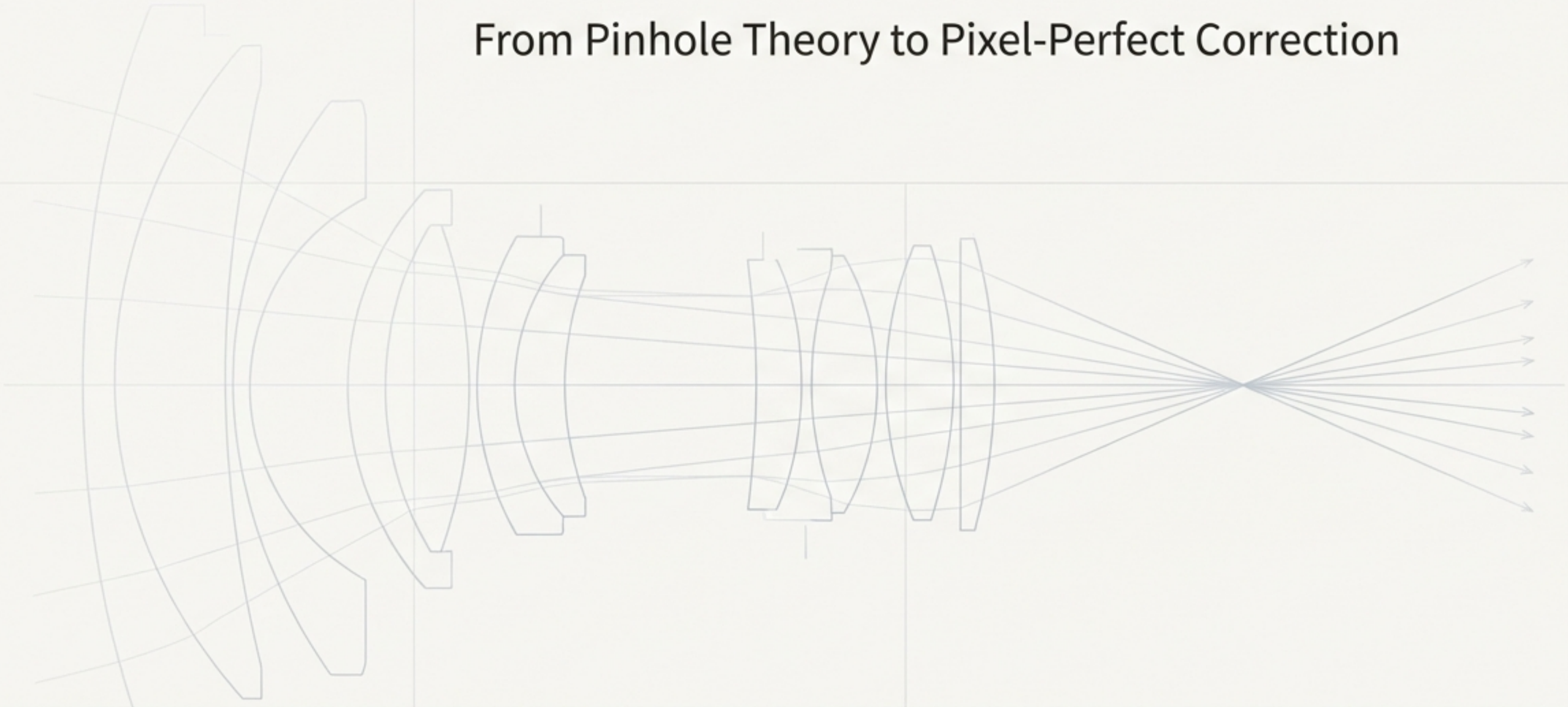
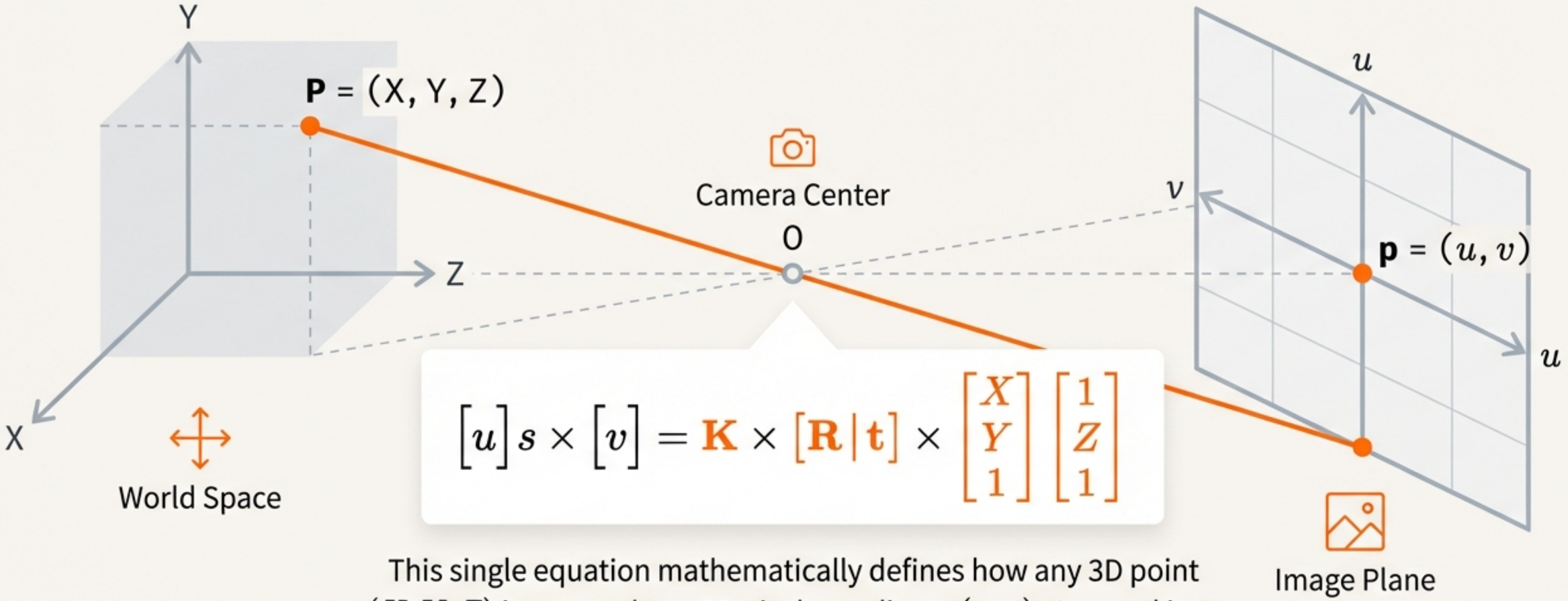
Module 7: Mastering Camera Calibration

From Pinhole Theory to Pixel-Perfect Correction



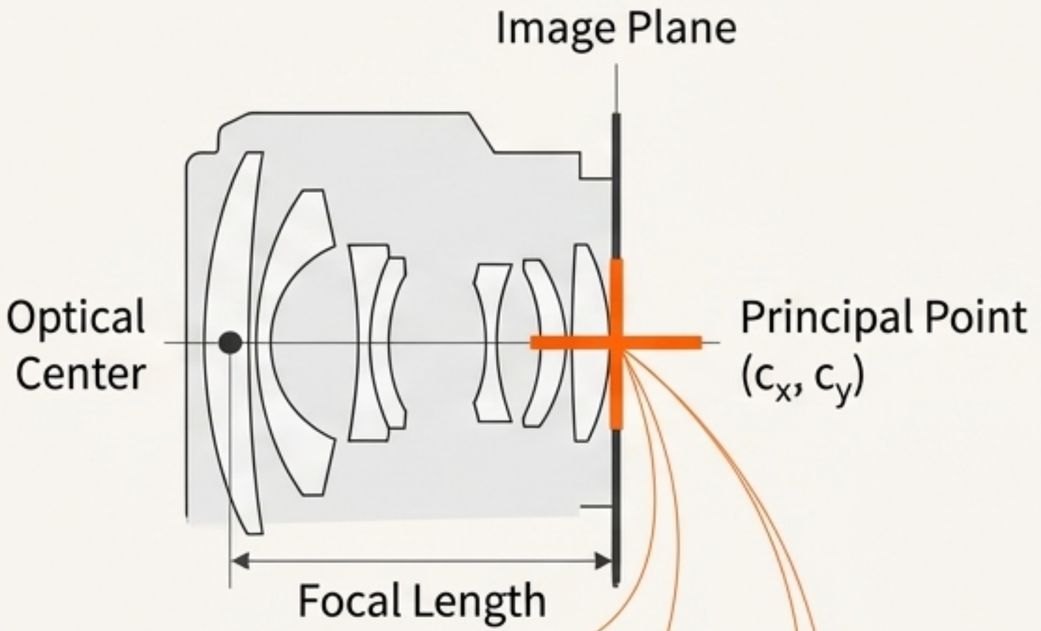
The Link Between a 3D World and a 2D Image



$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} s = \mathbf{K} \times [\mathbf{R}|\mathbf{t}] \times \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

This single equation mathematically defines how any 3D point (X, Y, Z) is mapped to a 2D pixel coordinate (u, v). Our goal is to understand and solve for each component of this model.

Building Block 1: The Camera's 'Soul' (Intrinsic Parameters)

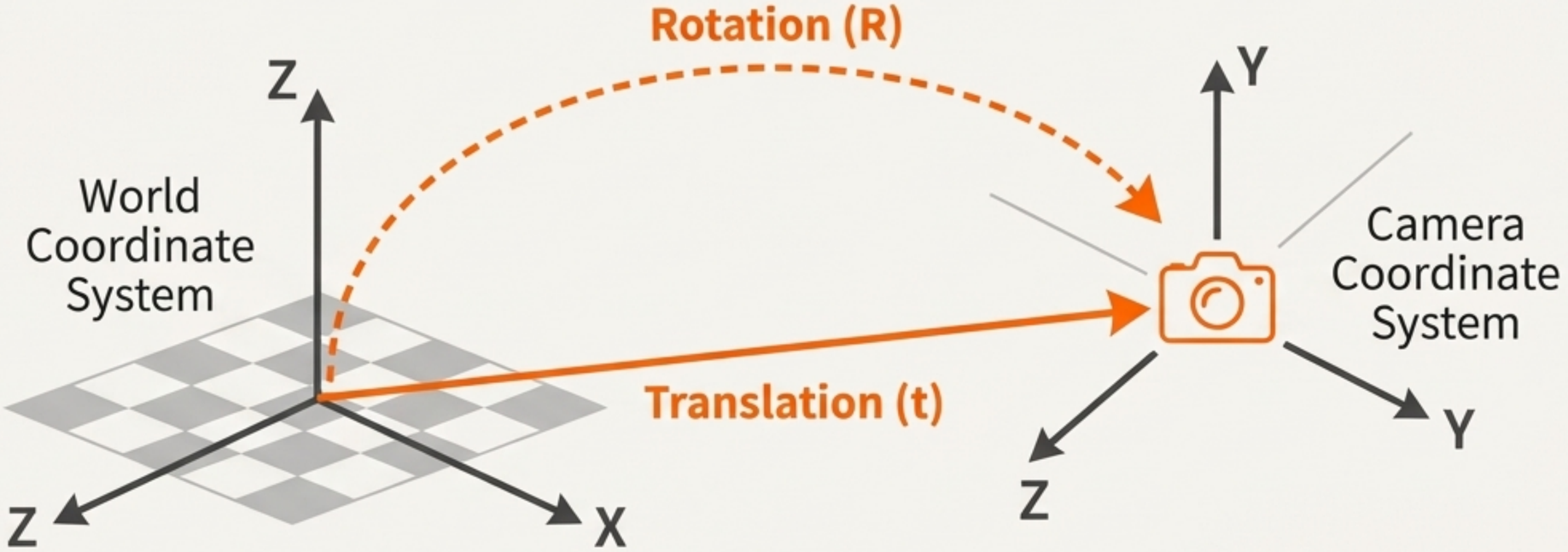


$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Symbol	Description
f_x, f_y	Focal length expressed in pixel dimensions.
c_x, c_y	The principal point: the pixel coordinate of the camera's optical axis.

These parameters are unique to a specific camera and lens combination. They don't change if the camera moves.

Building Block 2: The Camera's 'Stance' (Extrinsic Parameters)



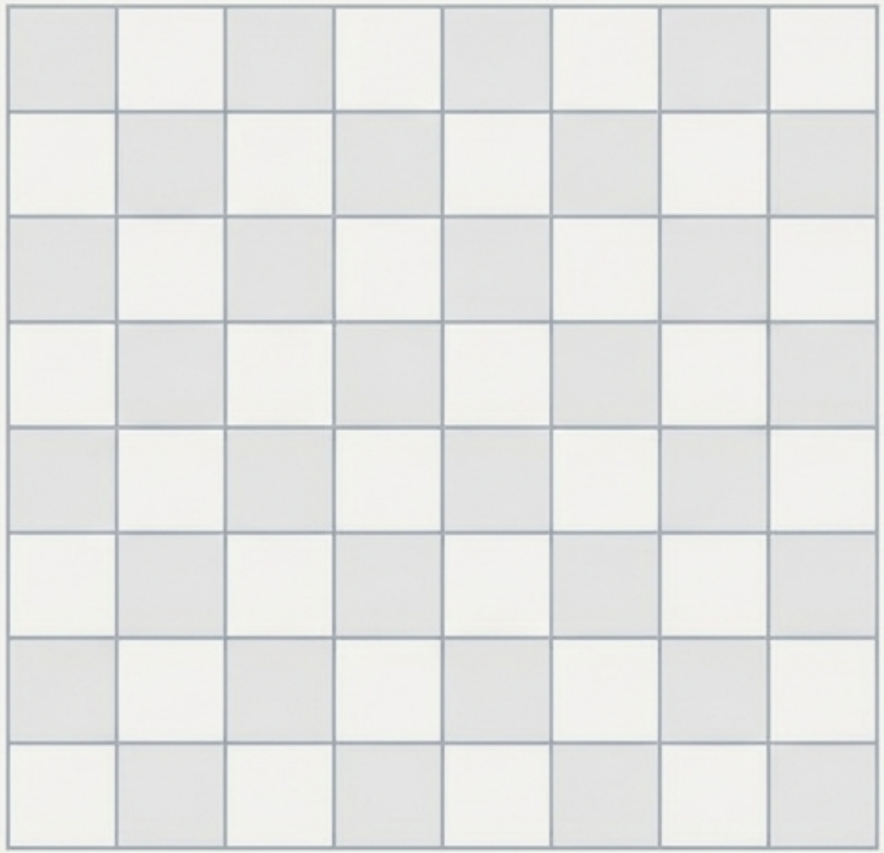
World-to-Camera Transformation: $[\mathbf{R} \mid \mathbf{t}]$

$$[X_c, Y_c, Z_c]^T = \mathbf{R} \times [X, Y, Z]^T + \mathbf{t}$$

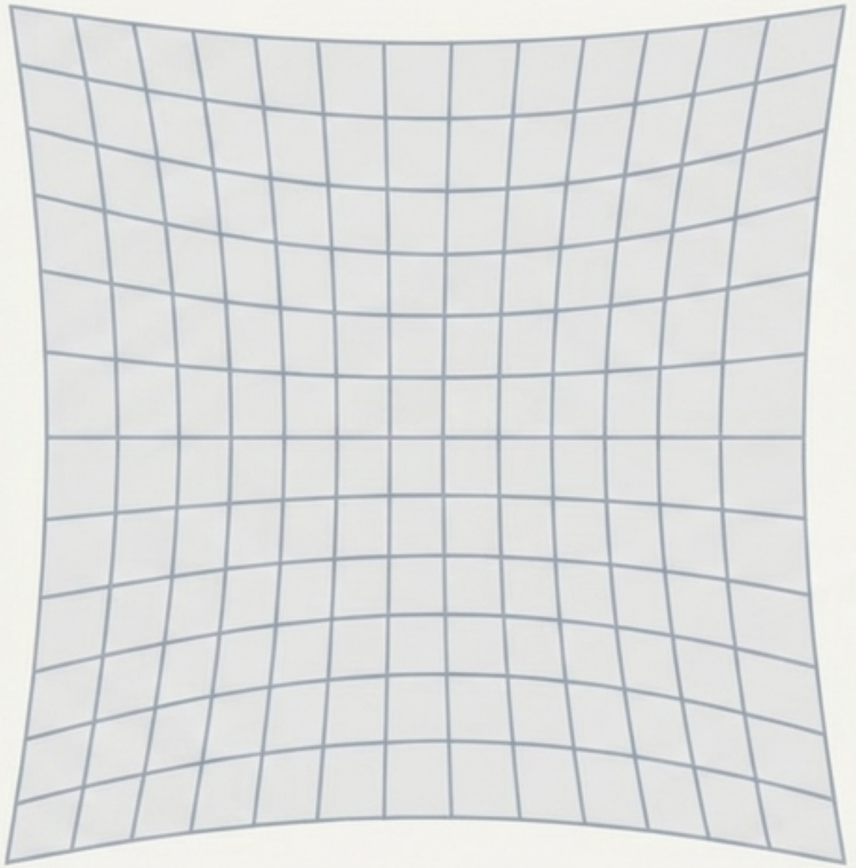
Extrinsics define the camera's pose. They change every time you move the camera.

The Complication: When Reality Deviates from the Ideal Model

Ideal Pinhole Model



Real-World Lens



Barrel Distortion ($k > 0$)

Pincushion Distortion ($k < 0$)

Real lenses are not perfect pinholes. They bend light in non-ideal ways, causing straight lines in the world to appear curved in the image. This is distortion.

The Mathematics of Imperfection: Modeling Distortion

Radial Distortion

The most significant distortion, caused by light bending more near the edges of the lens than at the center.

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

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

Tangential Distortion

Occurs when the lens and the image sensor are not perfectly parallel.

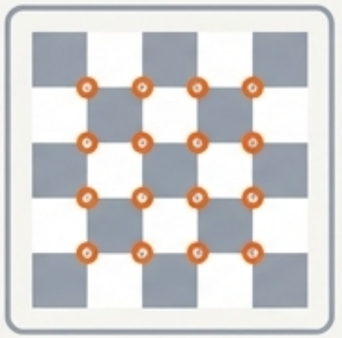
$$x_{distorted} = x + [2p_1xy + p_2(r^2 + 2x^2)]$$
$$y_{distorted} = y + [p_1(r^2 + 2y^2) + 2p_2xy]$$

The goal of calibration is to find the **Distortion Coefficients**, typically ordered in OpenCV as: $dist\_coeffs = [k_1, k_2, p_1, p_2, k_3]$.

The Solution: A Systematic Ritual to Find the Unknowns

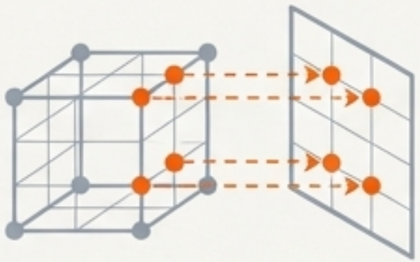
1. CAPTURE

Capture 10-20 images of a known pattern.



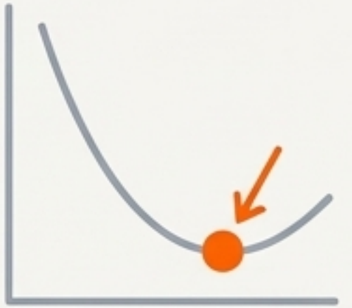
2. DETECT

Find the 2D pixel coordinates of pattern corners in each image.



3. ASSOCIATE

Create pairs of 3D world points `(X,Y,0)` and their corresponding 2D image points `(u,v)`.



4. SOLVE

Use an algorithm to find the **Camera Matrix (K)** and **Distortion Coefficients** that minimize the error between projected and detected points.

From Ritual to Code: Implementing Calibration in OpenCV

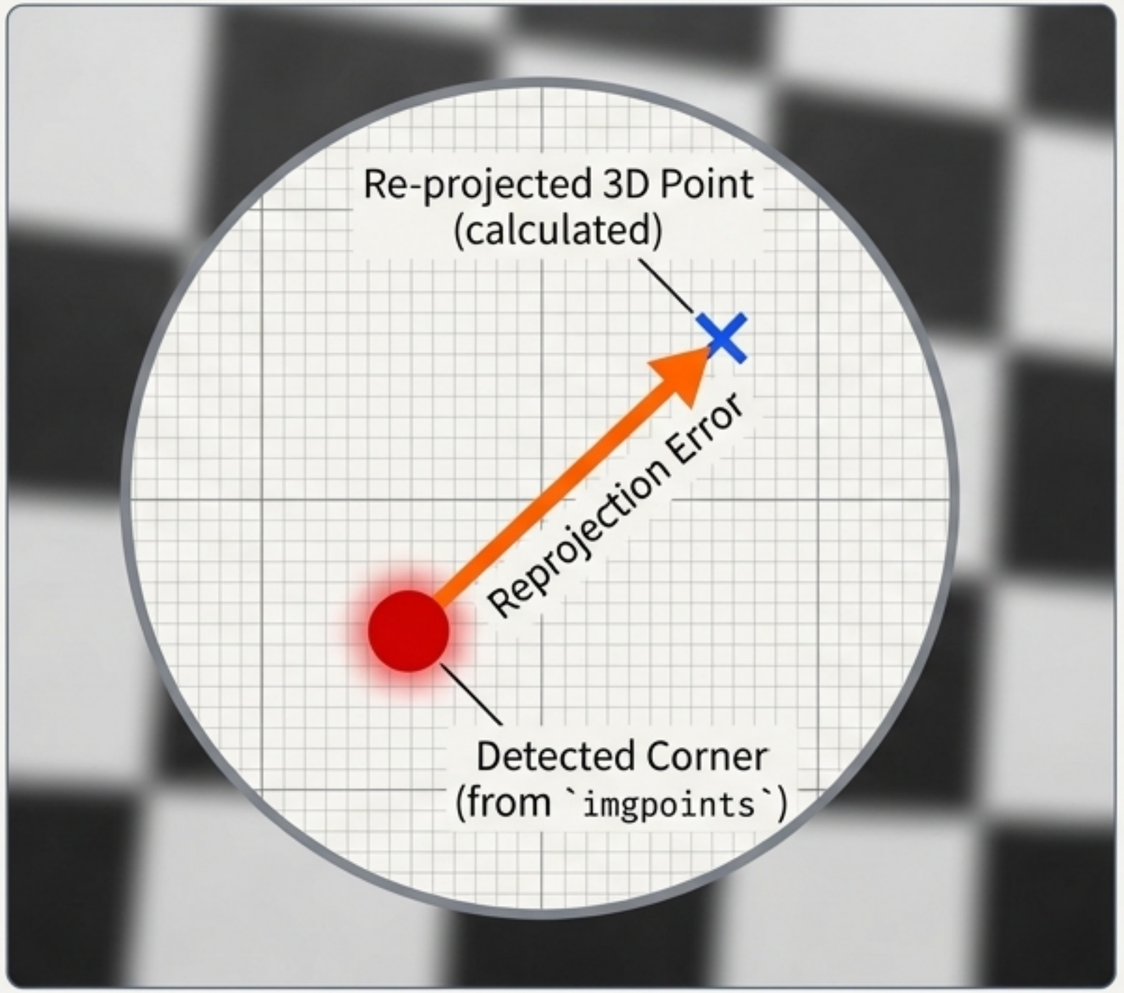
```
# objpoints: List of 3D world points for the checkerboard
# imgpoints: List of 2D detected corner points from all images

ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(
    objpoints,   # Input: Known 3D points
    imgpoints,   # Input: Detected 2D points
    imageSize,   # Input: Image resolution
    None, None
)
```

Output: The solved Camera Matrix **K**

Output: The solved **Distortion Coefficients**

Output: Extrinsic parameters (**R**, **t**) for each input image

The Measure of Success: Quantifying Reprojection Error

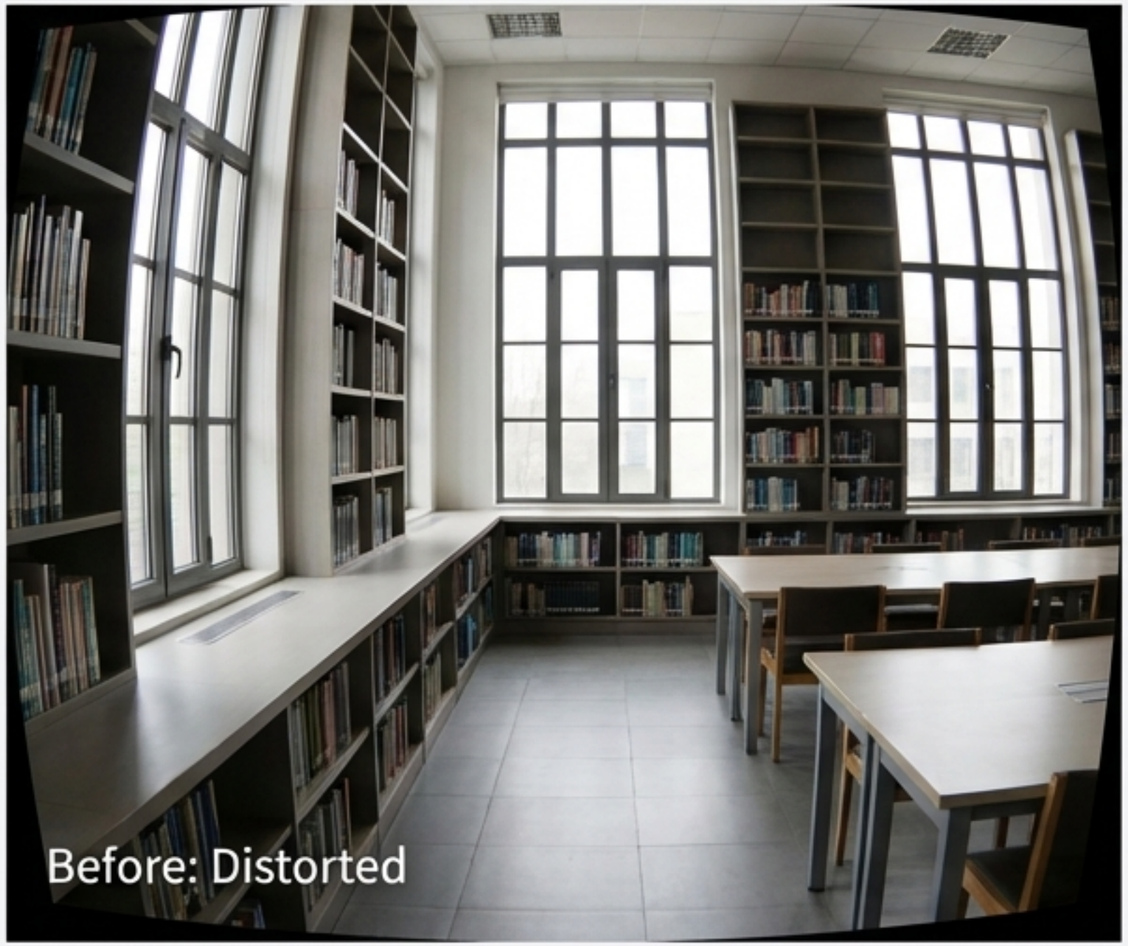


Reprojection error is the distance between a detected feature point and its corresponding 3D point projected back into the image using the computed camera parameters.

$$\texttt{error} = \frac{1}{N} \times \sum \| \texttt{projected\_point} - \texttt{detected\_point} \|^2$$

The ultimate goal: **A good calibration has a reprojection error of less than 0.5 pixels.**

Seeing is Believing: The Result of a Successful Calibration



Applying the Correction: Two Methods for Undistortion

Method 1: Simple & Direct

Source Sans Pro Regular

```
# Apply correction directly to a single image
undistorted_img = cv2.undistort(image, mtx, dist)
```

Use Case: Best for correcting individual images.

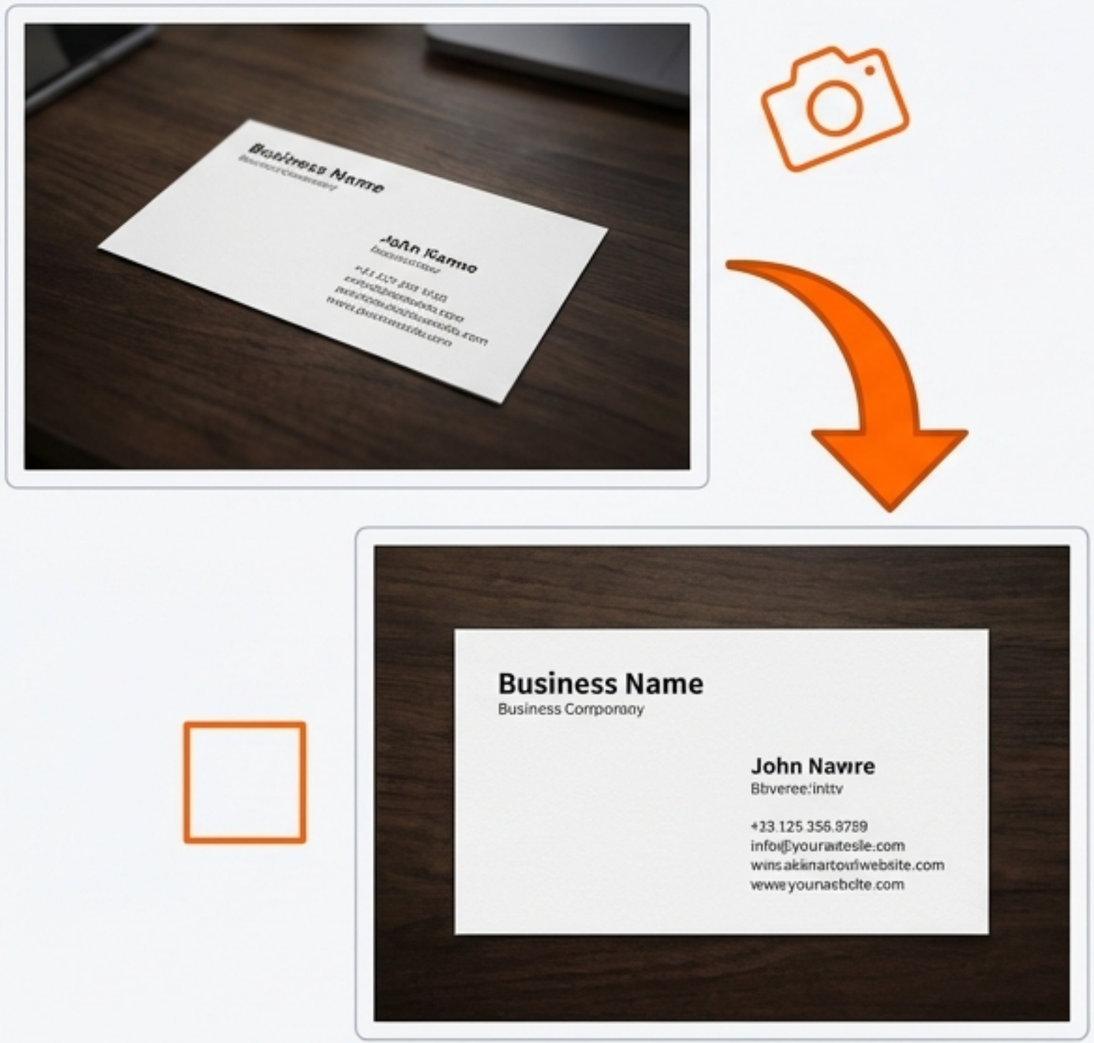
Method 2: Optimized Remapping

Source Sans Pro Regular

```
# Step 1: Compute the mapping once
mapx, mapy = cv2.initUndistortRectifyMap(mtx, dist, ...)

# Step 2: Apply the map to any image
undistorted_img = cv2.remap(image, mapx, mapy, cv2.INTER_LI
```

Use Case: Significantly faster for video streams or correcting many images, as the expensive computation is only done once.

Unlocking a New Tool: The Perspective Transform



How It Works

By providing 4 source points on the original image and their 4 corresponding destination points, we can compute a 3x3 Homography matrix (`H`) to warp the perspective.

</> Core Code Snippet

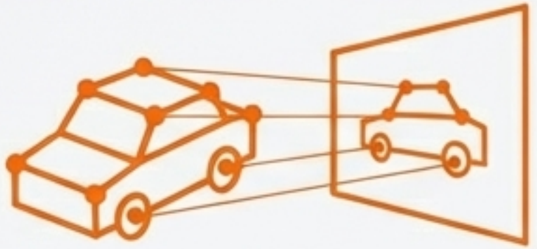
```
# Define source and destination corner points
src = np.float32([[x1,y1], [x2,y2], ...])
dst = np.float32([[0,0], [w,0], ...])

# Compute the transform and apply it
M = cv2.getPerspectiveTransform(src, dst)
warped = cv2.warpPerspective(image, M, (w, h))
```

The Power of Homography: A Fundamental Concept in Vision

The perspective transform is a specific application of Homography, a matrix that relates any two images of the same planar surface.

Image Stitching

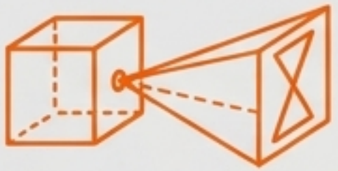
Augmented Reality



Pose Estimation

- **Degrees of Freedom:** 8
- **Minimum Points:** Requires 4 non-collinear point correspondences.
- **OpenCV Function:** `cv2.findHomography()`

Your Camera Calibration Toolkit



The Pinhole Model

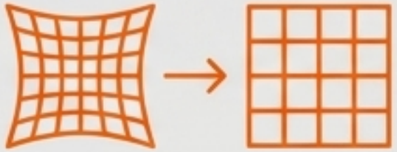
The core mathematical model ($\mathbf{s[u,v,1]^T = K[R|t]X}$) that links the 3D world to your 2D image.

The Unknowns

Calibration is the process of solving for the camera's internal DNA (**Intrinsics K**, **Distortion** coeffs) and its pose (**Extrinsics [R|t]**).

The Standard Method

Use a **known pattern** like a **checkerboard** to find point correspondences and minimize reprojection error to find the parameters.



The Payoff

A successful calibration allows you to remove lens distortion and perform powerful perspective transformations.

Further Reading & Core Resources

OpenCV Official Tutorial

For practical code examples and explanations.
Link: docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html

"A Flexible New Technique for Camera Calibration"

The original paper by Z. Zhang that is the basis for modern techniques.
Link: microsoft.com/en-us/research/wp-content/uploads/2016/02/tr98-71.pdf

"Multiple View Geometry in Computer Vision"

The definitive academic text for a deep theoretical understanding.
Reference: By Richard Hartley and Andrew Zisserman.