Planar Camera Calibration

1 Starting material

Known geometry

A planar calibration target provides a set of points whose coordinates are known in the target's own coordinate system. The model treats the target as a flat surface, so each point has a zero height coordinate. For point index $i \in \{1, \ldots, N\}$,

$$\mathbf{X}_i = \begin{bmatrix} X_i \\ Y_i \\ 0 \end{bmatrix}. \tag{1}$$

The pair (X_i, Y_i) are planar coordinates on the target. Any consistent unit system can be used. Units affect the numerical scale of translations and the implied scene scale, while leaving the structure of the calibration equations unchanged.

Two representations appear because different stages consume different forms. The planar coordinates (X_i, Y_i) are used when estimating a planar projective mapping. The lifted 3D points in (1) are used when predicting pixels through the full camera model.

Observations per view

For each view $v \in \{1, \ldots, M\}$, the measured image location of point i is recorded in pixels:

$$\mathbf{u}_{v,i} = \begin{bmatrix} u_{v,i} \\ v_{v,i} \end{bmatrix}. \tag{2}$$

These are the quantities the calibration procedure tries to reproduce. A successful calibration is one where a camera model can predict pixel locations that closely match the measured pixels across all views and points.

Parameters to estimate

Three parameter groups are estimated.

Shared intrinsics. The camera's intrinsic scalars are the focal scalings (f_x, f_y) and the principal point (c_x, c_y). Skew is fixed to $s = 0$, which matches typical sensors where pixel axes are orthogonal.

Shared distortion. Distortion parameters are shared across views because the lens does not change between images. The model uses two radial coefficients (k_1, k_2) and two tangential coefficients (p_1, p_2).

Per-view pose. Each view has a rigid transform from the target frame to the camera frame, represented by a rotation vector $\boldsymbol{\omega}_v \in \mathbb{R}^3$ and translation $\mathbf{t}_v \in \mathbb{R}^3$.

2 What you get back

The intrinsic matrix

The intrinsic matrix is constructed from the shared scalars:

$$\mathbf{K} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \qquad s = 0. \tag{3}$$

This matrix maps normalized direction ratios into pixel coordinates. The focal scalings turn unitless normalized coordinates into pixel distances. The principal point shifts the origin from the corner-based image coordinate system to the optical axis intersection.

Distortion

Distortion is returned as (k_1, k_2, p_1, p_2), where (k_1, k_2) shape radially symmetric warping and (p_1, p_2) capture asymmetric tangential effects.

Pose per view

For each view v, the returned pose is $(\boldsymbol{\omega}_v, \mathbf{t}_v)$. The rotation vector is a compact rotation representation used by the implementation, while the underlying rotation matrix appears inside the projection equations.

A pixel-domain score

A standard summary is the reprojection root-mean-square error (RMSE):

$$\mathrm{RMSE} = \sqrt{\frac{1}{MN} \sum_{v=1}^{M} \sum_{i=1}^{N} \|\mathbf{u}_{v,i} - \hat{\mathbf{u}}_{v,i}\|_2^2}. \tag{4}$$

This value is expressed in pixels. It can be read as the typical distance between a measured corner and its predicted location.

3 A flow sketch

Pseudocode stages

The procedure is organized as four stages. Each stage consumes outputs from earlier stages, and each stage is designed to be as stable as possible for the role it plays.

(1) Plane-to-image maps. For each view v, estimate a planar projective map $\mathbf{H}_v$ from planar coordinates (X_i, Y_i) and measured pixels $\mathbf{u}_{v,i}$.

$$\text{Inputs: planar points and pixels} \quad \to \quad \text{Output: } \mathbf{H}_v.$$

The collection $\{\mathbf{H}_v\}_{v=1}^{M}$ is consumed by stages (2) and (3).

(2) Intrinsic seed. Use all homographies $\{\mathbf{H}_v\}$ to solve for an initial intrinsic estimate $\mathbf{K}_0$.

$$\text{Inputs: } \{\mathbf{H}_v\} \text{ (from stage (1))} \quad \to \quad \text{Output: } \mathbf{K}_0.$$

This seed is consumed by stages (3) and (4).

(3) Pose seed per view. For each view, combine $\mathbf{K}_0$ and $\mathbf{H}_v$ to obtain an initial pose $(\boldsymbol{\omega}_v, \mathbf{t}_v)$.

Inputs: $\mathbf{K}_0$ (from stage (2)), $\mathbf{H}_v$ (from stage (1)) $\rightarrow$ Output: pose seed.

These pose seeds are consumed by stage (4).

(4) Reprojection refinement. Optimize intrinsics, distortion, and all poses by minimizing reprojection residuals in pixels using staged Levenberg–Marquardt.

Inputs: measurements and seeds (from stages (2)–(3)) $\rightarrow$ Outputs: refined parameters.

4 Core mapping equations

Rigid motion into camera coordinates

For view v, a target point $\mathbf{X}$ is transformed into the camera coordinate system:

$$\mathbf{X}_c = \mathbf{R}(\boldsymbol{\omega}_v)\,\mathbf{X} + \mathbf{t}_v, \tag{5}$$

where $\mathbf{R}(\boldsymbol{\omega}_v) \in SO(3)$ is the rotation matrix corresponding to the rotation vector $\boldsymbol{\omega}_v$. This equation is the mathematical statement that pose is a rigid motion: rotation aligns axes and translation shifts the origin.

Normalized perspective coordinates

With $\mathbf{X}_c = [X_c, Y_c, Z_c]^T$, normalized coordinates are

$$x = \frac{X_c}{Z_c}, \qquad y = \frac{Y_c}{Z_c}. \tag{6}$$

These ratios represent the direction of the 3D point relative to the camera, and they are the right space in which to apply a lens distortion model that depends on distance from the optical axis.

Distortion in normalized space

Define the squared radius

$$r^2 = x^2 + y^2, \qquad r^4 = (r^2)^2. \tag{7}$$

Radial distortion is a radius-dependent scale factor:

$$\text{radial}(r) = 1 + k_1 r^2 + k_2 r^4. \tag{8}$$

Tangential distortion adds asymmetric corrections. The model used is

$$x_d = x\,\text{radial}(r) + 2p_1 xy + p_2(r^2 + 2x^2), \tag{9}$$

$$y_d = y\,\text{radial}(r) + p_1(r^2 + 2y^2) + 2p_2 xy. \tag{10}$$

The structure of (9)–(10) is chosen so that distortion changes smoothly with radius while remaining low-order, which keeps the optimization problem well-behaved in typical calibration settings.

Normalized to pixels

The final step converts distorted normalized coordinates into pixel coordinates via intrinsics:

$$u = f_x x_d + c_x, \qquad v = f_y y_d + c_y. \tag{11}$$

This expresses that pixel coordinates are scaled versions of normalized direction ratios, shifted by the principal point.

Rodrigues rotation map

To compute $\mathbf{R}(\boldsymbol{\omega})$, let $\theta = \|\boldsymbol{\omega}\|$. For $\theta > 0$, define the unit axis $\mathbf{k} = \boldsymbol{\omega}/\theta$ and its cross-product matrix

$$[\mathbf{k}]_\times = \begin{bmatrix} 0 & -k_z & k_y \\ k_z & 0 & -k_x \\ -k_y & k_x & 0 \end{bmatrix}.$$

Rodrigues' formula is

$$\mathbf{R} = \mathbf{I} + \sin\theta\,[\mathbf{k}]_\times + (1-\cos\theta)\,[\mathbf{k}]_\times^2. \tag{12}$$

The inverse direction starts with the trace relation

$$\theta = \cos^{-1}\left(\frac{\operatorname{tr}(\mathbf{R}) - 1}{2}\right), \tag{13}$$

and recovers the axis from antisymmetric differences of $\mathbf{R}$.

5 Stage (1): plane-to-image maps

Homography model

A planar point in homogeneous coordinates $\mathbf{x} = [X, Y, 1]^T$ maps to an image point $\mathbf{u} = [u, v, 1]^T$ through a 3×3 homography:

$$\lambda\mathbf{u} = \mathbf{H}_v\mathbf{x}. \tag{14}$$

The scalar λ reflects projective equivalence: homogeneous vectors represent the same 2D point up to nonzero scale.

Normalization for numerical stability

Homography estimation is a linear least-squares problem in homogeneous form. Its numerical stability depends strongly on the coordinate scale. A similarity transform $\mathbf{T}$ is applied to both planar and image coordinates to center them and scale them so their mean distance to the origin is $\sqrt{2}$. Solving in this normalized space reduces imbalance in the linear system, which improves the reliability of the singular-vector solution used by DLT.

DLT linear system

Stacking N correspondences produces

$$\mathbf{A}\mathbf{h} = 0, \tag{15}$$

where $\mathbf{h}$ stacks the 9 entries of $\mathbf{H}_v$. The DLT solution is

$$\mathbf{h} = \arg\min_{\|\mathbf{h}\|=1} \|\mathbf{A}\mathbf{h}\|, \tag{16}$$

given by the right singular vector of $\mathbf{A}$ associated with the smallest singular value. The normalized homography is denormalized by

$$\mathbf{H}_v = \mathbf{T}_u^{-1}\,\tilde{\mathbf{H}}_v\,\mathbf{T}_x. \tag{17}$$

6 Stage (2): intrinsic seed

Homography structure under a pinhole model

Ignoring distortion during initialization, each homography decomposes as

$$\mathbf{H}_v \sim \mathbf{K}\,[\mathbf{r}_{1v}\ \mathbf{r}_{2v}\ \mathbf{t}_v], \tag{18}$$

where $\mathbf{r}_{1v}, \mathbf{r}_{2v}$ are the first two columns of a rotation matrix and $\mathbf{t}_v$ is translation. The symbol $\sim$ denotes equality up to a nonzero scale factor, which is inherent to homographies.

Linear constraints in B

Define

$$\mathbf{B} = \mathbf{K}^{-T}\mathbf{K}^{-1}. \tag{19}$$

Rotation columns are orthonormal, so

$$\mathbf{r}_{1v}^T\mathbf{r}_{2v} = 0, \qquad \mathbf{r}_{1v}^T\mathbf{r}_{1v} = \mathbf{r}_{2v}^T\mathbf{r}_{2v}. \tag{20}$$

Substituting $\mathbf{r}_{iv} \propto \mathbf{K}^{-1}\mathbf{h}_{iv}$ converts (20) into constraints on homography columns:

$$\mathbf{h}_{1v}^T\mathbf{B}\mathbf{h}_{2v} = 0, \qquad \mathbf{h}_{1v}^T\mathbf{B}\mathbf{h}_{1v} = \mathbf{h}_{2v}^T\mathbf{B}\mathbf{h}_{2v}. \tag{21}$$

Because $\mathbf{B}$ is symmetric, it has six unique entries. The constraints in (21) are linear in those six unknowns. Stacking all views yields

$$\mathbf{V}\mathbf{b} = 0, \tag{22}$$

which is solved by SVD in the same way as (16). The recovered $\mathbf{b}$ is then converted into (f_x, f_y, c_x, c_y) through closed-form relations coded in the implementation.

7 Stage (3): pose seed per view

Removing intrinsics and selecting scale

Given $\mathbf{K}_0$, compute

$$\mathbf{K}_0^{-1}\mathbf{H}_v = [\mathbf{h}_1' \ \mathbf{h}_2' \ \mathbf{h}_3']. \tag{23}$$

A scale s is selected so the first two columns become consistent with rotation columns. One convenient choice is an average-norm rule:

$$s = \frac{1}{\frac{1}{2}(\|\mathbf{h}_1'\| + \|\mathbf{h}_2'\|)}. \tag{24}$$

Then

$$\mathbf{r}_1 = s\mathbf{h}_1', \quad \mathbf{r}_2 = s\mathbf{h}_2', \quad \mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2, \quad \mathbf{t} = s\mathbf{h}_3'. \tag{25}$$

Projecting to a valid rotation

Noise can make $\mathbf{R} = [\mathbf{r}_1 \ \mathbf{r}_2 \ \mathbf{r}_3]$ slightly non-orthonormal. A standard correction is the SVD projection:

$$\mathbf{R} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T \quad \Rightarrow \quad \mathbf{R} \leftarrow \mathbf{U}\mathbf{V}^T. \tag{26}$$

If $\det(\mathbf{R}) < 0$, a sign flip is applied so that $\mathbf{R}$ becomes a proper rotation with determinant $+1$.

8 Stage (4): reprojection refinement

Packed parameter vector

All unknowns are placed into a single vector

$$\mathbf{p} = \begin{bmatrix} f_x & f_y & c_x & c_y & k_1 & k_2 & p_1 & p_2 & \boldsymbol{\omega}_1^T & \mathbf{t}_1^T \\ \cdots & \boldsymbol{\omega}_M^T & \mathbf{t}_M^T \end{bmatrix}^T. \tag{27}$$

This makes it possible to update shared parameters and all per-view poses within one least-squares optimization loop.

Residuals and objective

For each view and point, the model predicts $\hat{\mathbf{u}}_{v,i}$ by applying (5), (6), (9)–(10), and (11). The residual is

$$\mathbf{e}_{v,i} = \mathbf{u}_{v,i} - \hat{\mathbf{u}}_{v,i}. \tag{28}$$

Stacking all residuals produces $\mathbf{r}(\mathbf{p})$, and the cost is

$$C(\mathbf{p}) = \|\mathbf{r}(\mathbf{p})\|^2. \tag{29}$$

Writing the objective in pixel space keeps the meaning of the optimization transparent: it minimizes pixel disagreement between measured and predicted locations.

Finite-difference Jacobian with a stage mask

The Jacobian $\mathbf{J} = \partial \mathbf{r}/\partial \mathbf{p}$ is approximated by central differences:

$$\mathbf{J}_{:,j} \approx \frac{\mathbf{r}(\mathbf{p} + \epsilon_j \mathbf{e}_j) - \mathbf{r}(\mathbf{p} - \epsilon_j \mathbf{e}_j)}{2\epsilon_j}, \qquad \epsilon_j = \epsilon_{\text{scale}} \max(1, |p_j|). \tag{30}$$

A stage mask $\mathbf{m} \in \{0,1\}^{\dim(\mathbf{p})}$ selects which components of $\mathbf{p}$ are updated during a stage:

$$\Delta\mathbf{p} \leftarrow \mathbf{m} \odot \Delta\mathbf{p}. \tag{31}$$

Staging is used because early iterations can suffer from parameter coupling, especially between focal scalings and distortion. Allowing fewer degrees of freedom first helps the optimizer settle onto a geometrically consistent configuration before enabling additional parameters.

Levenberg–Marquardt update

Levenberg–Marquardt computes an update $\Delta\mathbf{p}$ by solving

$$(\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I})\Delta\mathbf{p} = -\mathbf{J}^T\mathbf{r}. \tag{32}$$

The damping λ balances local quadratic approximation and stability. When a proposed update reduces (29), λ is reduced, moving behavior toward Gauss–Newton. When a proposed update does not reduce cost, λ is increased, which increases damping and yields more conservative steps.

9 Synthetic validation components

Planar grid construction

A regular grid of target points is formed as

$$(X_i, Y_i) = (i\,s,\ j\,s), \tag{33}$$

then lifted to 3D by (1). A spread-out pattern improves conditioning for homography estimation and reduces degeneracy in the linear constraints used for the intrinsic seed.

Noisy observations

Synthetic pixel measurements are generated by adding image-plane noise to predicted pixels:

$$\mathbf{u}_{v,i} = \hat{\mathbf{u}}_{v,i} + \boldsymbol{\eta}, \qquad \boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \sigma_{\text{px}}^2\mathbf{I}). \tag{34}$$

This matches a common modeling assumption for corner localization uncertainty.

Reprojection scoring

The reprojection RMSE in (4) is used to compare seed estimates with the refined parameters. It also provides a quick check against the noise floor when ground truth parameters are available in a synthetic setting.

10 Notes that connect equations to the implementation

This section lists every function, states its role in the flow, and points to the equation(s) it implements or supports. Each function name is given only in parentheses.

Small utilities and parameter records

- Token parsing for optional script inputs (`parse_tokens`): no calibration equation; it prepares auxiliary configuration from a sequence of strings.
- Scalar bounding helper (`_clamp`): implements the clamp operation $\min(\max(x, \ell), h)$ used to keep values within chosen ranges.
- Intrinsic record and camera-matrix constructor (`Intrinsics` and its matrix accessor `K`): constructs (3).
- Distortion record (`Distortion`): stores (k_1, k_2, p_1, p_2) used by (9)–(10).
- Pose record (`Extrinsics`): stores $(\boldsymbol{\omega}_v, \mathbf{t}_v)$ used by (5).

Rotation and projection pieces

- Rotation-vector to rotation-matrix map (`rodrigues_to_R`): implements (12).
- Rotation-matrix to rotation-vector map (`rvec_from_R`): uses (13) and antisymmetric differences of $\mathbf{R}$.
- Normalized distortion map (`distort_normalized`): implements (7)–(10).
- Full pinhole-plus-distortion projector (`project_point`): composes (5), (6), (9)–(10), and (11).

Stage (1): homographies

- Similarity normalization of 2D coordinates (`normalize_2d`): supports the normalization discussion preceding (15).
- Homography estimation by DLT (`homography_dlt`): implements (14), (15)–(17).

Stage (2): intrinsic seed

- Constraint-vector constructor (`vij`): builds rows of $\mathbf{V}$ for (22) from homography columns in (21).
- Intrinsic initializer from many views (`intrinsics_from_homographies`): solves (22) and converts $\mathbf{b}$ into (f_x, f_y, c_x, c_y), defining (3).

Stage (3): pose seed

- Pose initializer from a homography and intrinsics (`extrinsics_from_KH`): implements (23)–(25) and enforces (26).

Stage (4): refinement

- Parameter packing (`pack_params`) and unpacking (`unpack_params`): implement the layout in (27).
- Parameter bounding (`clamp_params`): applies chosen bounds to keep the optimization numerically stable.
- Residual construction (`compute_residuals`): implements (28) and the stacking that defines $\mathbf{r}(\mathbf{p})$ for (29).
- Masked finite-difference Jacobian (`numeric_jacobian_active`): implements (30) for active parameters.
- Staged Levenberg–Marquardt loop (`lm_phased`): implements (32) with stage masking (31) and the damping adjustment logic.

Synthetic test and scoring

- Planar grid generator (`make_planar_grid`): implements (33) and lifts points via (1).
- Random pose sampler (`random_pose`): produces plausible $(\boldsymbol{\omega}, \mathbf{t})$ for synthetic views; it supports stage (1)–(4) testing.
- Synthetic measurement generator (`generate_observations`): implements (34) using the projector.
- Reprojection RMSE scorer (`rmse_reproj`): implements (4).
- In-bounds checker (`all_inside`) and first violating view finder (`first_outside_view`): no calibration equation; they validate synthetic visibility for reliable conditioning.
- End-to-end driver (`main`): wires together stages (1)–(4), then reports parameters and RMSE.

11 Why the chain holds together

A planar target makes it possible to summarize each view by a homography (stage (1)). Multiple homographies share a single intrinsic matrix, and the orthonormal structure of rotation columns yields linear constraints in $\mathbf{K}^{-T}\mathbf{K}^{-1}$ (stage (2)). With intrinsics seeded, each pose is obtained by removing intrinsics from a homography, choosing a consistent scale, and projecting the resulting matrix back to the rotation group (stage (3)). Reprojection refinement then minimizes pixel-domain residuals using a damped least-squares update (stage (4)), which aligns the model to the data while maintaining stability through damping and staged parameter activation.