mathbf{t}]$) that is combination of 3×3 rotation matrix \mathbf{R} and a 3×1 translation \mathbf{t} vector.

$$\mathbf{P} = \overbrace{\mathbf{K}}^{\text{Intrinsic Matrix}} \times \overbrace{[\mathbf{R} \mid \mathbf{t}]}^{\text{Extrinsic Matrix}}$$

Here, the intrinsic matrix K is upper triangular

$$\mathbf{K} = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

where,

f_x, f_y are the x and y focal lengths (yes, they are usually the same).

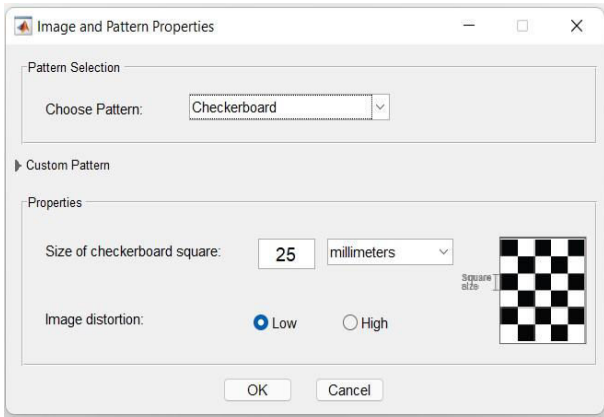
c_x, c_y are the x and y coordinates of the optical centre in the image plane. Using the centre of the image is usually a good enough approximation.

γ is the skew between the axes. It is usually 0.

IV. EXPERIMENTAL RESULTS

Application of the camera calibration algorithm and explanation of the outputs ^{6,7,8,9}:

In MATLAB, camera calibration can be applied through the MATLAB app function *Camera Calibrator*, and the same is implemented as per the serialtim, -



cameraParams.IntrinsicMatrix

| | 1 | 2 | 3 |
|---|------------|------------|---|
| 1 | 3.6835e+03 | 0 | 0 |
| 2 | 0 | 3.6012e+03 | 0 |
| 3 | 1.9288e+03 | 1.4349e+03 | 1 |

cameraParams =

cameraParameters with properties:

Camera Intrinsic

Intrinsic: [1x1 cameraIntrinsic]

Camera Extrinsic

RotationMatrices: [3x3x2 double]

TranslationVectors: [2x3 double]

Accuracy of Estimation

MeanReprojectionError: 1.0114

ReprojectionErrors: [42x2x2 double]

ReprojectedPoints: [42x2x2 double]

Calibration Settings

NumPatterns: 2

DetectedKeypoints: [42x2 logical]

WorldPoints: [42x2 double]

WorldUnits: 'millimeters'

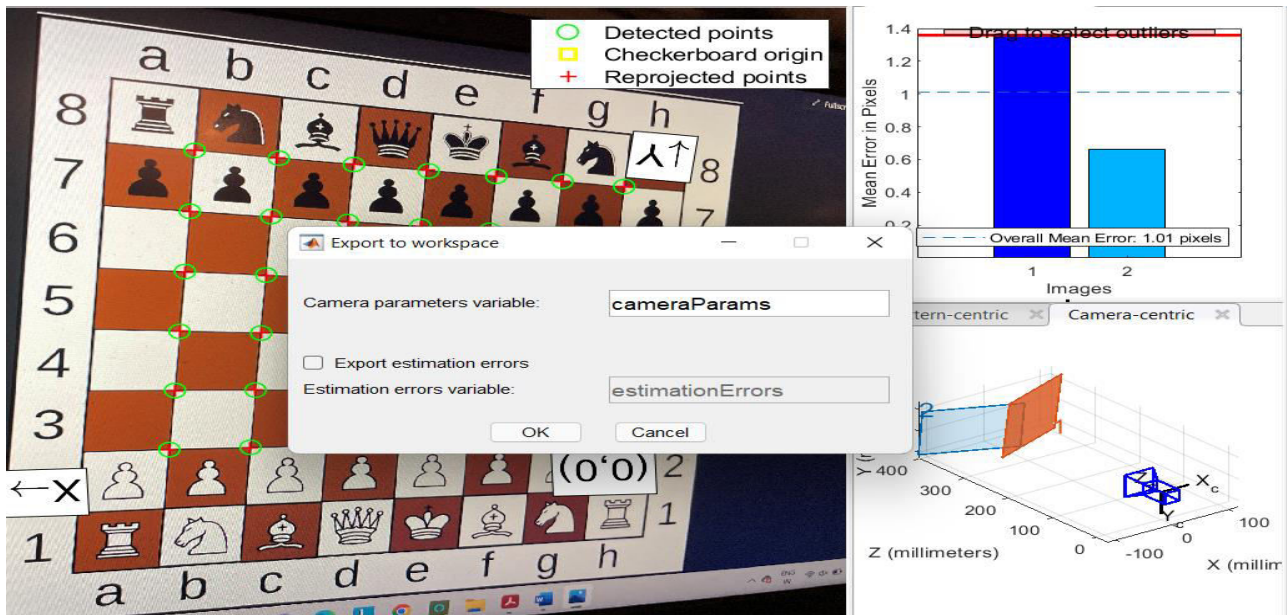
EstimateSkew: 0

NumRadialDistortionCoefficients: 2

EstimateTangentialDistortion: 0



| Property | Value |
|---------------------------------|-----------------------|
| ImageSize | [2872,3830] |
| RadialDistortion | [0.2076,-0.7030] |
| TangentialDistortion | [0,0] |
| WorldPoints | 42x2 double |
| WorldUnits | 'millimeters' |
| EstimateSkew | 0 |
| NumRadialDistortionCoefficients | 2 |
| EstimateTangentialDistortion | 0 |
| TranslationVectors | [110.6625,63.876...] |
| ReprojectionErrors | 42x2x2 double |
| DetectedKeyPoints | 42x2 logical |
| RotationVectors | [0.6146,0.0940,-3...] |
| NumPatterns | 2 |
| Intrinsics | 1x1 cameraIntrin... |
| IntrinsicMatrix | [3.6835e+03,0,0;...] |
| FocalLength | [3.6835e+03,3.60...] |
| PrincipalPoint | [1.9288e+03,1.43...] |
| Skew | 0 |
| MeanReprojectionError | 1.0114 |
| ReprojectedPoints | 42x2x2 double |
| RotationMatrices | 3x3x2 double |



Detection Results

| | |
|------------------------------|---|
| Total images processed: | 2 |
| Added images: | 2 |
| Partially detected patterns: | 1 |
| Rejected images: | 0 |

OK



MATLAB SCRIPT:

```
% Auto-generated by cameraCalibrator app on 15-Apr-2022
%-----

% Define images to process
imageFileNames = {'C:\Users\DEBABRATA PAUL\OneDrive\Desktop\Erasmus application at TU Delft from
UNIBO\Final\Revised\CIE4614, 3D Surveying\Small Assignment 2\Pic\LI.png',...
'C:\Users\DEBABRATA PAUL\OneDrive\Desktop\Erasmus application at TU Delft from
UNIBO\Final\Revised\CIE4614, 3D Surveying\Small Assignment 2\Pic\RI.png',...
};
% Detect calibration pattern in images
detector = vision.calibration.monocular.CheckerboardDetector();
[imagePoints, imagesUsed] = detectPatternPoints(detector, imageFileNames);
imageFileNames = imageFileNames(imagesUsed);

% Read the first image to obtain image size
originalImage = imread(imageFileNames{1});
[mrows, ncols, ~] = size(originalImage);
% Generate world coordinates for the planar pattern keypoints
squareSize = 25; % in units of 'millimeters'
worldPoints = generateWorldPoints(detector, 'SquareSize', squareSize);

% Calibrate the camera
[cameraParams, imagesUsed, estimationErrors] = estimateCameraParameters(imagePoints, worldPoints, ...
'EstimateSkew', false, 'EstimateTangentialDistortion', false, ...
'NumRadialDistortionCoefficients', 2, 'WorldUnits', 'millimeters', ...
'InitialIntrinsicMatrix', [], 'InitialRadialDistortion', [], ...
'ImageSize', [mrows, ncols]);

% View reprojection errors
h1=figure; showReprojectionErrors(cameraParams);

% Visualize pattern locations
h2=figure; showExtrinsics(cameraParams, 'CameraCentric');

% Display parameter estimation errors
displayErrors(estimationErrors, cameraParams);

% For example, you can use the calibration data to remove effects of lens distortion.
undistortedImage = undistortImage(originalImage, cameraParams);

% See additional examples of how to use the calibration data. At the prompt type:
% showdemo('MeasuringPlanarObjectsExample')
% showdemo('StructureFromMotionExample')
```

Using of distortion parameters for rectifying the images and displaying of the results ^{6,7,8,9}:

The MATLAB *fishEyeParameter* is used to rectify the images using the calibration results and the rectified images are shown in **Figure (iii)**.

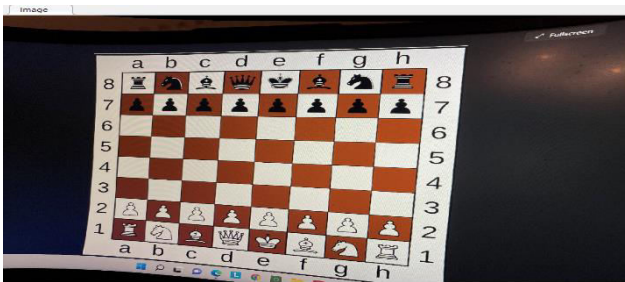


(A) Left image



(B) Right image

Figure (iii): Images after rectification



(A) Left image



(B) Right image

Figure (iv): Images after rectification (cropped)

MATLAB SCRIPT:

% Auto-generated by cameraCalibrator app on 15-Apr-2022

%-----

% Define images to process

imageFileNames = {'C:\Users\DEBABRATA PAUL\OneDrive\Desktop\Erasmus application at TU Delft from UNIBO\Final\Revised\CIE4614, 3D Surveying\Small Assignment 2\Pic\LI.png',...

'C:\Users\DEBABRATA PAUL\OneDrive\Desktop\Erasmus application at TU Delft from UNIBO\Final\Revised\CIE4614, 3D Surveying\Small Assignment 2\Pic\RI.png',...

};

% Detect calibration pattern in images

detector = vision.calibration.monocular.CheckerboardDetector();

[imagePoints, imagesUsed] = detectPatternPoints(detector, imageFileNames);

imageFileNames = imageFileNames(imagesUsed);

% Read the first image to obtain image size

originalImage = imread(imageFileNames{1});

[mrows, ncols, ~] = size(originalImage);

% Generate world coordinates for the planar pattern keypoints

squareSize = 25; % in units of 'millimeters'

worldPoints = generateWorldPoints(detector, 'SquareSize', squareSize);

% Calibrate the camera using fisheye parameters

[cameraParams, imagesUsed, estimationErrors] = estimateFisheyeParameters(imagePoints, worldPoints, ...

[mrows, ncols], ...



```
'EstimateAlignment', true, ...
'WorldUnits', 'millimeters');
```

```
% View reprojection errors
h1=figure; showReprojectionErrors(cameraParams);
% Visualize pattern locations
h2=figure; showExtrinsics(cameraParams, 'CameraCentric');
% Display parameter estimation errors
displayErrors(estimationErrors, cameraParams);
% For example, you can use the calibration data to remove effects of lens distortion.
undistortedImage = undistortFisheyeImage(originalImage, cameraParams.Intrinsics);
% See additional examples of how to use the calibration data. At the prompt type:
% showdemo('MeasuringPlanarObjectsExample')
% showdemo('StructureFromMotionExample')
```

V. RECOMMENDATION AND CONCLUSION

Using of the output parameters to obtain for one 3D point its corresponding image position by using the collinearity equations or an equivalent matrix multiplication in a suitable projection ^{5, 6, 7, 8, 9}:

It is vital to discover a projection relationship utilizing insides and outside parameters for anticipating a 3D point on the picture, as well as distortion coefficients of the camera. In photogrammetry, such relationship is depicted as the collinearity equations. We will utilize the increase of homogeneous matrices to represent the collinearity equations in a briefer way.

we got to apply the distortion parameters to realize the exact image coordinates and in MATLAB, the *fishEyeParameter* can help to achieve the projection, with the 3D object coordinates, rotation vector, translation vector, camera matrix and distortion parameters of an image as inputs and are explained as below, -

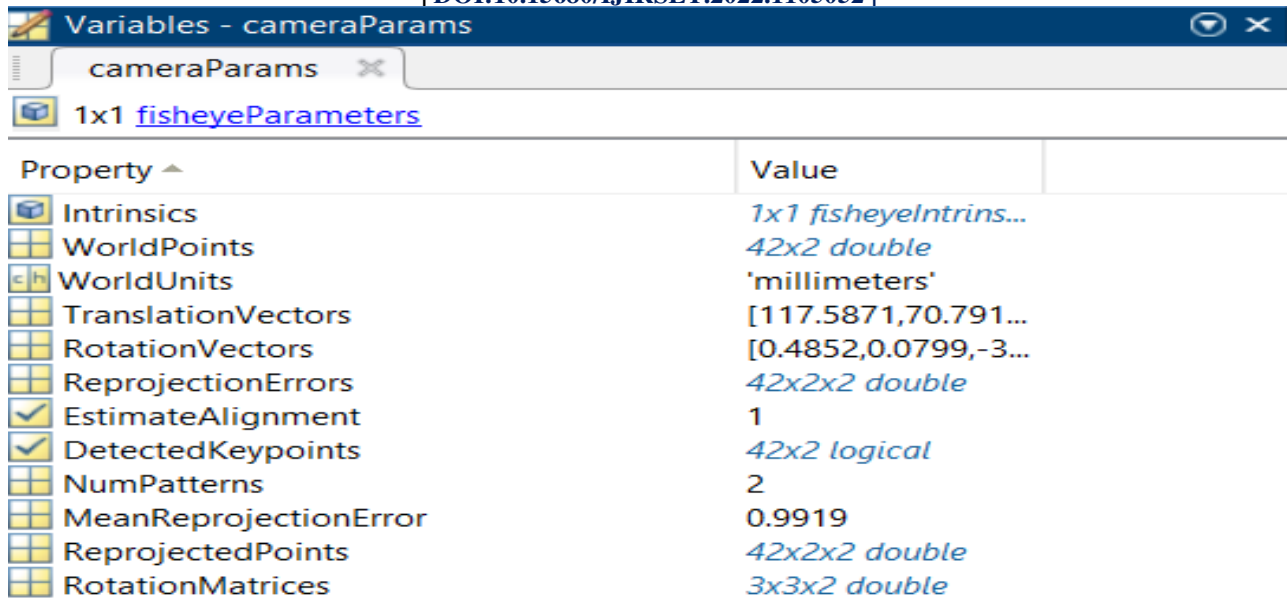
The image shows a MATLAB interface with three windows:

- Variables - cameraParams.TranslationVectors:** A table showing translation vectors for two camera centers.
- Variables - cameraParams.RotationMatrices:** Two matrices representing rotation for each camera center.
- Command Window:** A detailed display of the `cameraParams` structure, including intrinsic and extrinsic parameters, accuracy of estimation, and calibration settings.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|-----------|----------|----------|---|---|---|---|---|
| 1 | 117.5871 | 70.7916 | 302.2659 | | | | | |
| 2 | -109.2103 | -39.4266 | 313.1646 | | | | | |

```

cameraParams =
  fisheyeParameters with properties:
    Camera Intrinsics
      Intrinsics: [1x1 fisheyeIntrinsics]
    Camera Extrinsics
      RotationMatrices: [3x3x2 double]
      TranslationVectors: [2x3 double]
    Accuracy of Estimation
      MeanReprojectionError: 0.9919
      ReprojectionErrors: [42x2x2 double]
      ReprojectedPoints: [42x2x2 double]
    Calibration Settings
      NumPatterns: 2
      DetectedKeypoints: [42x2 logical]
      WorldPoints: [42x2 double]
      WorldUnits: 'millimeters'
      EstimateAlignment: 1
    
```



| Property ^ | Value |
|-----------------------|-----------------------|
| Intrinsics | 1x1 fisheyeIntrins... |
| WorldPoints | 42x2 double |
| WorldUnits | 'millimeters' |
| TranslationVectors | [117.5871,70.791... |
| RotationVectors | [0.4852,0.0799,-3... |
| ReprojectionErrors | 42x2x2 double |
| EstimateAlignment | 1 |
| DetectedKeypoints | 42x2 logical |
| NumPatterns | 2 |
| MeanReprojectionError | 0.9919 |
| ReprojectedPoints | 42x2x2 double |
| RotationMatrices | 3x3x2 double |

ACKNOWLEDGEMENTS

The authors, acknowledge the research and study enabling environment created by the Faculty of Civil Engineering and Geosciences (CEG), TU Delft.

We would like to give special thanks to **Prof. RoderikLindenbergh**, Teaching assistant **MiekeKuschnerus** and other group members of Optical and Laser Remote Sensing (TU Delft) to help us find the way of doing the study.

ETHICS DECLARATIONS

CONFLICT OF INTEREST

Both the authors state that there is no conflict of interest.

REFERENCES

- [1]<https://in.mathworks.com/help/vision/ug/cameracalibration.html#:~:text=The%20intrinsic%20parameters%20represent%20the,plane%20using%20the%20intrinsics%20parameters.>
- [2] <https://learnopencv.com/camera-calibration-using-opencv/>
- [3] <https://dsp.stackexchange.com/questions/24734/camera-calibration-why-chessboards.>
- [4] <https://www.apple.com/nl/iphone-12/specs/>
- [5] Qian Bai. "Assignment 2 - Camera Calibration", CIE4614 3D Surveying.
- [6] Zhang, Z. "A Flexible New Technique for Camera Calibration." *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Vol. 22, No. 11, 2000, pp. 1330–1334.
- [7]Heikkila, J., and O. Silven. "A Four-step Camera Calibration Procedure with Implicit Image Correction." *IEEE International Conference on Computer Vision and Pattern Recognition*.1997.
- [8]Bouguet, J. Y. "Camera Calibration Toolbox for Matlab." Computational Vision at the California Institute of Technology. Camera Calibration Toolbox for MATLAB.
- [9]Bradski, G., and A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. Sebastopol, CA: O'Reilly, 2008.



INNO SPACE
SJIF Scientific Journal Impact Factor
Impact Factor: 8.118



ISSN INTERNATIONAL
STANDARD
SERIAL
NUMBER
INDIA



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

 **9940 572 462**  **6381 907 438**  **ijirset@gmail.com**



www.ijirset.com

Scan to save the contact details