中国科学技术大学
University of Science and Technology of China

Camera Calibration

张举勇
中国科学技术大学

Camera Calibration

- Method to find a camera's internal and external parameters.

Topics:

- (1) Linear Camera Model
- (2) Camera Calibration
- (3) Extracting Intrinsic and Extrinsic Matrices
- (4) Example Application: Simple Stereo

Forward Imaging Model: 3D to 2D

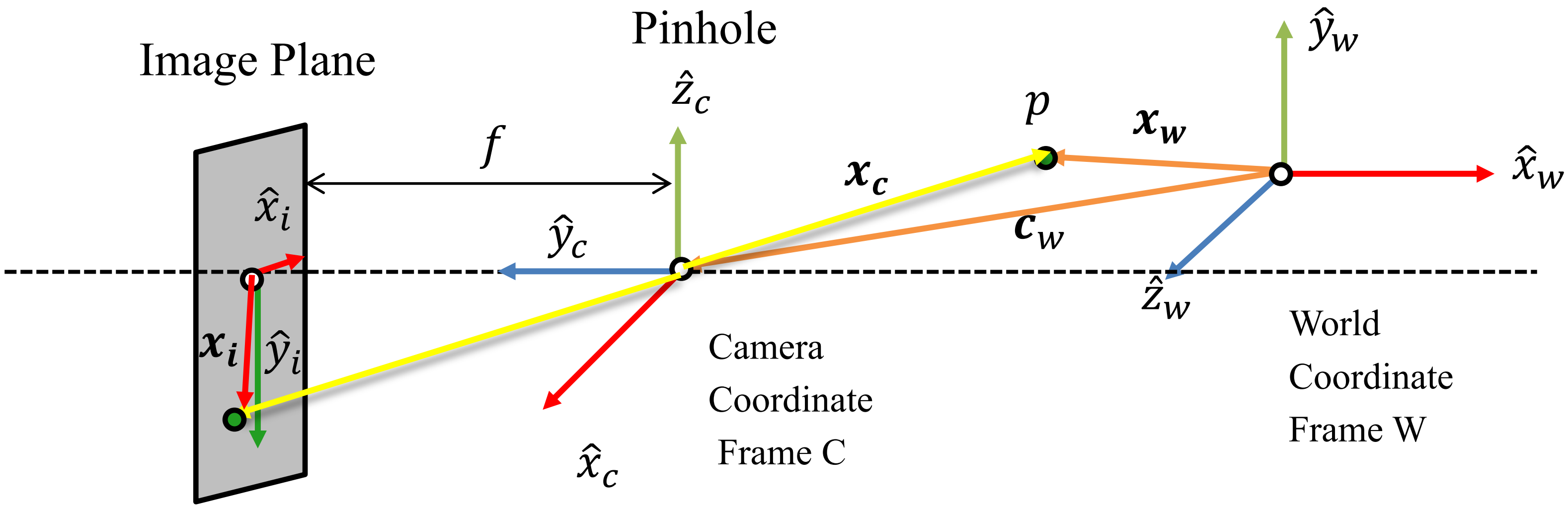


Image Coordinates

$$\boldsymbol{x}_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$

Perspective Projection

Camera Coordinates

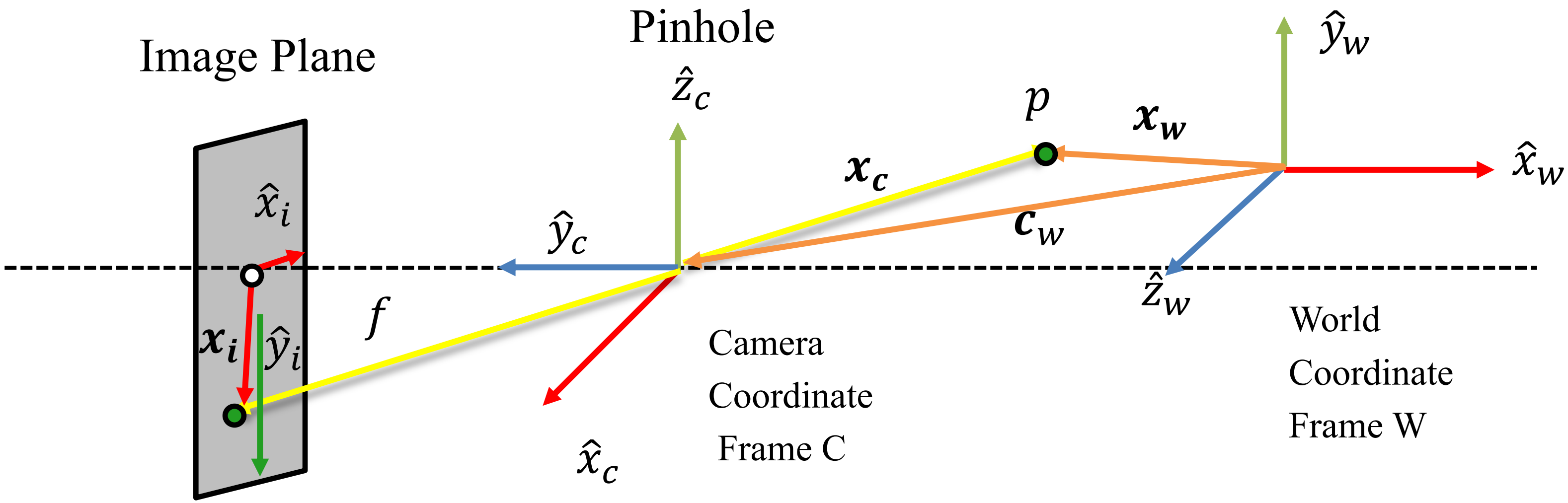
$$\boldsymbol{x}_c = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$$

Coordinate Transformation

World Coordinates

$$\boldsymbol{x}_w = \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix}$$

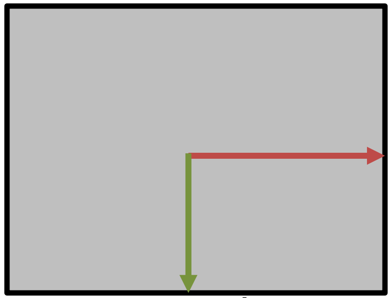
Forward Imaging Model: 3D to 2D

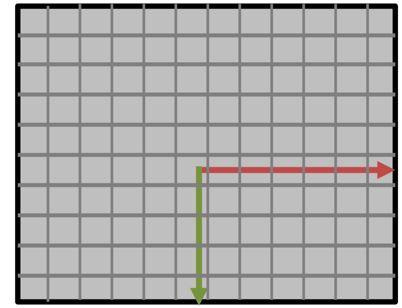


We know that $\frac{x_i}{f} = \frac{x_c}{z_c}$ and $\frac{y_i}{f} = \frac{y_c}{z_c}$

Therefore: $x_i = f\frac{x_c}{z_c}$ and $y_i = f\frac{y_c}{z_c}$

Image Plane to Image Sensor Mapping



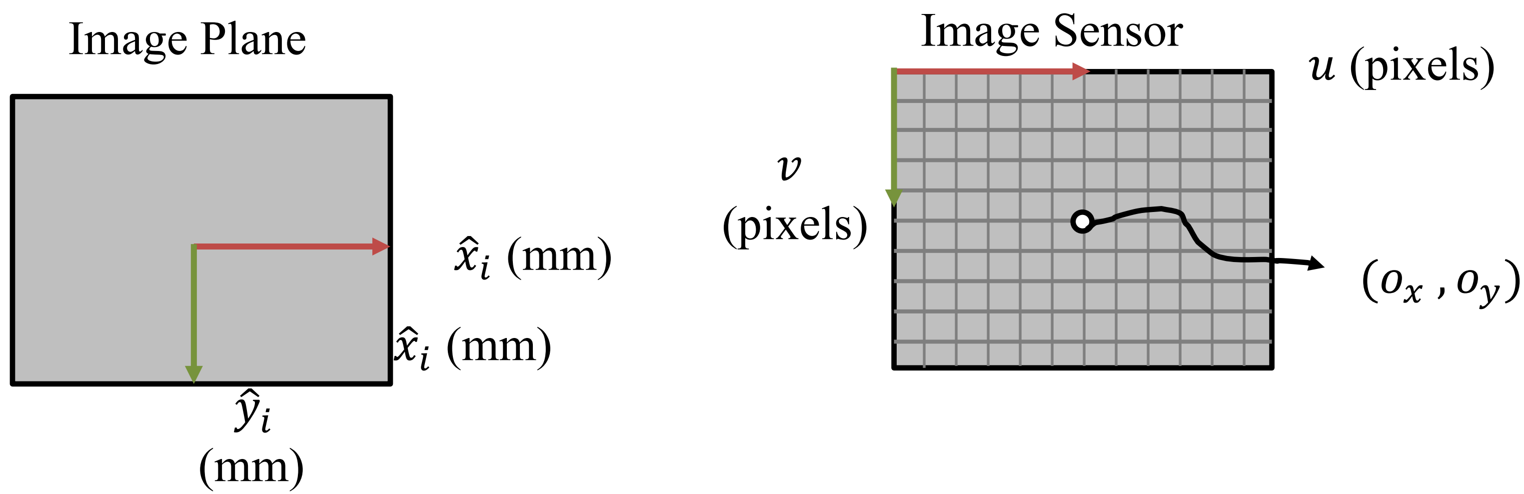


Pixels may be rectangular.

If m_x and m_y are the pixel densities (pixels/mm) in x and y directions, respectively, then pixel coordinates are:

$$u = m_x x_i = m_x f \frac{x_c}{z_c} \qquad v = m_y y_i = m_y f \frac{y_c}{z_c}$$

Image Plane to Image Sensor Mapping



We usually treat the top-left corner of the image sensor as its origin (easier for indexing). If pixel (o_x, o_y) is the **Principle Point** where the optical axis pierces the sensor, then:

$$u = m_x f \frac{x_c}{z_c} + o_x \qquad v = m_y f \frac{y_c}{z_c} + o_y$$

Perspective Projection

$$u = m_x f \frac{x_c}{z_c} + o_x \qquad v = m_y f \frac{y_c}{z_c} + o_y$$

$$\boxed{u = f_x \frac{x_c}{z_c} + o_x \qquad v = f_y \frac{y_c}{z_c} + o_y}$$

where: $(f_x, f_y) = (m_x f, m_y f)$ are the focal lengths in pixels in the x and y directions.

(f_x, f_y, o_x, o_y): **Intrinsic parameters** of the camera.
They represent the **camera's internal geometry.**

Perspective Projection

$$u = m_x f \frac{x_c}{z_c} + o_x \qquad v = m_y f \frac{y_c}{z_c} + o_y$$

$$\boxed{u = f_x \frac{x_c}{z_c} + o_x \qquad v = f_y \frac{y_c}{z_c} + o_y}$$

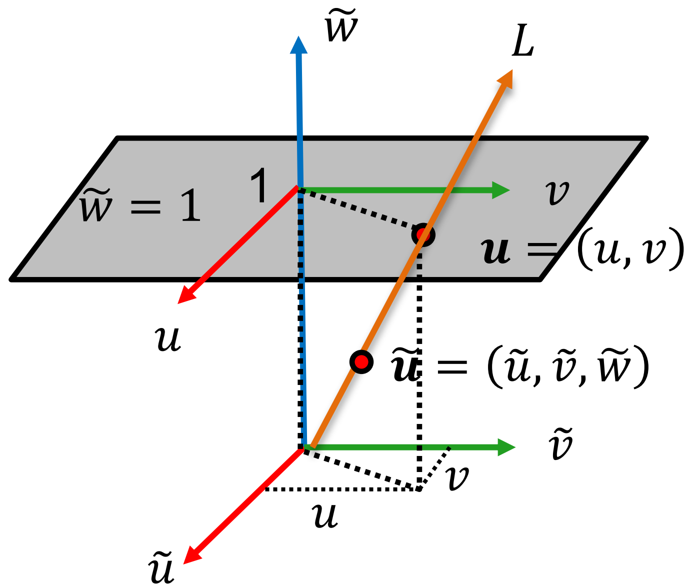
Equations for perspective projection are **Non-Linear**.
It is convenient to express them as linear equations.

Homogenous Coordinates

The homogenous representation of a 2D point $\boldsymbol{u} = (u, v)$ is a 3D point $\tilde{\boldsymbol{u}} = (\tilde{u}, \tilde{v}, \tilde{w})$. The third coordinate $\tilde{w} \neq 0$ is fictitious such that:

$$u = \frac{\tilde{u}}{\tilde{w}} \qquad v = \frac{\tilde{v}}{\tilde{w}}$$

$$\boxed{\boldsymbol{u} \equiv \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{w}u \\ \tilde{w}v \\ \tilde{u} \end{bmatrix} \equiv \begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{u} \end{bmatrix} = \tilde{\boldsymbol{u}}}$$



Every point on line L (except origin) represents the homogenous coordinate of u(u, v)

Homogenous Coordinates

The **homogenous** representation of a 3D point $\boldsymbol{x} = (x, y, z) \in \mathcal{R}^3$ is a 4D point $\widetilde{\boldsymbol{x}} = (\tilde{x}, \tilde{y}, \tilde{z}, \widetilde{w}) \in \mathcal{R}^4$. The fourth coordinate w $\neq$0 is fictitious such that:

$$x = \frac{\tilde{x}}{\widetilde{w}} \quad y = \frac{\tilde{y}}{\widetilde{w}} \quad z = \frac{\tilde{z}}{\widetilde{w}}$$

$$\boxed{\boldsymbol{x} \equiv \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \widetilde{w}x \\ \widetilde{w}y \\ \widetilde{w}z \\ \widetilde{w} \end{bmatrix} \equiv \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \\ \widetilde{w} \end{bmatrix} = \widetilde{\boldsymbol{x}}}$$

Perspective Projection

Perspective projection equations:

$$u = f_x \frac{x_c}{z_c} + o_x \quad v = f_y \frac{y_c}{z_c} + o_y$$

Homogenous coordinates of (u, v) :

$$\boxed{\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} \equiv \begin{bmatrix} z_c u \\ z_c v \\ z_c \end{bmatrix} = \begin{bmatrix} f_x x_c + z_c o_x \\ f_y y_c + z_c o_y \\ z_c \end{bmatrix} = \begin{bmatrix} f_x & 0 & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}}$$

where: $(u, v) = (\tilde{u}/\tilde{w}, \tilde{v}/\tilde{w})$

Linear Model for Perspective Projection

Intrinsic Matrix

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} = \begin{bmatrix} f_x & 0 & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

Calibration Matrix:

$$K = \begin{bmatrix} f_x & 0 & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix}$$

Upper Right Triangular Matrix

Intrinsic Matrix:

$$M_{int} = [\,K \mid 0\,] = \begin{bmatrix} f_x & 0 & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\boxed{\tilde{u} = [\,K \mid 0\,]\tilde{\boldsymbol{x}}_c = M_{int}\widetilde{x_c}}$$

Forward Imaging Model: 3D to 2D

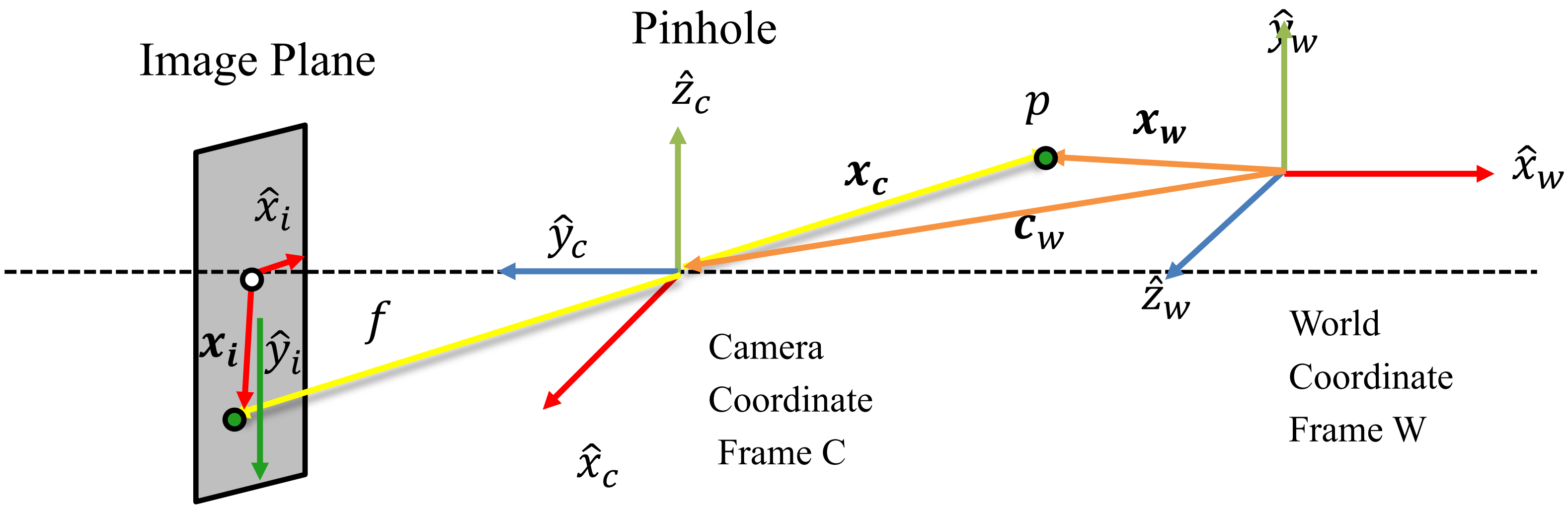
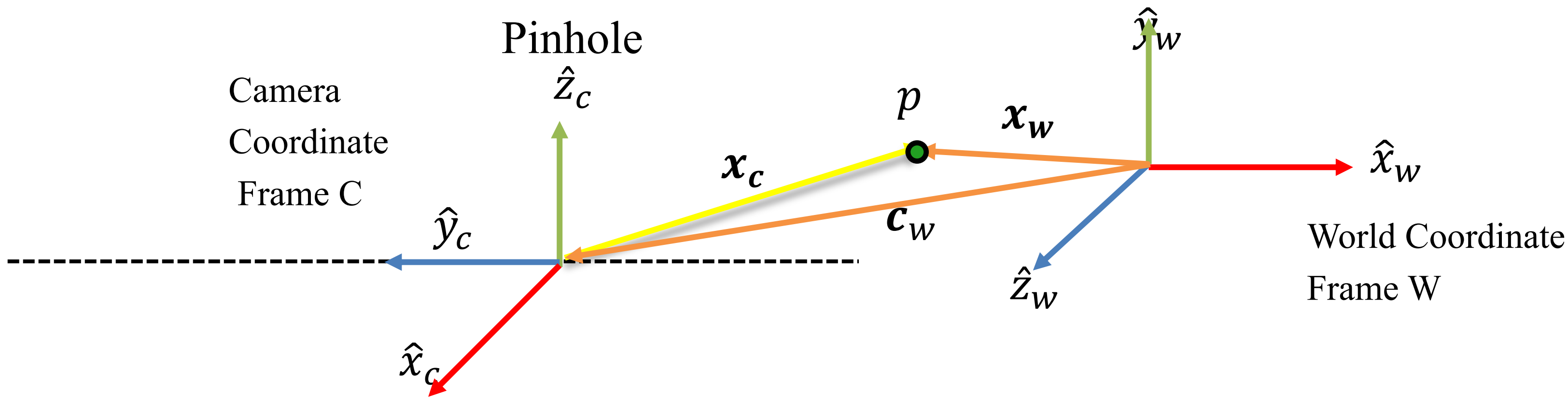


Image Coordinates	Camera Coordinates	World Coordinates
$\boldsymbol{x}_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}$ $\xleftarrow{\boldsymbol{M}_{int}}$ Perspective Projection	$\boldsymbol{x}_c = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$	$\xleftarrow{?}$ $\boldsymbol{x}_w = \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix}$ Coordinate Transformation

Extrinsic Parameters



Position c_w and **Orientation** R of the camera in the world coordinate frame W are the camera's **Extrinsic Parameters**.

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

→ Row 1: Direction of $\hat{x}_c$ in world coordinate frame

→ Row 2: Direction of $\hat{y}_c$ in world coordinate frame

→ Row 3: Direction of $\hat{z}_c$ in world coordinate frame

Orientation/Rotation Matrix R is **Orthonormal**

Orthonormal Vectors and Matrices

Orthonormal Vectors: Two vectors **u** and **v** are orthonormal if and only if:

$$dot(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}^T \boldsymbol{v} = 0 \quad \text{and} \quad \boldsymbol{u}^T \boldsymbol{u} = \boldsymbol{v}^T \boldsymbol{v} = 1$$

(Orthogonality) **(Unit length)**

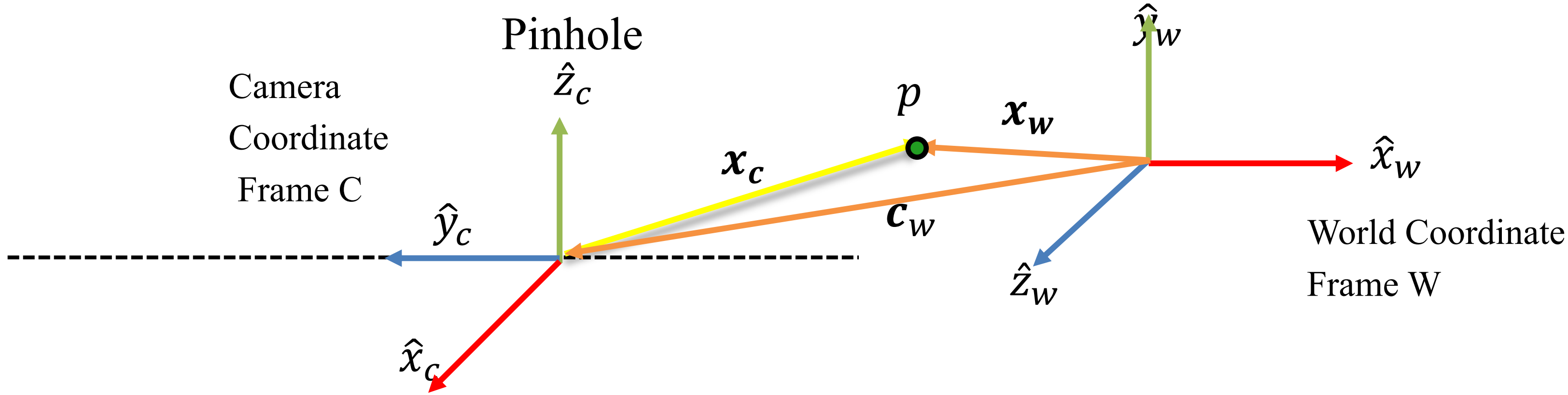
Example: The x-, y- and z-axes of R^3 Euclidean space

Orthonormal Matrix: A square matrix R whose row (or column) vectors are orthonormal. For such a matrix:

$$R^{-1} = R^T \qquad R^T R = R R^T = I$$

A Rotation Matrix is an Orthonormal Matrix

World-to-Camera Transformation



Given the **extrinsic parameters** (R, c_w) of the camera, the camera-centric location of the point P in the world coordinate frame is:

$$\boldsymbol{x}_c = R(\boldsymbol{x_w} - \boldsymbol{c}_w) = R\boldsymbol{x}_w - R\boldsymbol{c}_w = R\boldsymbol{x}_w + \boldsymbol{t}$$

$$\boxed{\boldsymbol{t} = -R\boldsymbol{c}_w}$$

$$\boldsymbol{x_c} = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

Extrinsic Matrix

Rewriting using homogenous coordinates:

$$\boxed{x_c = \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}}$$

Extrinsic Matrix:

$$M_{ext} = \begin{bmatrix} R_{3\times3} & t \\ 0_{1\times3} & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\boxed{\widetilde{x_c} = M_{\text{ext}} \, \widetilde{x_w}}$$

Forward Imaging Model: 3D to 2D

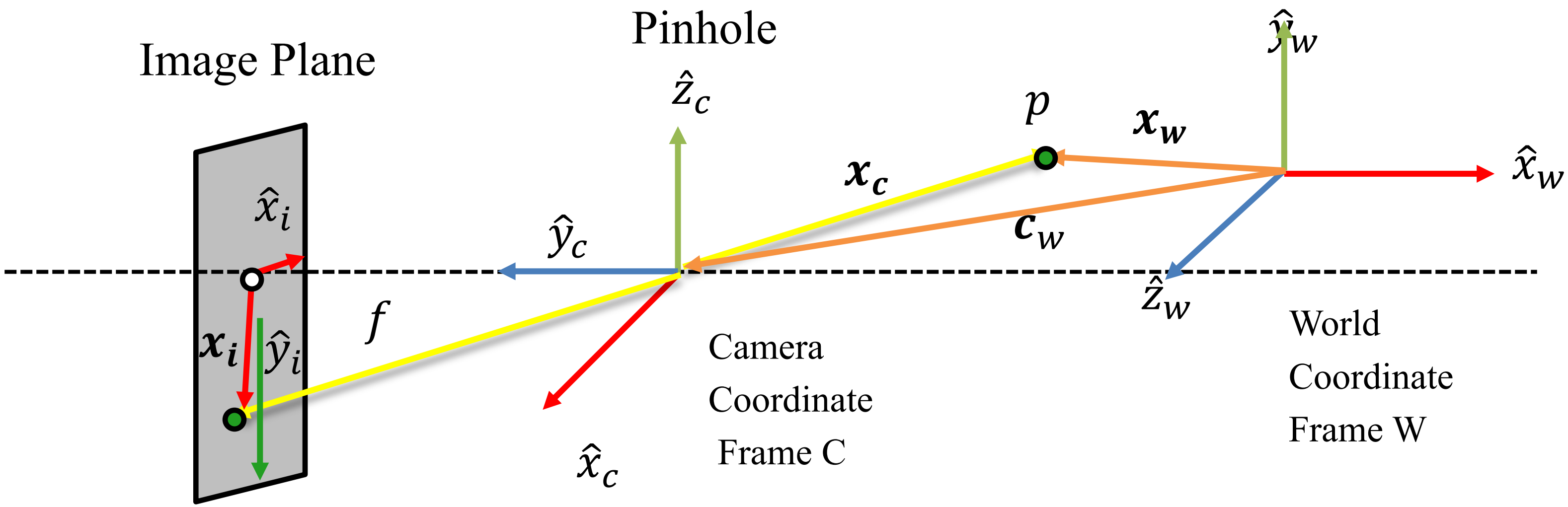


Image Coordinates		Camera Coordinates		World Coordinates
$\boldsymbol{x}_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}$	$\xleftarrow{\boldsymbol{M}_{int}}$	$\boldsymbol{x}_c = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$	$\xleftarrow{\boldsymbol{M}_{\mathbf{ext}}}$	$\boldsymbol{x}_w = \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix}$
	Perspective Projection		Coordinate Transformation	

Projection Matrix P

Camera to Pixel

$$\begin{bmatrix}\tilde{u}\\ \tilde{v}\\ \tilde{w}\end{bmatrix} = \begin{bmatrix} f_x & 0 & o_x & 0\\ 0 & f_y & o_y & 0\\ 0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}x_c\\ y_c\\ z_c\\ 1\end{bmatrix}$$

$$\tilde{u} = \boldsymbol{M}_{int}\widetilde{x_c}$$

World to Camera

$$\begin{bmatrix}x_c\\ y_c\\ z_c\\ 1\end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x\\ r_{21} & r_{22} & r_{23} & t_y\\ r_{31} & r_{32} & r_{33} & t_2\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}x_w\\ y_w\\ z_w\\ 1\end{bmatrix}$$

$$\widetilde{x_c} = \boldsymbol{M}_{\mathbf{ext}}\widetilde{x_w}$$

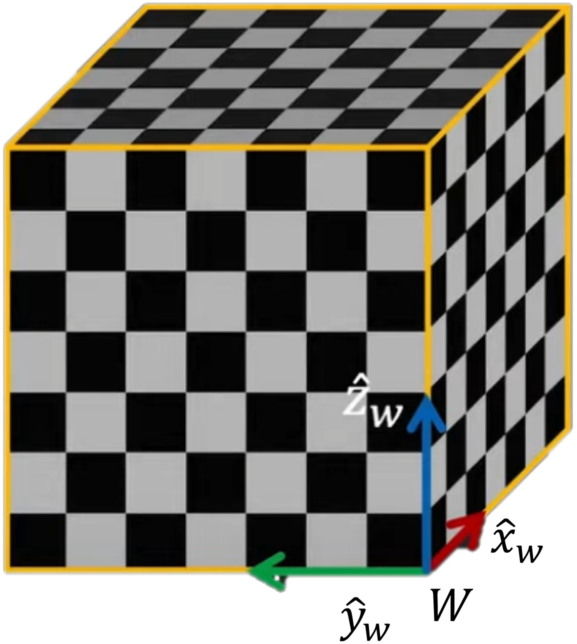
Combining the above two equations, we get the full **projection matrix P**:

$$\tilde{u} = M_{int}M_{\text{ext}}\,\widetilde{x_w} = P\widetilde{x_w}$$

$$\begin{bmatrix}\tilde{u}\\ \tilde{v}\\ \tilde{w}\end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14}\\ p_{21} & p_{22} & p_{23} & p_{24}\\ p_{31} & p_{32} & p_{33} & p_{34}\end{bmatrix}\begin{bmatrix}x_w\\ y_w\\ z_w\\ 1\end{bmatrix}$$

Camera Calibration Procedure

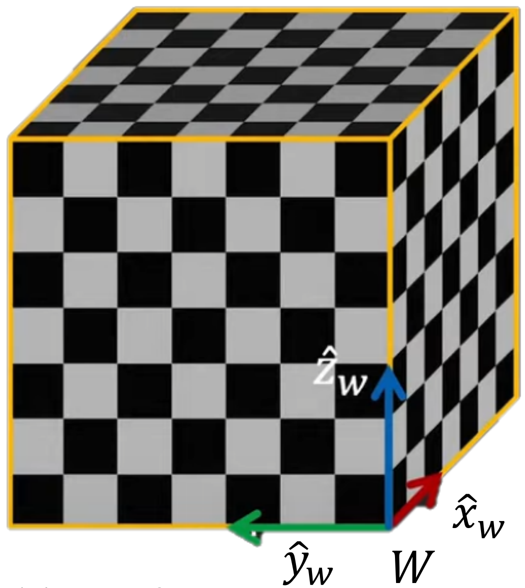
Step1: Capture an image of an object with known geometry.



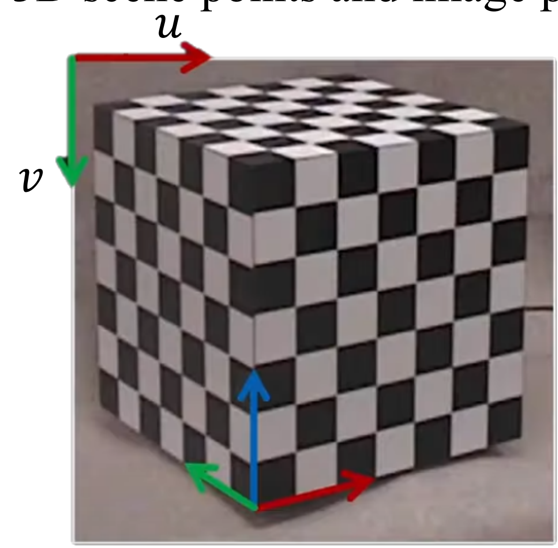
Object of Known Geometry

Camera Calibration Procedure

Step 2: Identify correspondences between 3D scene points and image points.



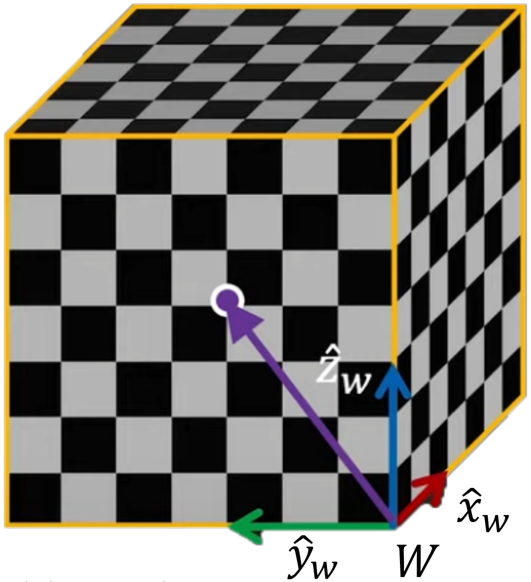
Object of Known Geometry



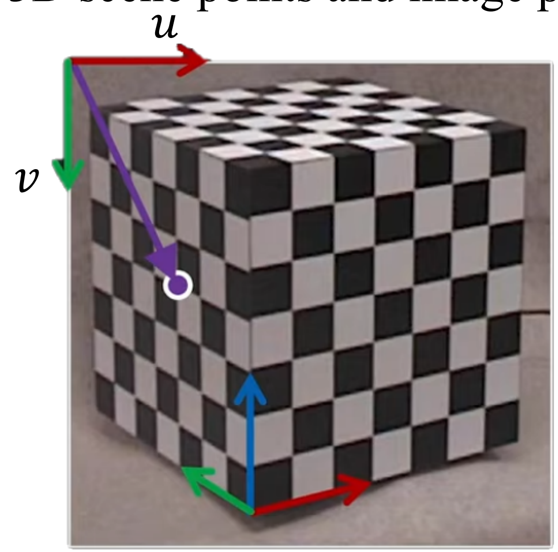
Captured Image

Camera Calibration Procedure

Step 2: Identify correspondences between 3D scene points and image points.



Object of Known Geometry



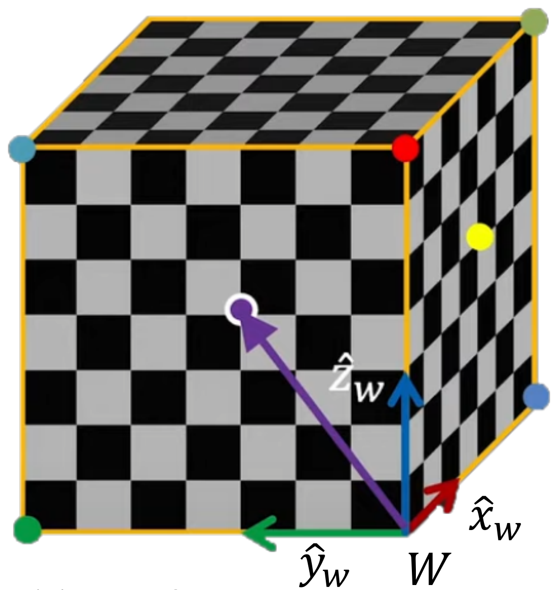
Captured Image

- $x_w = \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \\ 4 \end{bmatrix}$ (inches)

- $u = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} 56 \\ 115 \end{bmatrix}$ (pixels)

Camera Calibration Procedure

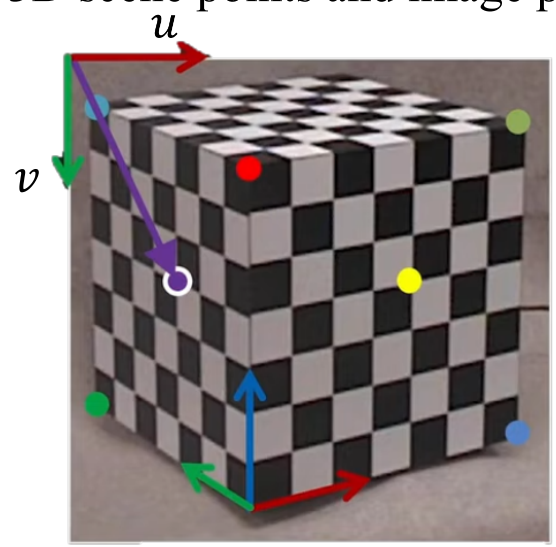
Step 2: Identify correspondences between 3D scene points and image points.



Object of Known Geometry

$$\bullet \quad x_w = \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \\ 4 \end{bmatrix}$$

(inches)



Captured Image

$$\bullet \quad u = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} 56 \\ 115 \end{bmatrix}$$

(pixels)

Camera Calibration Procedure

Step 3: For each corresponding point i in scene and image:

$$\underbrace{\begin{bmatrix} u^{(i)} \\ v^{(i)} \\ 1 \end{bmatrix}}_{\text{Known}} \equiv \underbrace{\begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}}_{\text{Unknown}} \underbrace{\begin{bmatrix} x_w^{(i)} \\ y_w^{(i)} \\ z_w^{(i)} \\ 1 \end{bmatrix}}_{\text{Known}}$$

Expanding the matrix as linear equations:

$$u^{(i)} = \frac{p_{11}x_w^{(i)} + p_{12}y_w^{(i)} + p_{13}z_w^{(i)} + p_{14}}{p_{31}x_w^{(i)} + p_{32}y_w^{(i)} + p_{33}z_w^{(i)} + p_{34}}$$

$$v^{(i)} = \frac{p_{21}x_w^{(i)} + p_{22}y_w^{(i)} + p_{23}z_w^{(i)} + p_{24}}{p_{31}x_w^{(i)} + p_{32}y_w^{(i)} + p_{33}z_w^{(i)} + p_{34}}$$

Camera Calibration Procedure

Step4: Rearranging the terms

$$\begin{bmatrix} x_w^{(1)} & y_w^{(1)} & z_w^{(1)} & 1 & 0 & 0 & 0 & 0 & -u_1 x_w^{(1)} & -u_1 y_w^{(1)} & -u_1 z_w^{(1)} & -u_1 \\ 0 & 0 & 0 & 0 & x_w^{(1)} & y_w^{(1)} & z_w^{(1)} & 1 & -v_1 x_w^{(1)} & -v_1 y_w^{(1)} & -v_1 z_w^{(1)} & -v_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_w^{(i)} & y_w^{(i)} & z_w^{(i)} & 1 & 0 & 0 & 0 & 0 & -u_i x_w^{(i)} & -u_i y_w^{(i)} & -u_i z_w^{(i)} & -u_i \\ 0 & 0 & 0 & 0 & x_w^{(i)} & y_w^{(i)} & z_w^{(i)} & 1 & -v_i x_w^{(i)} & -v_i y_w^{(i)} & -v_i z_w^{(i)} & -v_i \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_w^{(n)} & y_w^{(n)} & z_w^{(n)} & 1 & 0 & 0 & 0 & 0 & -u_n x_w^{(n)} & -u_n y_w^{(n)} & -u_n z_w^{(n)} & -u_n \\ 0 & 0 & 0 & 0 & x_w^{(n)} & y_w^{(n)} & z_w^{(n)} & 1 & -v_n x_w^{(n)} & -v_n y_w^{(n)} & -v_n z_w^{(n)} & -v_n \end{bmatrix} \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \\ p_{31} \\ p_{32} \\ p_{33} \\ p_{34} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

A

Known

$\boldsymbol{p}$

Unknown

Step5: Solve for P

$$\boxed{A\,\boldsymbol{p} = 0}$$

Scale of Projection Matrix

Projection matrix acts on homogenous coordinates.

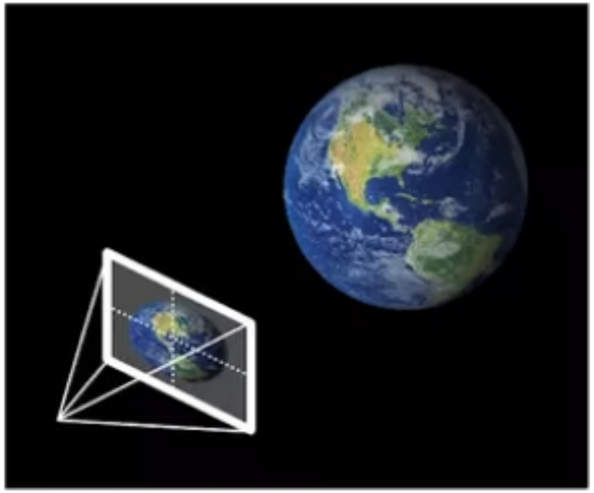
We know that: $\begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} \equiv k \begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix}$ (k ≠0 is any constant)

That is: $$\begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \equiv k \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

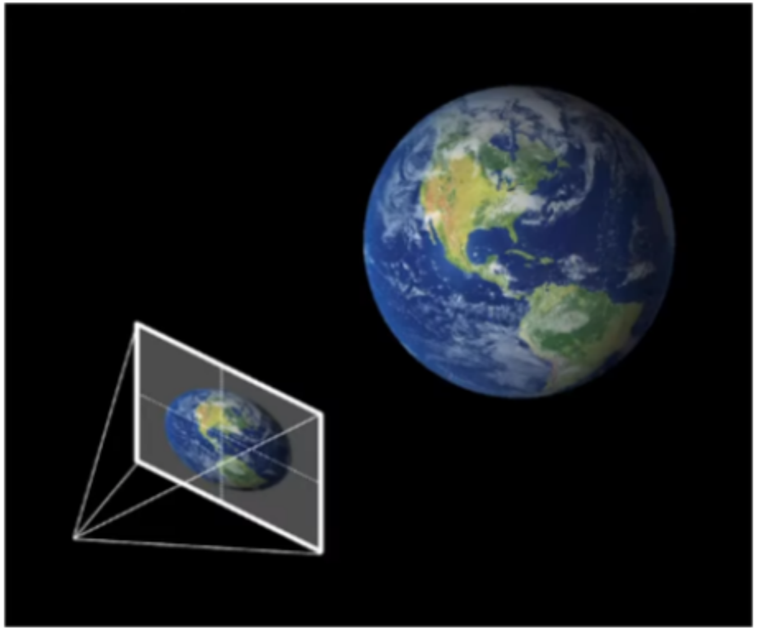
Therefore, Projection Matrices P and kP produce the same homogenous pixel coordinates.

Projection Matrix P is defined only up to a scale.

Scale of Projection Matrix



Scale = k_1



Scale = k_2

Scaling projection matrix, implies simultaneously scaling the world and camera, which does not change the image.

Set projection matrix to some arbitrary scale!

Least squares Solution for P

Option 1: Set scale so that: $p_{34} = 1$

Option 2: Set scale so that: $\boxed{||p||^2 = 1}$

We want Ap as close to 0 as possible and $||p||^2 = 1$:

$$\min_{\mathrm{p}} \ ||\mathrm{Ap}||^2 \text{ such that } ||\mathrm{p}||^2 = 1$$

$$\min_{p}(p^T A^T A p) \ \text{ such that } \ p^T p = 1$$

Define **Loss function** $L(\boldsymbol{p}, \lambda)$:

$$L(\boldsymbol{p}, \lambda) = \boldsymbol{p}^{\boldsymbol{T}} A^T A \boldsymbol{p} - \lambda(\boldsymbol{p}^{\boldsymbol{T}} \boldsymbol{p} - 1)$$

(Similar to Solving Homography in Image Stitching)

Constrained Least Squares Solution

Taking derivatives of $L(\boldsymbol{p}, \lambda)$ w.r.t $\boldsymbol{p}$: $2A^TA\boldsymbol{p} - 2\lambda\boldsymbol{p} = 0$

$$\boxed{A^TA\boldsymbol{p} = \lambda\boldsymbol{p}}$$ **Eigenvalue Problem**

Eigenvector $\boldsymbol{p}$ with **smallest eigenvalue** λ of matrix A^TA minimizes the loss function $L(\boldsymbol{p})$.

Rearrange solution $\boldsymbol{p}$ to form the projection matrix $\boldsymbol{P}$.

Extracting Intrinsic/Extrinsic Parameters

We know that:

$$P = \begin{bmatrix} \boxed{\begin{matrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{matrix}} & \begin{matrix} p_{14} \\ p_{24} \\ p_{34} \end{matrix} \end{bmatrix} = \underbrace{\begin{bmatrix} f_x & 0 & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{M_{\text{int}}} \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{M_{\text{ext}}}$$

That is:

$$\begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} = \begin{bmatrix} f_x & 0 & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \boxed{\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}} = KR$$

Given that K is an **Upper Right Triangular** matrix and R is an **Orthonormal** matrix, it is possible to uniquely "**decouple**" K and R from their product using "**QR factorization**".

Extracting Intrinsic/Extrinsic Parameters

We know that:

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & \boxed{p_{14}} \\ p_{21} & p_{22} & p_{23} & \boxed{p_{24}} \\ p_{31} & p_{32} & p_{33} & \boxed{p_{34}} \end{bmatrix} = \underbrace{\begin{bmatrix} f_x & 0 & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{M_{\text{int}}} \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{M_{\text{ext}}}$$

That is:

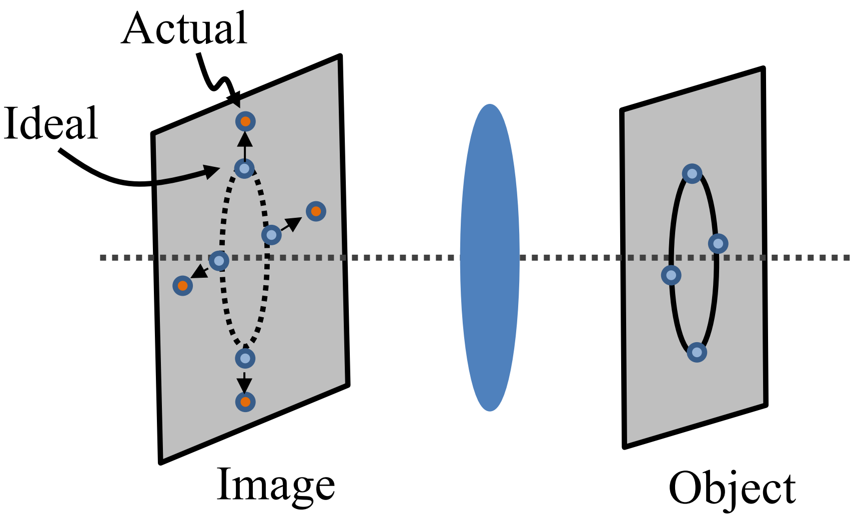
$$\begin{bmatrix} p_{14} \\ p_{24} \\ p_{34} \end{bmatrix} = \begin{bmatrix} f_x & 0 & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} = K\boldsymbol{t}$$

Therefore:

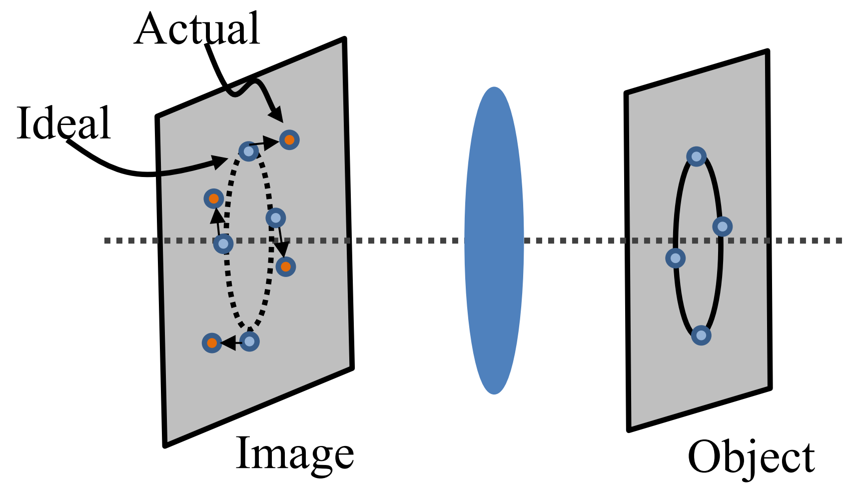
$$\boxed{\boldsymbol{t} = K^{-1} \begin{bmatrix} p_{14} \\ p_{24} \\ p_{34} \end{bmatrix}}$$

Camera Calibration

Pinholes do not exhibit image distortions. But, lenses do!



Tangential Distortion

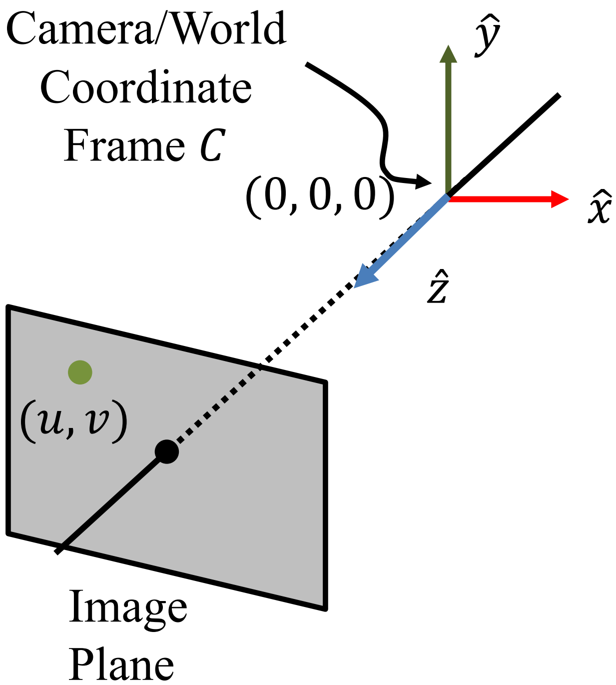


Radial Distortion

The intrinsic model of the camera will need to include the distortion coefficients. We ignore distortions here.

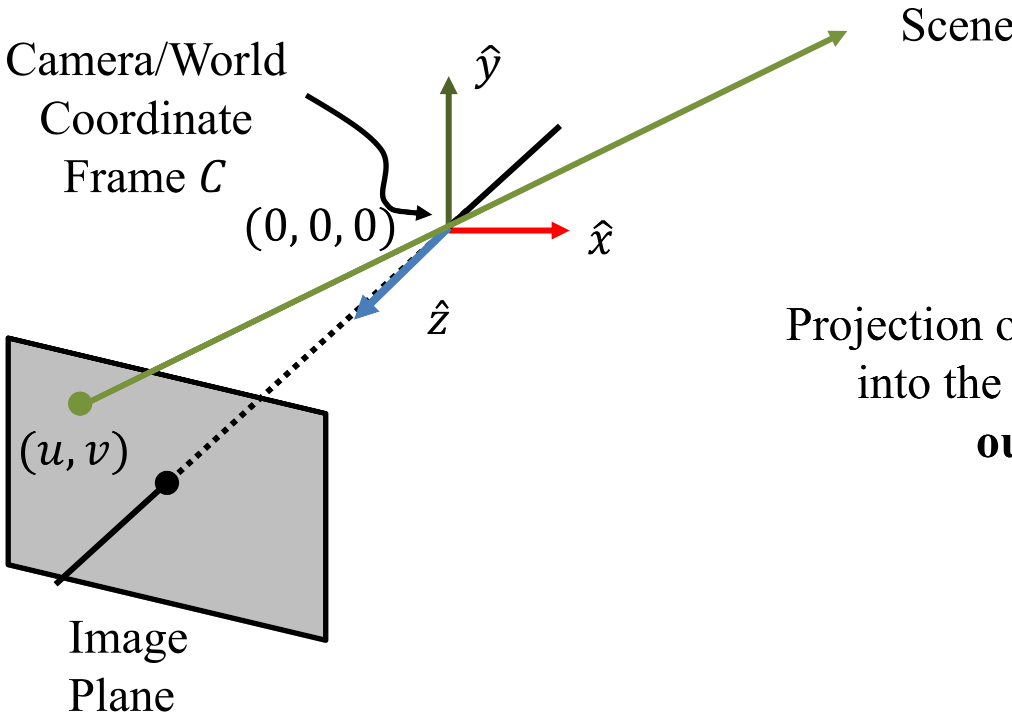
Backward Projection: From 2D to 3D

Given a calibrated camera, can we find the 3D scene point from a single 2D image?



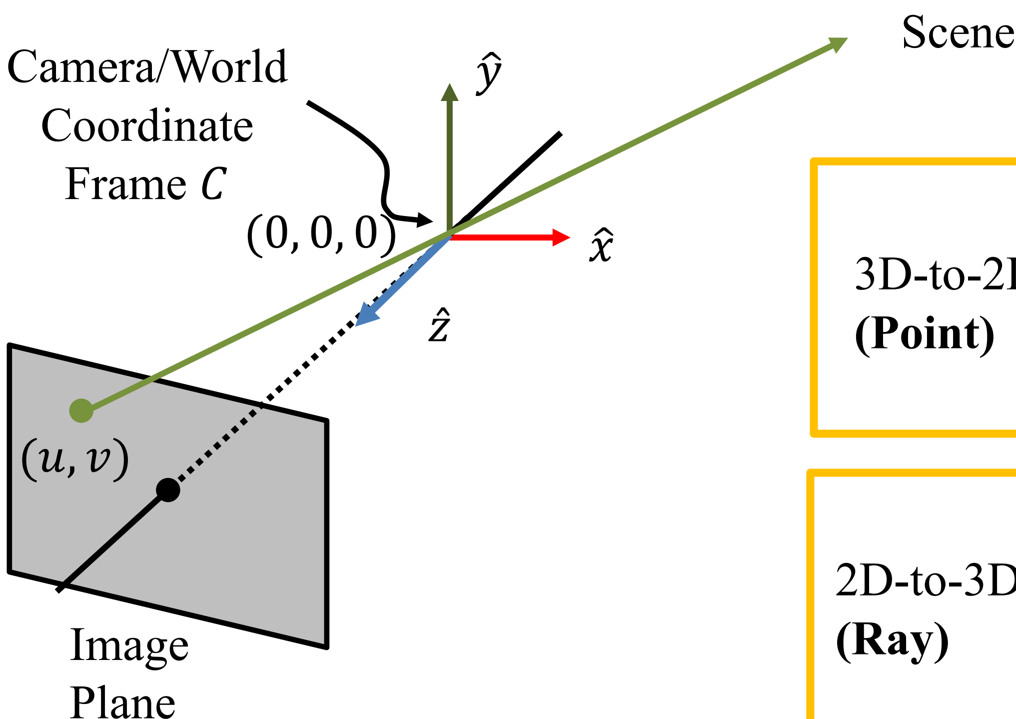
Backward Projection: From 2D to 3D

Given a calibrated camera, can we find the 3D scene point from a single 2D image?



Projection of an image point back into the scene results in an **outgoing ray.**

Computing 2D-to-3D Outgoing Ray



3D-to-2D:
(Point)

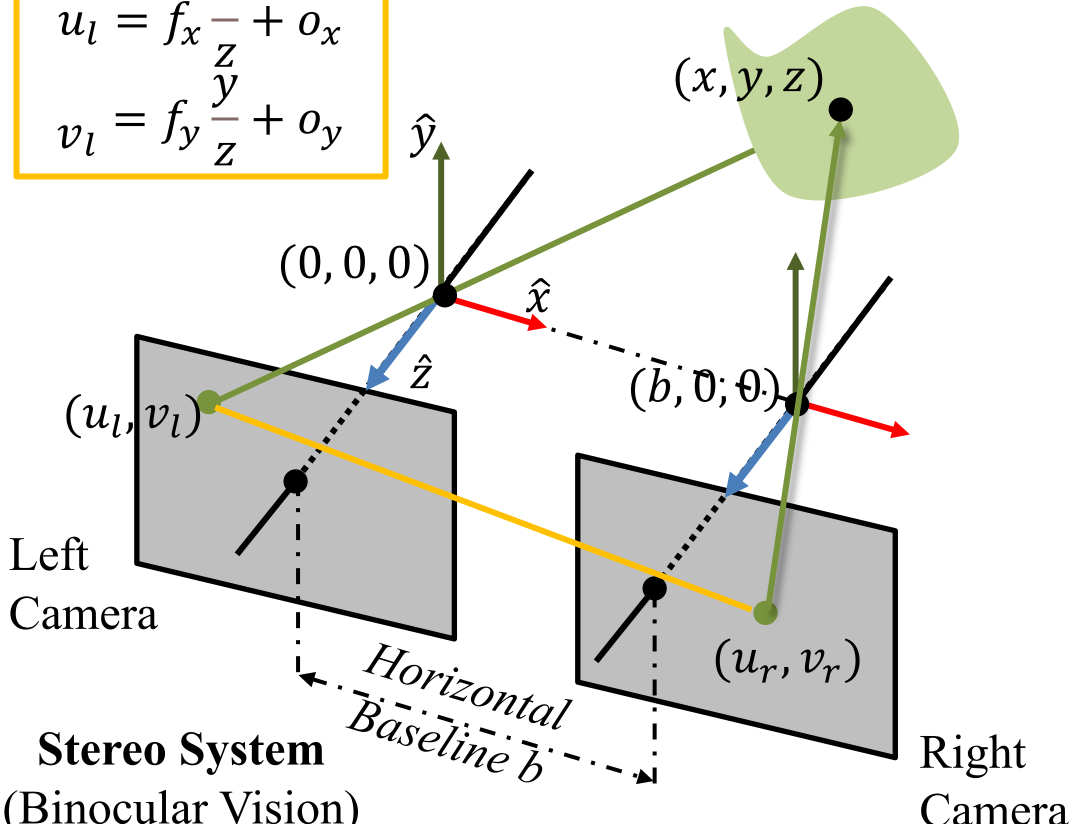
$$u = f_x \frac{x_c}{z_c} + o_x$$
$$v = f_y \frac{y_c}{z_c} + o_y$$

2D-to-3D:
(Ray)

$$\mathrm{x} = \mathrm{z}/f_x(u - o_x)$$
$$\mathrm{y} = z/f_y(v - o_y)$$
$$z > 0$$

Triangulation using Two Cameras

$$u_l = f_x \frac{x}{z} + o_x$$
$$v_l = f_y \frac{y}{z} + o_y$$



Stereo System
(Binocular Vision)

$$u_r = f_x \frac{x - b}{z} + o_x$$
$$v_r = f_y \frac{y}{z} + o_y$$

f_x, f_y, b, o_x, o_y are known.

Simple Stereo: Depth and Disparity

From perspective projection:

$$(u_l, v_l) = \left(f_x \frac{x}{z} + o_x, f_y \frac{y}{z} + o_y\right) \quad (u_r, v_r) = \left(f_x \frac{x-b}{z} + o_x, f_y \frac{y}{z} + o_y\right)$$

Solving for (x, y, z):

$$x = \frac{b(u_l - o_x)}{(u_l - u_r)} \quad y = \frac{b f_x (v_l - o_y)}{f_y (u_l - u_r)} \quad \boxed{z = \frac{b f_x}{(u_l - u_r)}}$$

where $(u_l - u_r)$ is called **Disparity**.

Depth z is inversely proportional to Disparity.

Disparity is proportional to Baseline.

A Simple Stereo Camera

Fujifilm FinePix REAL 3D W3

Stereo Matching: Finding Disparities

Goal: Find the disparity between left and right stereo pairs.

Left/Right Camera Images

Disparity Map(Ground Truth)

From perspective projection: $v_l = v_r = f_y \frac{y}{z} + o_y$

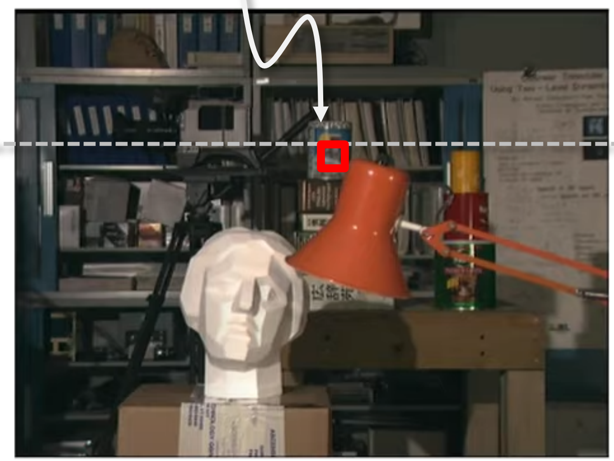
Corresponding scene points lie on the same horizontal scan line.

Window Based Methods

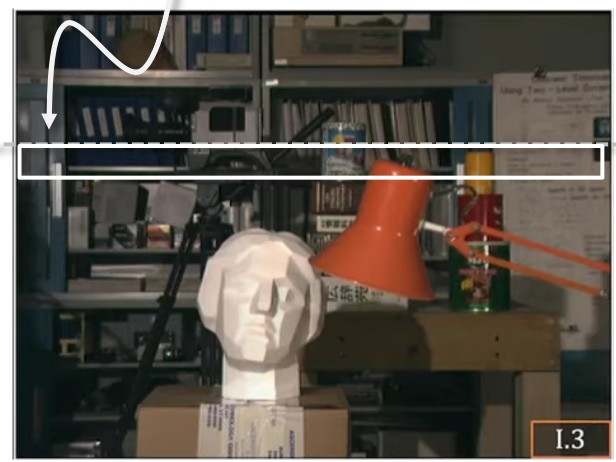
Determine Disparity using **Template Matching**

Template Window T

Search Scan Line L



Left Camera Image E_l



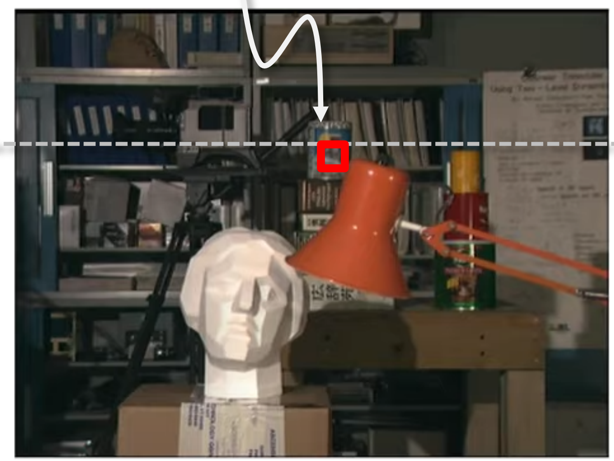
Right Camera Image E_r

Window Based Methods

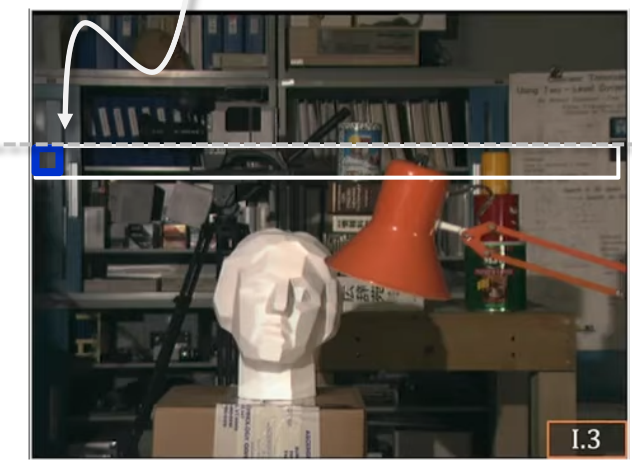
Determine Disparity using **Template Matching**

Template Window T

Search Scan Line L



Left Camera Image E_l



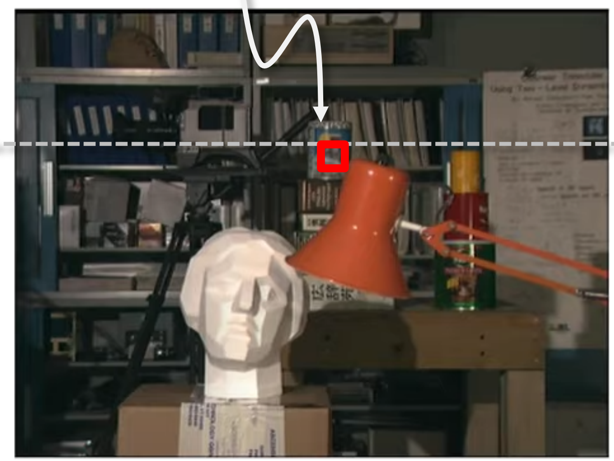
Right Camera Image E_r

Window Based Methods

Determine Disparity using **Template Matching**

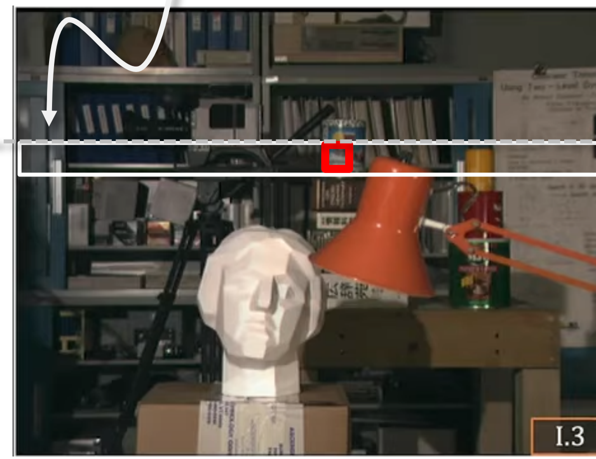
Template Window T

Search Scan Line L



Left Camera Image E_l

Disparity: $d = u_l - u_r$



Right Camera Image E_r

Depth: $z = \dfrac{b f_x}{(u_l - u_r)}$

Similarity Metrics for Template Matching

Find pixel $(k, l) \in L$ with Minimum **Sum of Absolute Differences:**

$$SAD(k,l) = \sum_{(i,j)\in T} |E_l(i,j) - E_r(i+k, j+l)|$$

Find pixel $(k, l) \in L$ with Minimum **Sum of Squared Differences:**

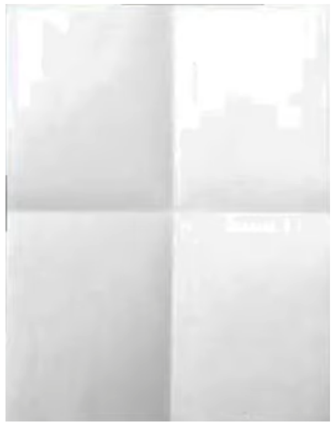
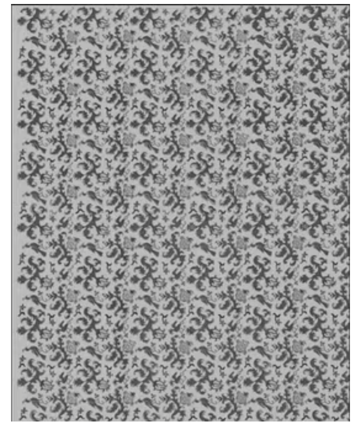
$$SSD(k,l) = \sum_{(i,j)\in T} |E_l(i,j) - E_r(i+k, j+l)|^2$$

Find pixel $(k, l) \in L$ with Maximum **Normalized Cross-Correlation:**

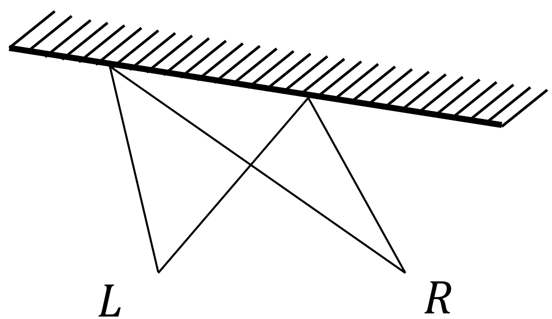
$$NCC(k,l) = \frac{\sum_{(i,j)\in T} E_l(i,j) E_r(i+k, j+l)}{\sqrt{\sum_{(i,j)\in T} E_l(i,j)^2 \sum_{(i,j)\in T} E_r(i+k, j+l)^2}}$$

Issues with Stereo Matching

- Surface must have (non-repetitive) texture

- Foreshortening effect makes matching challenging



How Large Should Window Be?

Window size = 5 pixels
(Sensitive to noise)

Window size = 30 pixels
(Poor localization)

Adaptive Window Method Solution: For each point, match using windows of multiple sizes and use the disparity that is a result of the best similarity measure (minimize SSD per pixel).

Window Based Methods: Results

Left Image

Right Image

Ground Truths

SSD - Adaptive Window

SD (Window size=21)

State of the Art

http://vision.middlebury.edu/stereo