Planar Camera Calibration Method

Planar camera calibration in this repository estimates a camera's internal parameters and lens distortion from multiple images of a flat target (checkerboard, dot grid, or any planar pattern). You provide two things for each view: the target point coordinates in the target's own coordinate system and the measured pixel coordinates of those same points in the image. The output is one shared camera model (intrinsics and distortion) plus one pose per view describing how the camera sat relative to the target in that image.

The method relies on the planar assumption: all target points lie on one plane in the target coordinate system. Under perspective projection, any plane-to-image mapping can be represented by one projective transform per view, called a homography. The algorithm uses homographies to get a strong initialization, then refines everything by minimizing pixel reprojection error.

1 Camera model being estimated

1.1 Intrinsics (pinhole camera)

A 3D point expressed in the camera coordinate system is written as a 3-vector whose components are measured along the camera axes: horizontal, vertical, and forward (depth). We denote that point as (X_c, Y_c, Z_c), where Z_c is the depth and must be positive for visible points.

$$\mathbf{X}_c = \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}, \qquad Z_c > 0$$

The pinhole camera first converts the 3D point to normalized image-plane coordinates by dividing by depth. Here x and y are dimensionless coordinates on the normalized plane.

$$x = \frac{X_c}{Z_c}, \qquad y = \frac{Y_c}{Z_c}$$

Intrinsics convert normalized coordinates into pixel coordinates. The intrinsic matrix K contains horizontal focal scaling f_x (pixels per unit of normalized x), vertical focal scaling f_y, and principal point (c_x, c_y) (the pixel location of the optical axis intersection). This repository assumes skew is negligible, so the skew entry is fixed at 0.

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

1.2 Lens distortion (Brown–Conrady: 2 radial + 2 tangential)

Real lenses deviate from the ideal pinhole model. This repository estimates four distortion coefficients using the Brown–Conrady form. Radial distortion mainly depends on distance from the principal point in normalized coordinates, while tangential distortion captures decentering effects. Define the squared radius r^2 using the normalized coordinates x and y.

$$r^2 = x^2 + y^2$$

The distorted normalized coordinates are (x_d, y_d). The coefficients k_1 and k_2 are radial distortion parameters, and p_1 and p_2 are tangential distortion parameters. All four are dimensionless.

$$x_d = x\,(1 + k_1 r^2 + k_2 r^4) + 2p_1 xy + p_2(r^2 + 2x^2)$$

$$y_d = y\,(1 + k_1 r^2 + k_2 r^4) + p_1(r^2 + 2y^2) + 2p_2 xy$$

Pixel projection then maps distorted normalized coordinates through the intrinsic matrix. The predicted pixel coordinates are (u, v), where u is the horizontal pixel coordinate and v is the vertical pixel coordinate. Using homogeneous coordinates $[u, v, 1]^\top$ is convenient because the mapping is affine in pixel space after distortion is applied.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix}$$

1.3 Per-view pose (extrinsics)

Each view has its own pose, meaning the camera's orientation and position relative to the target. A target point expressed in the target coordinate system is $\mathbf{X}_w = (X_w, Y_w, Z_w)^\top$. The pose is represented by a rotation matrix R (a 3×3 matrix that rotates vectors) and a translation vector $\mathbf{t}$ (a 3×1 vector that shifts the origin). The rigid transform from target coordinates to camera coordinates is:

$$\mathbf{X}_c = R\,\mathbf{X}_w + \mathbf{t}$$

For a planar target, all points satisfy $Z_w = 0$ in the target coordinate system. Here X and Y refer to coordinates along the plane axes you define on the target.

$$\mathbf{X}_w = \begin{bmatrix} X \\ Y \\ 0 \end{bmatrix}$$

2 Homography view model and estimation

The planar constraint implies a special structure: for a fixed view, the mapping from plane coordinates (X, Y) to image pixels (u, v), ignoring distortion, is a homography. In homogeneous form, the plane point is $[X, Y, 1]^\top$ and the pixel point is $[u, v, 1]^\top$. The scalar s is an arbitrary nonzero scale because homogeneous coordinates are defined up to scale.

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = H \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

The homography H for a view can be expressed in terms of intrinsics and pose. Let $\mathbf{r}_1$ and $\mathbf{r}_2$ be the first two columns of the rotation matrix R; these are the camera's rotated axes projected onto the plane directions. The matrix $[\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{t}]$ is a 3×3 matrix formed by concatenating those three 3-vectors as columns.

$$H = K\,[\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{t}]$$

2.1 Homography estimation per view (normalized DLT)

For each view, the algorithm estimates H from correspondences between target points (X_i, Y_i) and measured pixels (u_i, v_i). It uses normalized DLT. DLT builds a linear system $A\mathbf{h} = 0$, where $\mathbf{h}$ is a 9-vector containing the entries of H stacked into one column (it is solved up to a scale).

$$A\,\mathbf{h} = \mathbf{0}$$

The solution is taken from an SVD of A; the right-singular vector associated with the smallest singular value gives $\mathbf{h}$, which is reshaped into the 3×3 homography matrix.

Numerical stability depends heavily on coordinate scaling. The method applies Hartley normalization to both plane points and pixel points. Let T_X be the 3×3 normalization transform applied to plane homogeneous points $[X, Y, 1]^\top$, and let T_u be the 3×3 normalization transform applied to pixel homogeneous points $[u, v, 1]^\top$. The solver estimates a normalized homography H_n in the normalized coordinate systems, then converts it back:

$$H = T_u^{-1}\,H_n\,T_X$$

Here T_u^{-1} is the matrix inverse of T_u. Each view yields one homography H_v. With M views, you obtain $H_1, \ldots, H_M$, where M is the number of images used for calibration.

3 Intrinsics initialization from multiple homographies (Zhang-style constraints)

A single homography mixes intrinsics and pose. The algorithm exploits the fact that rotation columns are orthonormal. Using the rotation matrix $R = [\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{r}_3]$, the first two columns satisfy: they are orthogonal and have equal norm.

$$\mathbf{r}_1^\top \mathbf{r}_2 = 0$$

$$\mathbf{r}_1^\top \mathbf{r}_1 = \mathbf{r}_2^\top \mathbf{r}_2$$

To convert these into constraints on intrinsics, define a symmetric 3×3 matrix B that depends only on K. The notation $(\cdot)^{-\top}$ means "inverse transpose," so $K^{-\top} = (K^{-1})^\top$.

$$B = K^{-\top} K^{-1}$$

Write the homography columns as $H = [\mathbf{h}_1\ \mathbf{h}_2\ \mathbf{h}_3]$, where $\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3$ are the 3×1 column vectors of H. Each view yields two constraints that are linear in the entries of B:

$$\mathbf{h}_1^\top B\,\mathbf{h}_2 = 0$$

$$\mathbf{h}_1^\top B\,\mathbf{h}_1 = \mathbf{h}_2^\top B\,\mathbf{h}_2$$

Stacking these constraints across all views forms a linear system whose solution gives B up to scale. From B, the algorithm reconstructs K, extracting the intrinsics f_x, f_y, c_x, c_y under the assumption of zero skew.

At this initialization stage, distortion is set to zero to keep the initialization linear and stable. That means:

$$k_1 = 0, \quad k_2 = 0, \quad p_1 = 0, \quad p_2 = 0$$

4 Pose initialization per view

With an initial estimate of K, each view's homography can be normalized:

$$\tilde{H} = K^{-1} H$$

Let $\tilde{H} = [\tilde{\mathbf{h}}_1\ \tilde{\mathbf{h}}_2\ \tilde{\mathbf{h}}_3]$ be the columns of $\tilde{H}$. The algorithm uses a scale factor λ so that the first rotation column has unit length; $\|\cdot\|$ is Euclidean norm.

$$\lambda = \frac{1}{\|\tilde{\mathbf{h}}_1\|}$$

Then it forms initial rotation columns and translation:

$$\mathbf{r}_1 = \lambda \tilde{\mathbf{h}}_1, \quad \mathbf{r}_2 = \lambda \tilde{\mathbf{h}}_2, \quad \mathbf{t} = \lambda \tilde{\mathbf{h}}_3$$

The third rotation column is obtained from a cross product so that the rotation is right-handed:

$$\mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2$$

Because noise and linear estimation can produce a matrix that is close to, but not exactly, a valid rotation, the method projects the resulting 3×3 matrix $[\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{r}_3]$ onto the nearest rotation matrix using an SVD-based orthonormalization. For optimization, the rotation is then represented as a Rodrigues vector (a 3-vector whose direction is the rotation axis and whose magnitude is the rotation angle in radians). This is done because rotation matrices must remain orthonormal, while Rodrigues vectors can be updated freely during nonlinear optimization.

5 Nonlinear refinement by minimizing reprojection error

After initialization, the algorithm refines intrinsics, distortion, and all per-view poses by minimizing reprojection error across all points and views. For view v and point i, let the observed pixel be $\mathbf{u}_{v,i} = [u_{v,i}, v_{v,i}]^\top$. Let the predicted pixel from the current model be $\hat{\mathbf{u}}_{v,i}$. The residual is a 2D pixel error:

$$\mathbf{e}_{v,i} = \mathbf{u}_{v,i} - \hat{\mathbf{u}}_{v,i}$$

The objective function sums squared residual norms over all views and points. Here $\|\cdot\|$ is Euclidean norm in pixel units, and the total cost is a scalar:

$$J = \sum_{v=1}^{M} \sum_{i=1}^{N} \|\mathbf{e}_{v,i}\|^2$$

The parameter vector being optimized, denoted p, includes intrinsics (f_x, f_y, c_x, c_y), distortion (k_1, k_2, p_1, p_2), and each view's pose parameters (Rodrigues rotation vector components and translation components). The stacked residual vector is denoted $r(p)$; it contains all u and v residual components for all points across all views. The Jacobian matrix, denoted J_p, contains partial derivatives of residual components with respect to parameters.

5.1 Levenberg–Marquardt update rule

Levenberg–Marquardt computes an update Δp by solving a damped normal equation. The damping parameter is $\lambda \geq 0$, and I is an identity matrix sized to the number of active parameters.

$$\left(J_p^\top J_p + \lambda I\right) \Delta p = -J_p^\top r$$

The algorithm proposes $p \leftarrow p + \Delta p$ and checks whether the total cost J decreases. If it decreases, the step is accepted and λ is typically reduced to allow more Gauss–Newton-like behavior. If it does not decrease, the step is rejected and λ is increased to make the next attempt more conservative.

5.2 Numerical Jacobian (finite differences)

Both implementations compute the Jacobian numerically using finite differences. For the j-th parameter component p_j, a small step h is chosen, and $\mathbf{e}_j$ is a unit vector that selects only that component. The approximation uses the difference in residual vectors divided by the step size.

$$\frac{\partial r}{\partial p_j} \approx \frac{r(p + h\,\mathbf{e}_j) - r(p)}{h}$$

Here h must be chosen carefully because different parameters have different natural scales: focal scalings and principal point are in pixels, while distortion coefficients are dimensionless and often small. The code therefore uses relative step sizing, update caps, and parameter bounds to keep updates physically plausible and numerically stable.

5.3 Phased activation of parameters

To improve convergence stability, the refinement is done in phases that control which subsets of parameters are active. Phase 1 optimizes intrinsics and poses with distortion fixed to zero. Phase 2 additionally enables radial distortion coefficients k_1 and k_2. Phase 3 additionally enables tangential distortion coefficients p_1 and p_2. This phased approach reduces early coupling between intrinsics and distortion and makes it more likely the solver converges to a low-error basin.

6 Synthetic testing and ground-truth comparison

Both implementations include a synthetic harness used to validate correctness. It generates a planar grid of target points, samples camera poses, projects points through a known reference camera model, adds pixel noise, then runs the full calibration pipeline and compares estimated parameters to the reference parameters.

In these synthetic tests, the "ground truth" is the reference parameter set used to generate the observations. The baseline reprojection error is computed by projecting with the reference parameters and comparing to the noisy observations. This baseline is a noise-limited floor: even a perfect estimator should not systematically beat it because the measurements contain injected noise. The final refined reprojection error is computed using the estimated parameters after nonlinear refinement. When the final error is close to the baseline, it indicates the calibration recovered a model that explains the noisy measurements nearly as well as the reference model can.

The sweep reports repeat this process over multiple trials, often by resampling reference intrinsics for each trial while keeping distortion fixed. This produces averages and reveals sensitivity to pose diversity, noise, and optimization settings. The reports also show that the problem is non-convex: occasional runs can land in a poorer local minimum, especially when pose diversity is limited or when numerical Jacobian settings are not ideal.

7 Limitations and failure modes

The refinement objective is non-convex, so local minima can occur. Insufficient viewpoint diversity can make it difficult to separate focal scalings, principal point, and distortion. Correspondence errors or outlier measurements can dominate the residual sum and pull the solution away from a correct basin. Finite-difference Jacobians can be sensitive to step sizes and parameter scaling. Distortion terms can partially mimic changes in intrinsics if the dataset does not strongly constrain both.

Common mitigations include increasing iteration budgets, using central differences or analytic Jacobians for better derivative accuracy, adding restarts from slightly perturbed initial conditions, enforcing stronger pose diversity during data collection, and using robust losses when outliers exist.

8 Summary

The method combines a geometry-driven initialization with pixel-domain refinement. Per-view homographies compress each planar view into a stable projective mapping. Multiple homographies yield linear constraints that recover intrinsics shared across views. Homography factorization initializes per-view poses. A Levenberg–Marquardt refinement then minimizes reprojection error under a pinhole plus Brown–Conrady distortion model, enabling distortion coefficients in phases to stabilize convergence and improve reliability.