Object for storing camera parameters

Syntax

```
cameraParams = cameraParameters
cameraParams = cameraParameters(Name,Value)
```

Description

`cameraParams = cameraParameters` returns an object that contains the intrinsic, extrinsic, and lens distortion parameters of a camera.

`cameraParams = cameraParameters(Name,Value)` configures the camera parameters object properties, specified as one or more `Name,Value` pair arguments. Unspecified properties use default values.

Input Arguments

collapse all

The object contains intrinsic, extrinsic, lens distortion, and estimation properties.

Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name,Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside single quotes (`' '`). You can specify several name and value pair arguments in any order as `Name1,Value1,...,NameN,ValueN`.

Example: `'RadialDistortion',[0 0 0]` sets the `'RadialDistortion'` to `[0 0 0]`.

`'IntrinsicMatrix'` — Projection matrix

3-by-3 identity matrix

Projection matrix, specified as the comma-separated pair consisting of `'IntrinsicMatrix'` and a 3-by-3 identity matrix. For the matrix format, the object uses the following format:

$$\begin{bmatrix} f_x & 0 & 0 \\ s & f_y & 0 \\ c_x & c_y & 1 \end{bmatrix}$$

The coordinates $[c_x\ c_y]$ represent the optical center (the principal point), in pixels. When the x and y axis are exactly perpendicular, the skew parameter, s, equals `0`.

$f_x = F * s_x$
$f_y = F * s_y$

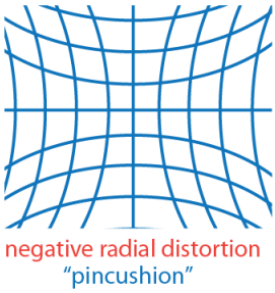
F, is the focal length in world units, typically expressed in millimeters.
$[s_x, s_y]$ are the number of pixels per world unit in the x and y direction respectively.
f_x and f_y are expressed in pixels.

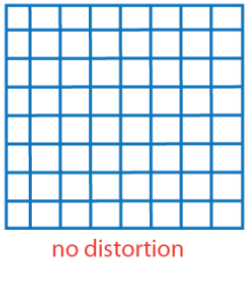
`'RadialDistortion'` — Radial distortion coefficients

`[0 0 0]` (default) | 2-element vector | 3-element vector

Radial distortion coefficients, specified as the comma-separated pair consisting of `'RadialDistortion'` and either a 2- or 3-element vector. If you specify a 2-element vector, the object sets the third element to `0`.

Radial distortion occurs when light rays bend more near the edges of a lens than they do at its optical center. The smaller the lens, the greater the distortion.





The camera parameters object calculates the radial distorted location of a point. You can denote the distorted points as ($x_{distorted}$, $y_{distorted}$), as follows:

$$x_{distorted} = x(1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6)$$

$$y_{distorted} = y(1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6)$$

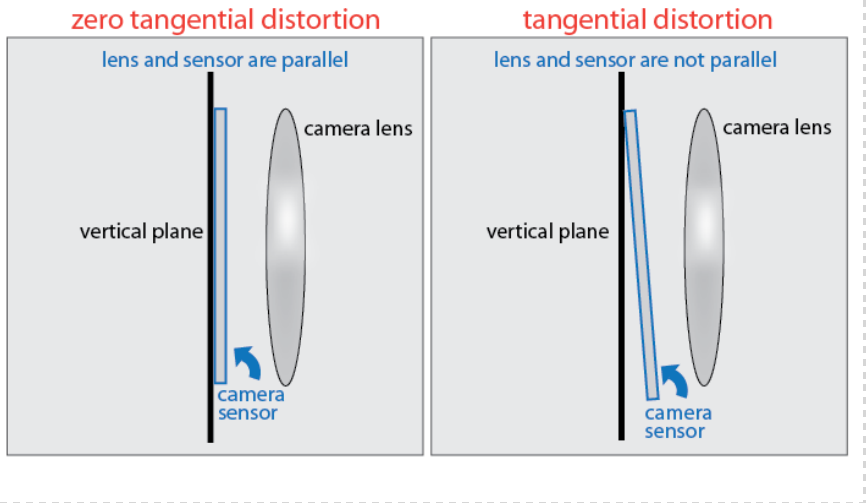
x, y = undistorted pixel locations
k_1, k_2, and k_3 = radial distortion coefficients of the lens
$r^2 = x^2 + y^2$

Typically, two coefficients are sufficient. For severe distortion, you can include k_3. The undistorted pixel locations appear in normalized image coordinates, with the origin at the optical center. The coordinates are expressed in world units.

`'TangentialDistortion'` — Tangential distortion coefficients

`[0 0]'` (default) | 2-element vector

Tangential distortion coefficients, specified as the comma-separated pair consisting of `'TangentialDistortion'` and a 2-element vector. Tangential distortion occurs when the lens and the image plane are not parallel.



The camera parameters object calculates the tangential distorted location of a point. You can denote the distorted points as ($x_{distorted}$, $y_{distorted}$), as follows:

$$x_{distorted} = x + [2 * p_1 * x * y + p_2 * (r^2 + 2 * x^2)]$$

$$y_{distorted} = y + [p_1 * (r^2 + 2 * y^2) + 2 * p_2 * x * y]$$

x, y = undistorted pixel locations
$p1$ and $p2$ = tangential distortion coefficients of the lens
$r^2 = x^2 + y^2$

The undistorted pixel locations appear in normalized image coordinates, with the origin at the optical center. The coordinates are expressed in world units.

'RotationVectors' — Camera rotations

M-by-3 matrix | [ ]

Camera rotations, specified as the comma-separated pair consisting of 'RotationVectors' and an M-by-3 matrix. The matrix contains rotation vectors for M images, which contain the calibration pattern that estimates the calibration parameters. Each row of the matrix contains a vector that describes the 3-D rotation of the camera relative to the corresponding pattern.

Each vector specifies the 3-D axis about which the camera is rotated. The magnitude of the vector represents the angle of rotation in medians. You can convert any rotation vector to a 3-by-3 rotation matrix using the Rodrigues formula.

You must set the RotationVectors and TranslationVectors properties together in the constructor to ensure that the number of rotation vectors equals the number of translation vectors. Setting only one property but not the other results in an error.

'TranslationVectors' — Camera translations

M-by-3 matrix | [ ]

Camera translations, specified as the comma-separated pair consisting of 'RotationVectors' and an M-by-3 matrix. This matrix contains translation vectors for M images. The vectors contain the calibration pattern that estimates the calibration parameters. Each row of the matrix contains a vector that describes the translation of the camera relative to the corresponding pattern, expressed in world units.

The following equation provides the transformation that relates a world coordinate $[X\ Y\ Z]$ and the corresponding image point $[x\ y]$:

$$s[x \quad y \quad 1] = [X \quad Y \quad Z \quad 1]\begin{bmatrix} R \\ t \end{bmatrix} K$$

R is the 3-D rotation matrix.
t is the translation vector.
K is the IntrinsicMatrix.
s is a scalar.

This equation does not take distortion into consideration. Distortion is removed by the undistortImage function.

You must set the RotationVectors and TranslationVectors properties together in the constructor to ensure that the number of rotation vectors equals the number of translation vectors. Setting only one property results in an error.

'WorldPoints' — World coordinates

M-by-2 array | [ ]

World coordinates of key points on calibration pattern, specified as the comma-separated pair consisting of 'WorldPoints' and an M-by-2 array. M represents the number of key points in the pattern.

'WorldUnits' — World points units

`'mm'` (default) | string

World points units, specified as the comma-separated pair consisting of '`WorldUnits`' and a string. The string describes the units of measure.

`'EstimateSkew'` — Estimate skew flag

`false` (default) | logical scalar

Estimate skew flag, specified as the comma-separated pair consisting of '`EstimateSkew`' and a logical scalar. When you set the logical to `true`, the object estimates the image axes skew. When you set the logical to `false`, the image axes are exactly perpendicular.

`'NumRadialDistortionCoefficients'` — Number of radial distortion coefficients

`2` (default) | `3`

Number of radial distortion coefficients, specified as the comma-separated pair consisting of '`NumRadialDistortionCoefficients`' and the number '`2`' or '`3`'.

`'NumRadialDistortionCoefficients'` — Number of radial distortion coefficients

`2` (default) | `3`

Number of radial distortion coefficients, specified as the comma-separated pair consisting of '`NumRadialDistortionCoefficients`' and the number '`2`' or '`3`'.

`'EstimateTangentialDistortion'` — Estimate tangential distortion flag

`false` (default) | logical scalar

Estimate tangential distortion flag, specified as the comma-separated pair of consisting of '`EstimateTangentialDistortion`' and the logical scalar `true` or `false`. When you set the logical to `true`, the object estimates the tangential distortion. When you set the logical to `false`, the tangential distortion is negligible.

Properties

collapse all

Intrinsic camera parameters:

`IntrinsicMatrix` — Projection matrix

3-by-3 identity matrix

Projection matrix, specified as a 3-by-3 identity matrix. The object uses the following format for the matrix format:

$$\begin{bmatrix} f_x & 0 & 0 \\ s & f_y & 0 \\ c_x & c_y & 1 \end{bmatrix}$$

The coordinates $[c_x\ c_y]$ represent the optical center (the principal point), in pixels. When the x and y axis are exactly perpendicular, the skew parameter, s, equals `0`.

$f_x = F * s_x$
$f_y = F * s_y$

F, is the focal length in world units, typically expressed in millimeters.
$[s_x, s_y]$ are the number of pixels per world unit in the x and y direction respectively.

fx and fy are expressed in pixels.

Camera lens distortion:

`RadialDistortion` — Radial distortion coefficients

`[0 0 0]` (default) | 2-element vector | 3-element vector

Radial distortion coefficients, specified as either a 2- or 3-element vector. When you specify a 2-element vector, the object sets the third element to `0`. Radial distortion occurs when light rays bend more near the edges of a lens than they do at its optical center. The smaller the lens, the greater the distortion. The camera parameters object calculates the radial distorted location of a point. You can denote the distorted points as ($x$distorted, $y$distorted), as follows:

$$x\text{distorted} = x(1 + k1*r^2 + k2*r^4 + k3*r^6)$$

$$y\text{distorted} = y(1 + k1*r^2 + k2*r^4 + k3*r^6)$$

x, y = undistorted pixel locations
$k1$, $k2$, and $k3$ = radial distortion coefficients of the lens
$r^2 = x^2 + y^2$

Typically, two coefficients are sufficient. For severe distortion, you can include $k3$. The undistorted pixel locations appear in normalized image coordinates, with the origin at the optical center. The coordinates are expressed in world units.

`TangentialDistortion` — Tangential distortion coefficients

`[0 0]'` (default) | 2-element vector

Tangential distortion coefficients, specified as a 2-element vector. Tangential distortion occurs when the lens and the image plane are not parallel. The camera parameters object calculates the tangential distorted location of a point. You can denote the distorted points as ($x$distorted, $y$distorted), as follows:

$$x\text{distorted} = x + [2 * p1 * y + p2 * (r^2 + 2 * x^2)]$$

$$y\text{distorted} = y + [p1 * (r^2 + 2*y^2) + 2 * p2 * x]$$

x, y = undistorted pixel locations
$p1$ and $p2$ = tangential distortion coefficients of the lens
$r^2 = x^2 + y^2$

The undistorted pixel locations appear in normalized image coordinates, with the origin at the optical center. The coordinates are expressed in world units.

Extrinsic camera parameters:

`RotationMatrices` — 3-D rotation matrix

3-by-3-by-P matrix (read-only)

3-D rotation matrix, specified as a 3-by-3-by-P, with P number of pattern images. Each 3-by-3 matrix represents the same 3-D rotation as the corresponding vector.

The following equation provides the transformation that relates a world coordinate [X Y Z] and the corresponding image point [x y]:

$$s[x \quad y \quad 1] = [X \quad Y \quad Z \quad 1]\begin{bmatrix} R \\ t \end{bmatrix} K$$

R is the 3-D rotation matrix.
t is the translation vector.
K is the `IntrinsicMatrix`.
s is a scalar.

This equation does not take distortion into consideration. Distortion is removed by the `undistortImage` function.

`TranslationVectors` — Camera translations

M-by-3 matrix | [ ]

Camera translations, specified as an M-by-3 matrix. This matrix contains translation vectors for M images. The vectors contain the calibration pattern that estimates the calibration parameters. Each row of the matrix contains a vector that describes the translation of the camera relative to the corresponding pattern, expressed in world units.

The following equation provides the transformation that relates a world coordinate $[X\ Y\ Z]$ and the corresponding image point $[x\ y]$:

$$s[x \quad y \quad 1] = [X \quad Y \quad Z \quad 1]\begin{bmatrix} R \\ t \end{bmatrix} K$$

R is the 3-D rotation matrix.
t is the translation vector.
K is the `IntrinsicMatrix`.
s is a scalar.

This equation does not take distortion into consideration. Distortion is removed by the `undistortImage` function.

You must set the `RotationVectors` and `TranslationVectors` properties in the constructor to ensure that the number of rotation vectors equals the number of translation vectors. Setting only one property but not the other results in an error.

Estimated camera parameter accuracy:

`MeanReprojectionError` — Average Euclidean distance

numeric value (read-only)

Average Euclidean distance between reprojected and detected points, specified as a numeric value in pixels.

`ReprojectionErrors` — Estimated camera parameters accuracy

M-by-2-by-P array

Estimated camera parameters accuracy, specified as an M-by-2-by-P array of $[x\ y]$ coordinates. The $[x\ y]$ coordinates represent the translation in x and y between the reprojected pattern key points and the detected pattern key points. The values of this property represent the accuracy of the estimated camera parameters. P is the number of pattern images that estimates camera parameters. M is the number of keypoints in each image.

`ReprojectedPoints` — World points reprojected onto calibration images

M-by-2-by-P array

World points reprojected onto calibration images, specified as an M-by-2-by-P array of $[x\ y]$ coordinates. P is the number of pattern images and M is the number of keypoints in each image.

Estimate camera parameters settings:

`NumPatterns` — Number of calibrated patterns
integer

Number of calibration patterns that estimates camera extrinsics, specified as an integer. The number of calibration patterns equals the number of translation and rotation vectors.

`WorldPoints` — World coordinates
M-by-2 array | `[]`

World coordinates of key points on calibration pattern, specified as an M-by-2 array. M represents the number of key points in the pattern.

`WorldUnits` — World points units
`'mm'` (default) | string

World points units, specified as a string. The string describes the units of measure.

`EstimateSkew` — Estimate skew flag
`false` (default) | logical scalar

Estimate skew flag, specified as a logical scalar. When you set the logical to `true`, the object estimates the image axes skew. When you set the logical to `false`, the image axes are exactly perpendicular.

`NumRadialDistortionCoefficients` — Number of radial distortion coefficients
`2` (default) | `3`

Number of radial distortion coefficients, specified as the number '2' or '3'.

`EstimateTangentialDistortion` — Estimate tangential distortion flag
`false` (default) | logical scalar

Estimate tangential distortion flag, specified as the logical scalar `true` or `false`. When you set the logical to `true`, the object estimates the tangential distortion. When you set the logical to `false`, the tangential distortion is negligible.

Methods

pointsToWorld	Determine world coordinates of image points

Output Arguments

expand all

`cameraParams` — Camera parameters
`cameraParameters` object

Examples

expand all

Remove Distortion from an Image Using the cameraParameters Object

- Measuring Planar Objects with a Calibrated Camera

References

[1] Zhang, Z. "A flexible new technique for camera calibration". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 11, pp. 1330–1334, 2000.

[2] 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.

See Also

`cameraCalibrator` | `detectCheckerboardPoints` | `estimateCameraParameters` | `generateCheckerboardPoints` | `showExtrinsics` | `showReprojectionErrors` | `stereoParameters` | `undistortImage`

More About

- Single Camera Calibration Using the Camera Calibrator App