stereoCalibrate() [1/2]

```
double cv::stereoCalibrate ( InputArrayOfArrays  objectPoints,
                             InputArrayOfArrays  imagePoints1,
                             InputArrayOfArrays  imagePoints2,
                             InputOutputArray    cameraMatrix1,
                             InputOutputArray    distCoeffs1,
                             InputOutputArray    cameraMatrix2,
                             InputOutputArray    distCoeffs2,
                             Size                imageSize,
                             InputOutputArray    R,
                             InputOutputArray    T,
                             OutputArray         E,
                             OutputArray         F,
                             OutputArray         perViewErrors,
                             int                 flags = CALIB_FIX_INTRINSIC ,
                             TermCriteria        criteria = TermCriteria(TermCriteria::COUNT+TermCriteria::EPS, 30, 1e-6)
                           )
```

Python:

```
cv.stereoCalibrate(          objectPoints, imagePoints1, imagePoints2, cameraMatrix1, distCoeffs1, cameraMatrix2, distCoeffs2, imageSize[, R[, T[, E[, F[, flags[, criteria]]]]]]

cv.stereoCalibrateExtended( objectPoints, imagePoints1, imagePoints2, cameraMatrix1, distCoeffs1, cameraMatrix2, distCoeffs2, imageSize, R, T[, E[, F[, perViewErrors[, flags[, criteria
```

```
#include <opencv2/calib3d.hpp>
```

Calibrates a stereo camera set up. This function finds the intrinsic parameters for each of the two cameras and the extrinsic parameters between the two cameras.

Parameters

objectPoints	Vector of vectors of the calibration pattern points. The same structure as in **calibrateCamera**. For each pattern view, both cameras need to see the same object objectPoints.size(), imagePoints1.size(), and imagePoints2.size() need to be equal as well as objectPoints[i].size(), imagePoints1[i].size(), and imagePoints2[i].s equal for each i.
imagePoints1	Vector of vectors of the projections of the calibration pattern points, observed by the first camera. The same structure as in **calibrateCamera**.
imagePoints2	Vector of vectors of the projections of the calibration pattern points, observed by the second camera. The same structure as in **calibrateCamera**.
cameraMatrix1	Input/output camera intrinsic matrix for the first camera, the same as in **calibrateCamera**. Furthermore, for the stereo case, additional flags may be used, see be
distCoeffs1	Input/output vector of distortion coefficients, the same as in **calibrateCamera**.
cameraMatrix2	Input/output second camera intrinsic matrix for the second camera. See description for cameraMatrix1.
distCoeffs2	Input/output lens distortion coefficients for the second camera. See description for distCoeffs1.
imageSize	Size of the image used only to initialize the camera intrinsic matrices.
R	Output rotation matrix. Together with the translation vector T, this matrix brings points given in the first camera's coordinate system to points in the second camer system. In more technical terms, the tuple of R and T performs a change of basis from the first camera's coordinate system to the second camera's coordinate s duality, this tuple is equivalent to the position of the first camera with respect to the second camera coordinate system.
T	Output translation vector, see description above.
E	Output essential matrix.
F	Output fundamental matrix.
perViewErrors	Output vector of the RMS re-projection error estimated for each pattern view.
flags	Different flags that may be zero or a combination of the following values:

- **CALIB_FIX_INTRINSIC** Fix cameraMatrix? and distCoeffs? so that only R, T, E, and F matrices are estimated.
- **CALIB_USE_INTRINSIC_GUESS** Optimize some or all of the intrinsic parameters according to the specified flags. Initial values are provided by the user.
- **CALIB_USE_EXTRINSIC_GUESS** R and T contain valid initial values that are optimized further. Otherwise R and T are initialized to the median value of (each dimension separately).
- **CALIB_FIX_PRINCIPAL_POINT** Fix the principal points during the optimization.
- **CALIB_FIX_FOCAL_LENGTH** Fix $f^{(j)}_x$ and $f^{(j)}_y$.
- **CALIB_FIX_ASPECT_RATIO** Optimize $f^{(j)}_y$. Fix the ratio $f^{(j)}_x/f^{(j)}_y$
- **CALIB_SAME_FOCAL_LENGTH** Enforce $f^{(0)}_x = f^{(1)}_x$ and $f^{(0)}_y = f^{(1)}_y$.
- **CALIB_ZERO_TANGENT_DIST** Set tangential distortion coefficients for each camera to zeros and fix there.
- **CALIB_FIX_K1**,..., **CALIB_FIX_K6** Do not change the corresponding radial distortion coefficient during the optimization. If **CALIB_USE_INTRINSIC_GUI** coefficient from the supplied distCoeffs matrix is used. Otherwise, it is set to 0.
- **CALIB_RATIONAL_MODEL** Enable coefficients k4, k5, and k6. To provide the backward compatibility, this extra flag should be explicitly specified to mak function use the rational model and return 8 coefficients. If the flag is not set, the function computes and returns only 5 distortion coefficients.
- **CALIB_THIN_PRISM_MODEL** Coefficients s1, s2, s3 and s4 are enabled. To provide the backward compatibility, this extra flag should be explicitly specif calibration function use the thin prism model and return 12 coefficients. If the flag is not set, the function computes and returns only 5 distortion coefficients
- **CALIB_FIX_S1_S2_S3_S4** The thin prism distortion coefficients are not changed during the optimization. If **CALIB_USE_INTRINSIC_GUESS** is set, the supplied distCoeffs matrix is used. Otherwise, it is set to 0.
- **CALIB_TILTED_MODEL** Coefficients tauX and tauY are enabled. To provide the backward compatibility, this extra flag should be explicitly specified to ma

function use the tilted sensor model and return 14 coefficients. If the flag is not set, the function computes and returns only 5 distortion coefficients.

- **CALIB_FIX_TAUX_TAUY** The coefficients of the tilted sensor model are not changed during the optimization. If **CALIB_USE_INTRINSIC_GUESS** is set, the supplied distCoeffs matrix is used. Otherwise, it is set to 0.

criteria	Termination criteria for the iterative optimization algorithm.

The function estimates the transformation between two cameras making a stereo pair. If one computes the poses of an object relative to the first camera and to the second camera, (R
T_2), respectively, for a stereo camera where the relative position and orientation between the two cameras are fixed, then those poses definitely relate to each other. This means, if the orientation (R, T) of the two cameras is known, it is possible to compute (R_2, T_2) when (R_1, T_1) is given. This is what the described function does. It computes (R, T) such that:

$$R_2 = RR_1$$

$$T_2 = RT_1 + T.$$

Therefore, one can compute the coordinate representation of a 3D point for the second camera's coordinate system when given the point's coordinate representation in the first camera system:

$$\begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \\ 1 \end{bmatrix} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \\ 1 \end{bmatrix}.$$

Optionally, it computes the essential matrix E:

$$E = \begin{bmatrix} 0 & -T_2 & T_1 \\ T_2 & 0 & -T_0 \\ -T_1 & T_0 & 0 \end{bmatrix} R$$

where T_i are components of the translation vector $T : T = [T_0, T_1, T_2]^T$. And the function can also compute the fundamental matrix F:

$$F = cameraMatrix2^{-T} \cdot E \cdot cameraMatrix1^{-1}$$

Besides the stereo-related information, the function can also perform a full calibration of each of the two cameras. However, due to the high dimensionality of the parameter space and r
data, the function can diverge from the correct solution. If the intrinsic parameters can be estimated with high accuracy for each of the cameras individually (for example, using **calibrate**
recommended to do so and then pass **CALIB_FIX_INTRINSIC** flag to the function along with the computed intrinsic parameters. Otherwise, if all the parameters are estimated at once,
restrict some parameters, for example, pass **CALIB_SAME_FOCAL_LENGTH** and **CALIB_ZERO_TANGENT_DIST** flags, which is usually a reasonable assumption.

Similarly to calibrateCamera, the function minimizes the total re-projection error for all the points in all the available views from both cameras. The function returns the final value of the r