Computer Vision

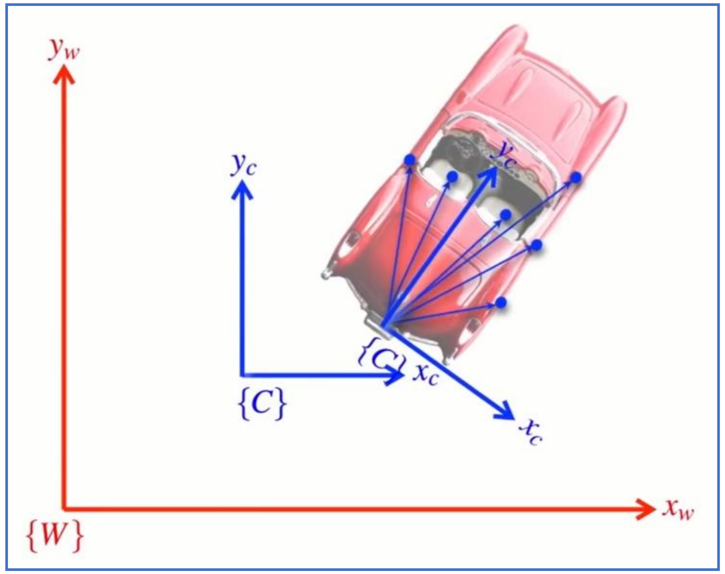
(Course Code: 4047)

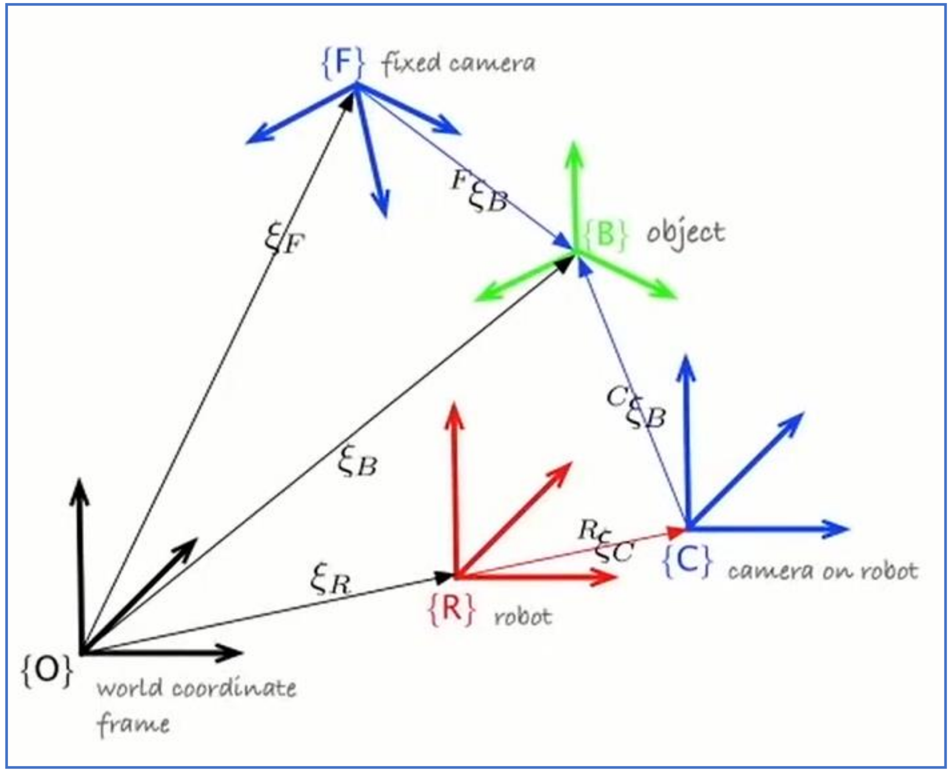
Module-3:Lecture-5: Epipolar Geometry - Stereo

Gundimeda Venugopal, Professor of Practice, SCOPE

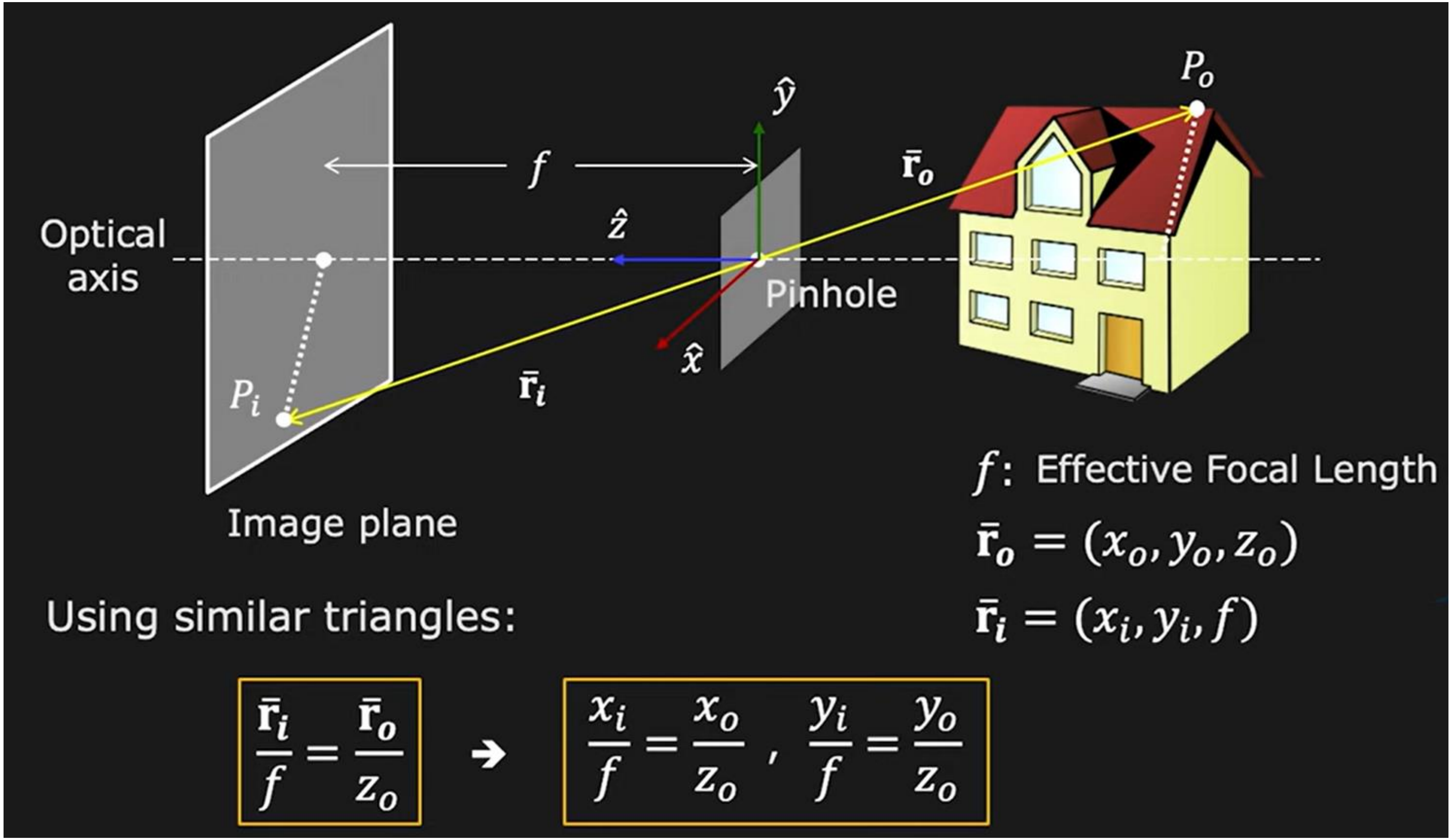
Coordinate Frames

Need for multiple Coordinate Frames





Pinhole Projection of an Image



Camera Calibration

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

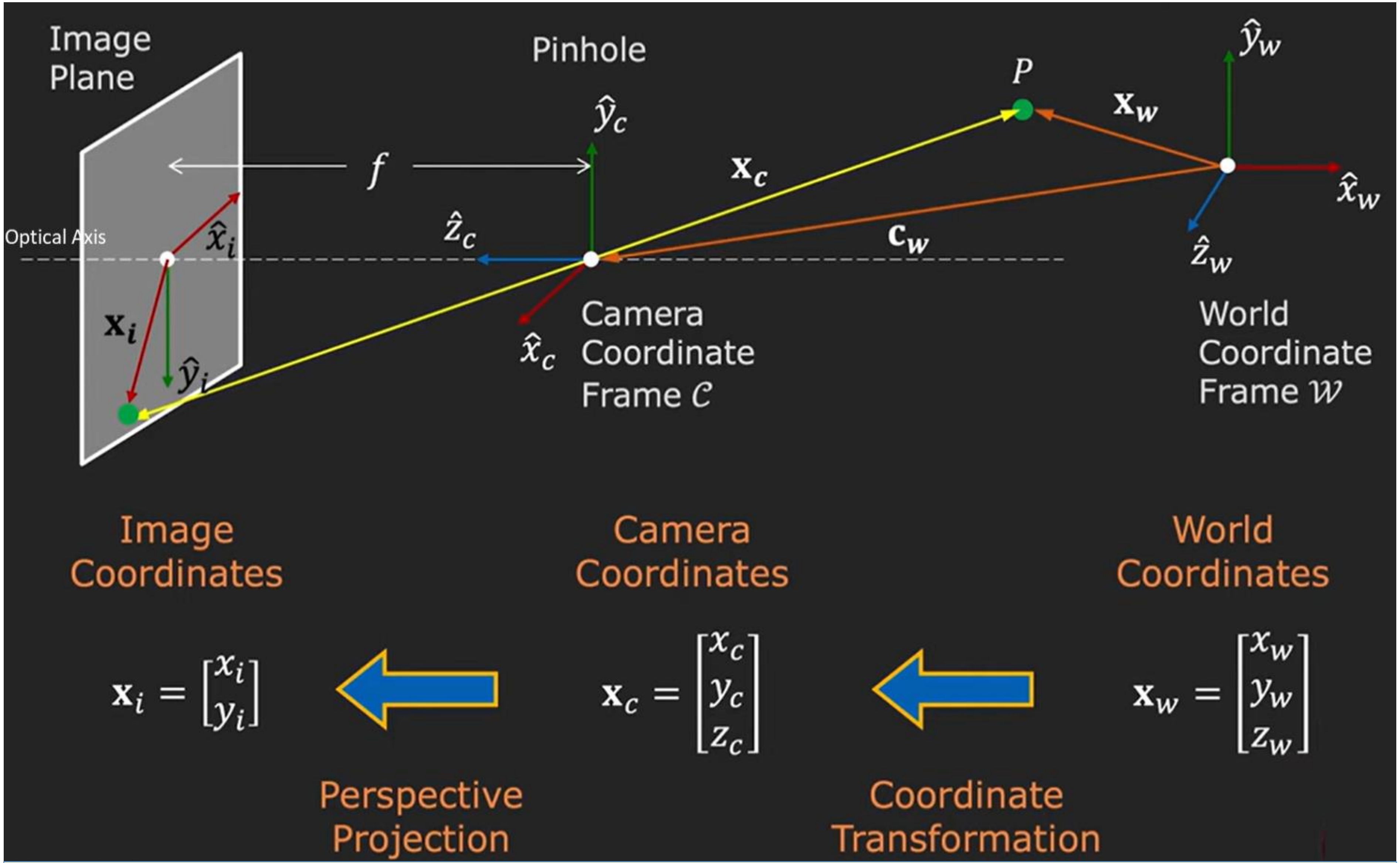
(4) Example Application: Simple Stereo

What is Camera Calibration?

- Key problem: Recover a 3D structure of a real world scene from its Images

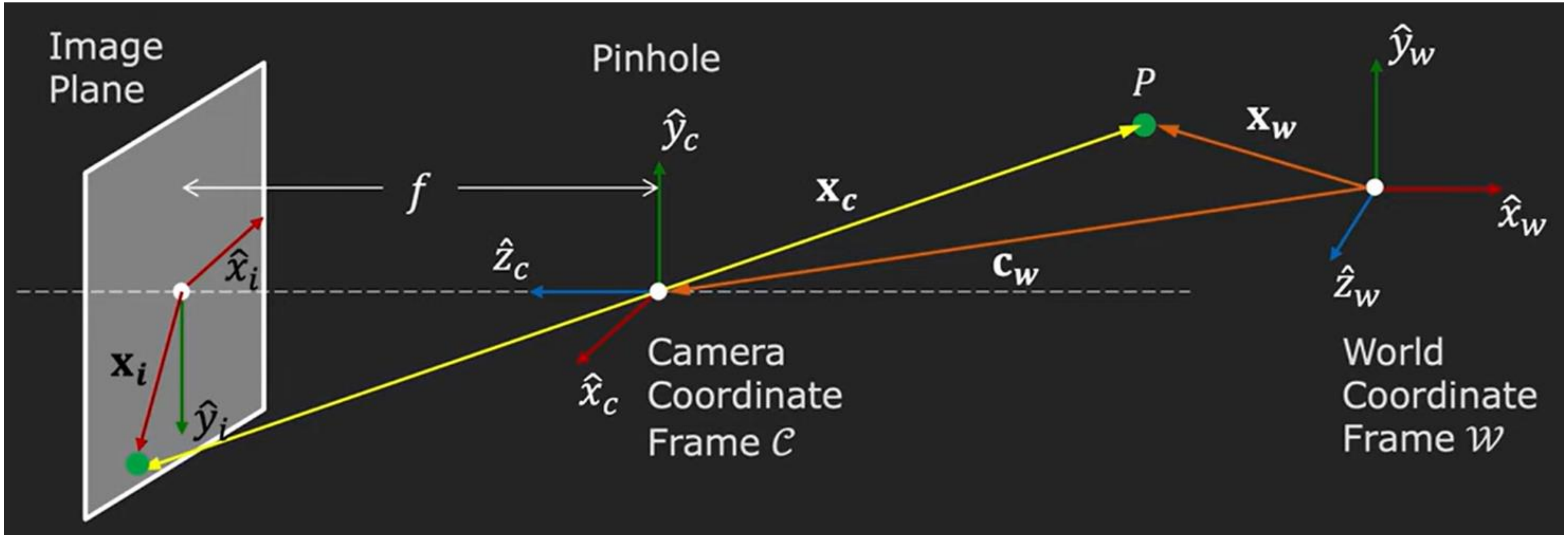
- A scene that is defined in some world coordinate frame has all scene points are measured say in mm
- In the images of the scene, points are measured in terms of pixels.
- To reconstruct , we need:
  - External Parameters: The position and orientation of the camera w.r.t World Coordinate Frame
  - Internal Parameters: How the camera maps the projection points in the world onto its image plane (e.g., Focal length)
- Determining the External and Internal parameters of the camera is referred to as Camera Calibration
- Once the camera calibration is complete, we can use the calibrated camera to reconstruct a 3D Scene

Forward Imaging Model: 3D to 2D



$X_i = X_{image} = (x_i\ ,\ y_i)$ $X_{camera} = X_c = (x_c\ ,\ y_c\ ,\ z_c)$ $X_{World} = X_W = (x_w\ ,\ y_w\ ,\ z_w)$

Perspective Projection

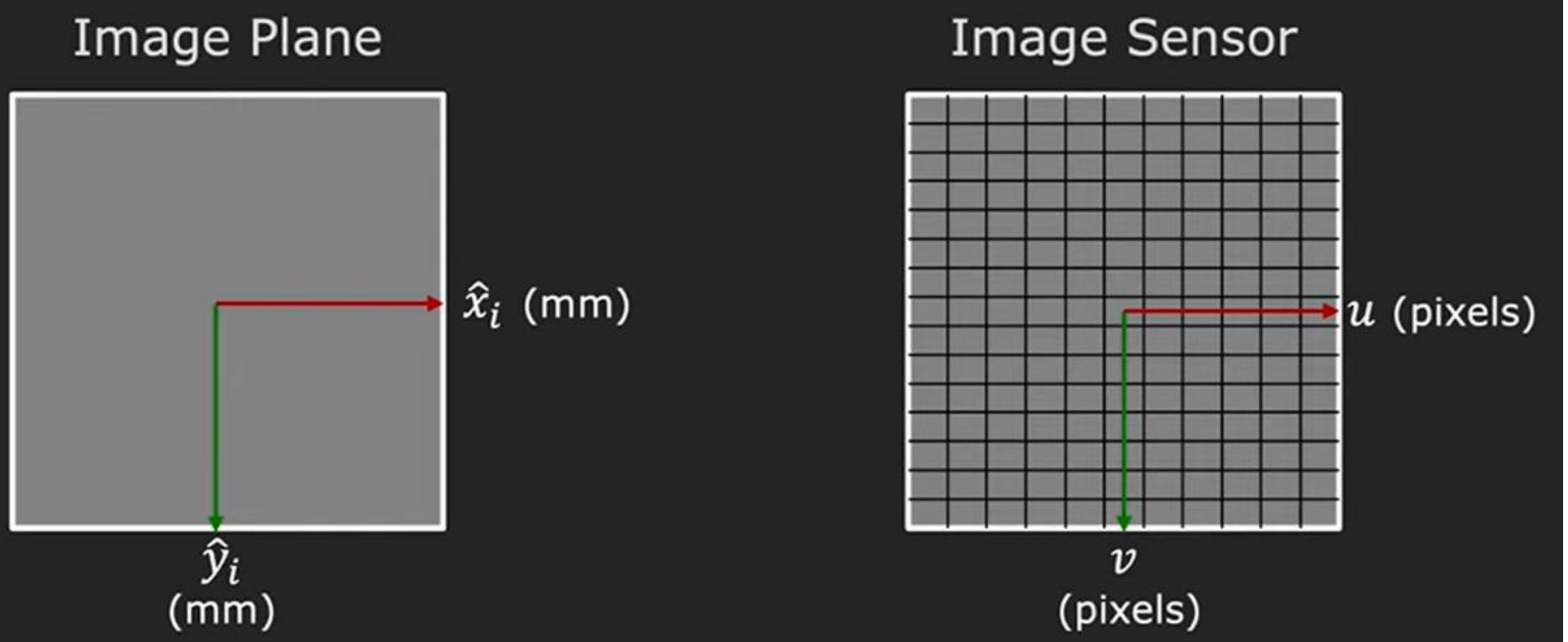


We know that: $\dfrac{x_i}{f} = \dfrac{x_c}{z_c}$ and $\dfrac{y_i}{f} = \dfrac{y_c}{z_c}$

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

$X_i = X_{image} = (x_i\ ,\ y_i)$ $X_{camera} = X_c = (x_c\ ,\ y_c\ ,\ z_c)$ $X_{World} = X_W = (x_w\ ,\ y_w\ ,\ z_w)$

Image Plane to Image Sensor mapping

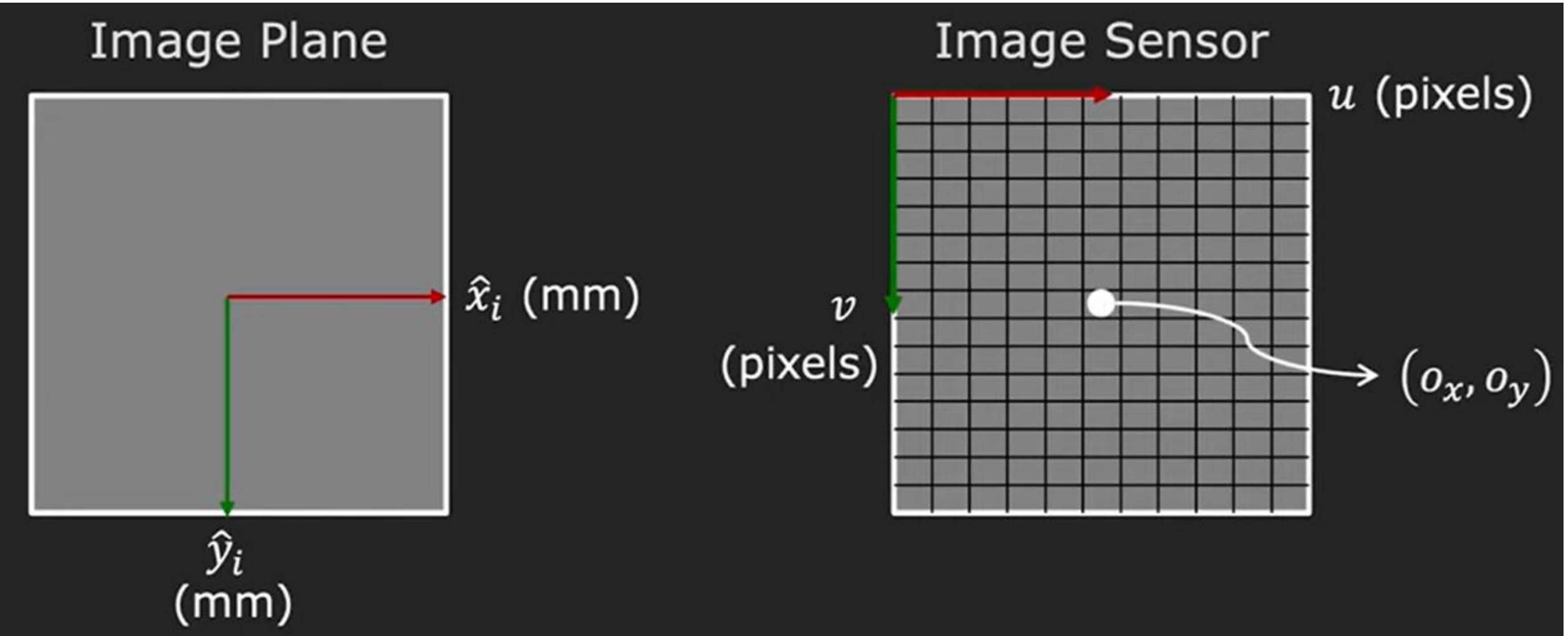


Pixels may be rectangular.

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

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

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

Equations for perspective projection are Non-Linear.

It is convenient to express them as linear equations.

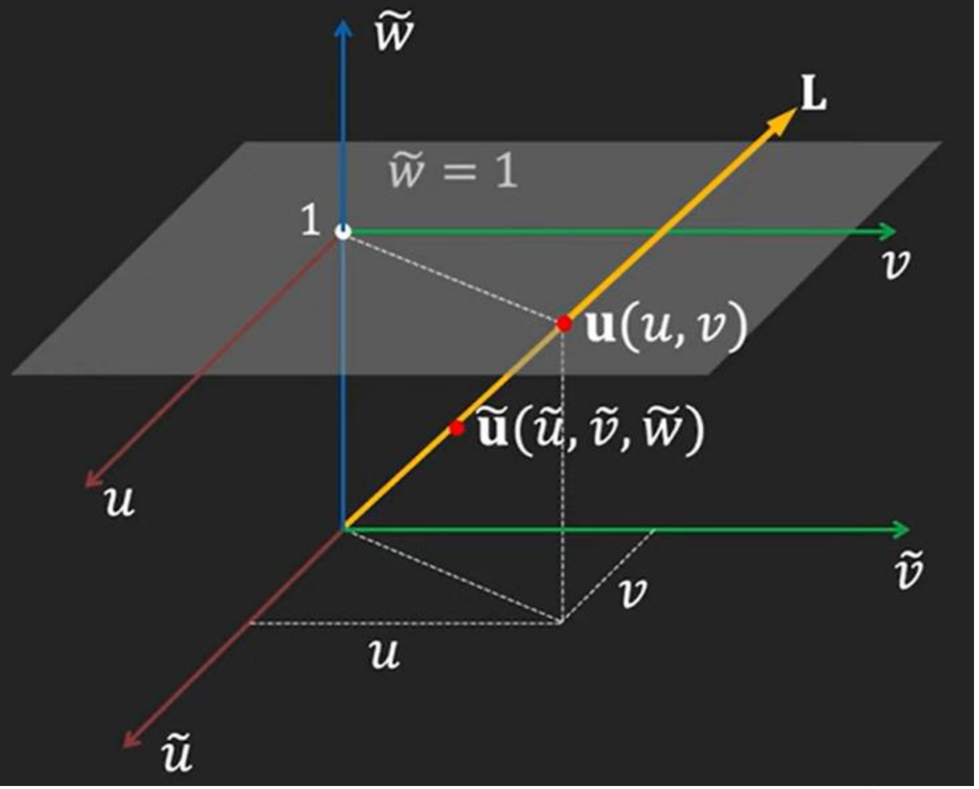
How to solve this problem? Using Homogenous Coordinate System

Homogeneous Coordinates (2D)

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

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

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



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

Homogeneous Coordinates (3D)

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

$$x = \frac{\tilde{x}}{\tilde{w}} \qquad y = \frac{\tilde{y}}{\tilde{w}} \qquad z = \frac{\tilde{z}}{\tilde{w}}$$

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

Perspective Projection

Perspective projection equations:

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

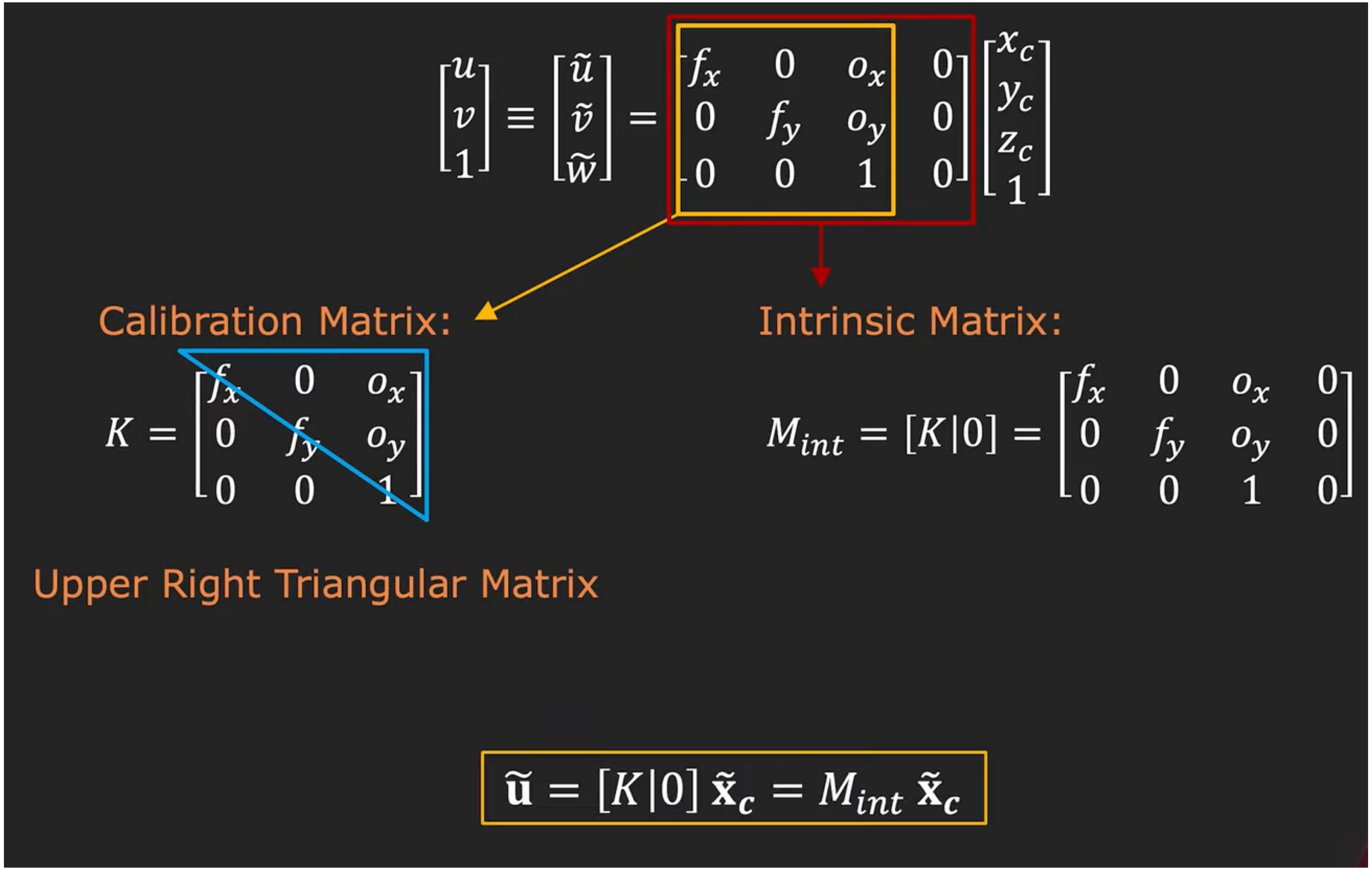
Homogenous coordinates of (u, v):

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

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

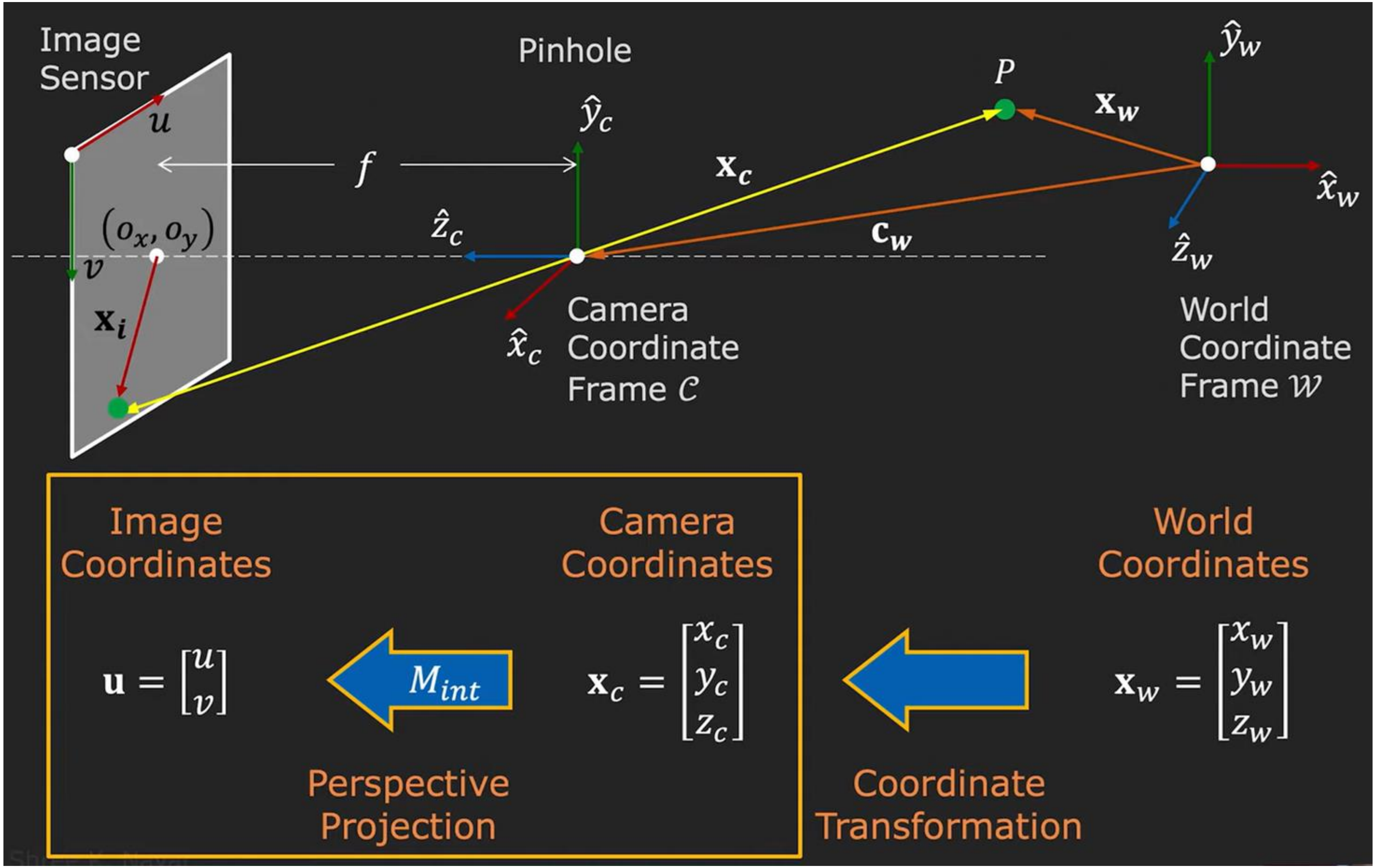
Linear Model for Perspective Projection

Intrinsic Matrix



M_{int} takes you from a homogenous representation of a point in the camera 3D coordinate frame to its 2D pixel coordinates in the image.

Forward Imaging Model: 3D to 2D



Recall: 2D Rotation Matrix

- For the 2D case we described the new axes {B} in terms of the old axes {A}

$$\hat{x}_B = \cos\theta\,\hat{x}_A + \sin\theta\,\hat{y}_A$$

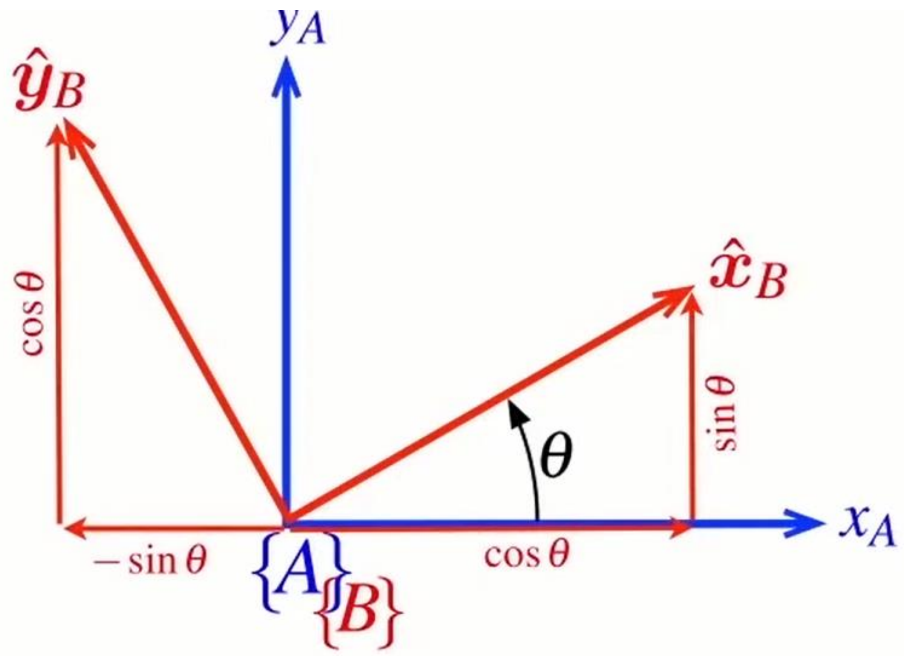
$$\hat{y}_B = -\sin\theta\,\hat{x}_A + \cos\theta\,\hat{y}_A$$

- Which transforms vectors from the new frame {B} to the old frame {A}

new x-axis

new y-axis

$$\begin{pmatrix} {}^{A}x \\ {}^{A}y \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} {}^{B}x \\ {}^{B}y \end{pmatrix}$$

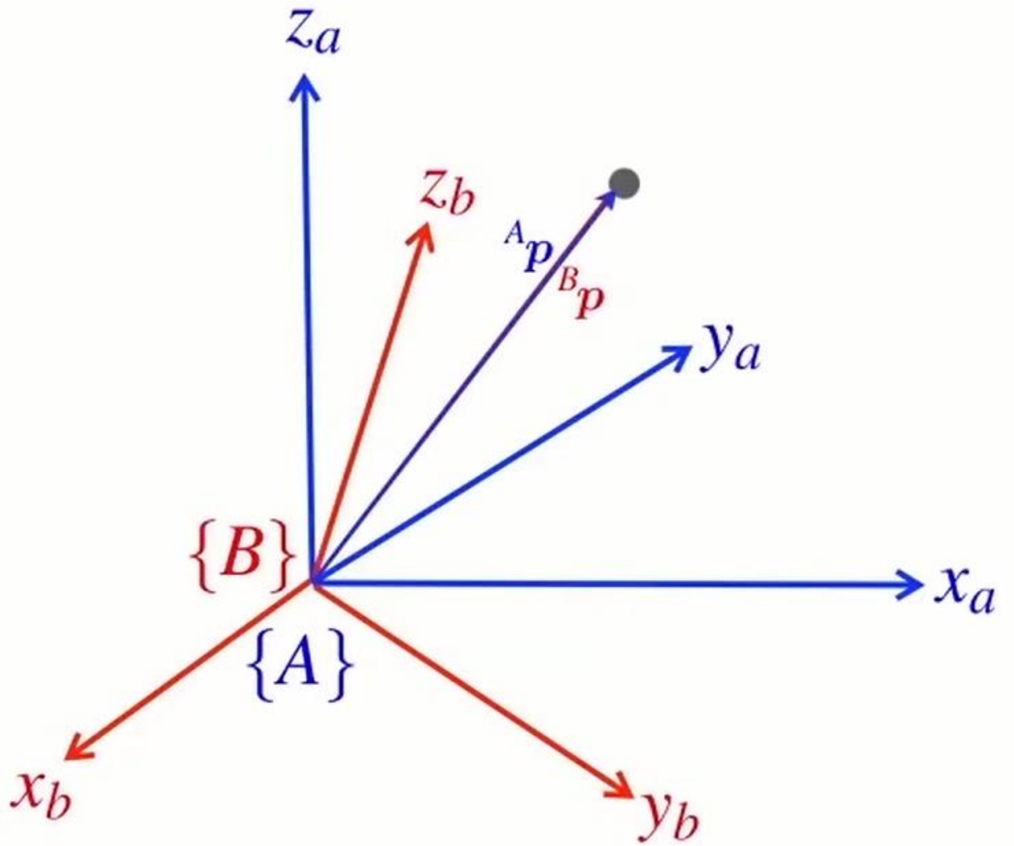


Rotation in 3D

- For the 3D case we also describe the new axes {B} in terms of the old axes {A}

$$\begin{pmatrix} {}^{A}x \\ {}^{A}y \\ {}^{A}z \end{pmatrix} = \begin{array}{c} \begin{array}{ccc} \text{new x-axis} & \text{new y-axis} & \text{new z-axis} \end{array} \\ \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \end{array} \begin{pmatrix} {}^{B}x \\ {}^{B}y \\ {}^{B}z \end{pmatrix}$$

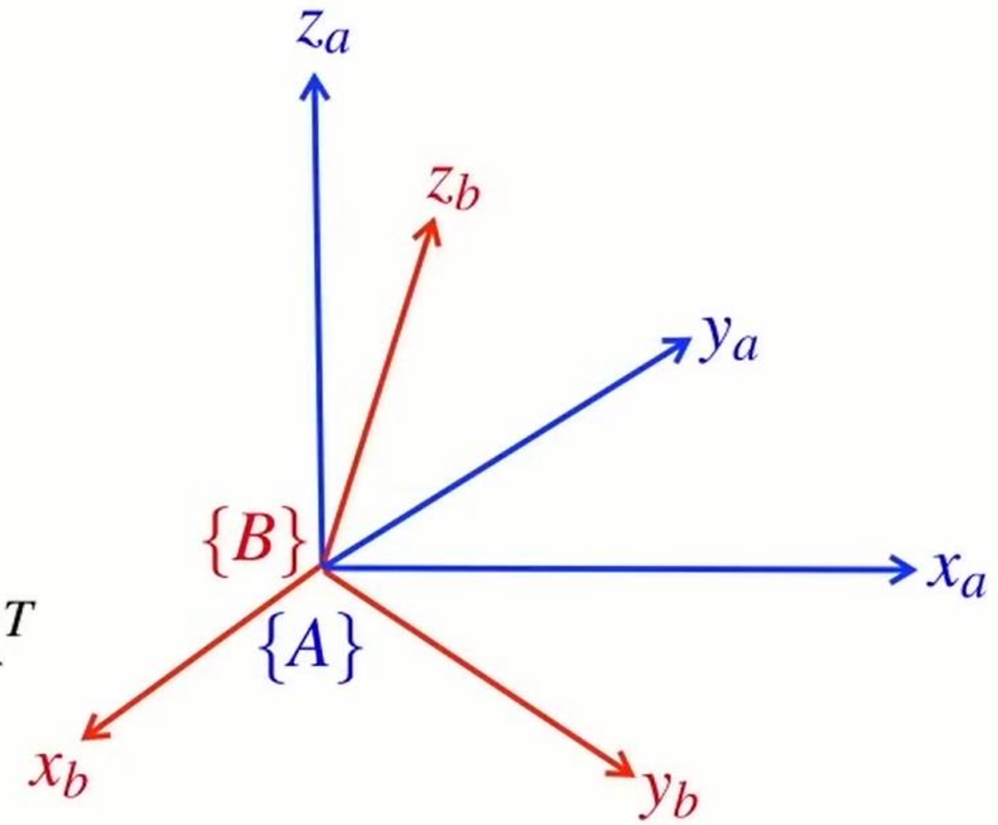
- Which transforms vectors from the new frame {B} to the old frame {A}



Note: Here both the coordinate frames are at the same location.
If they are different locations, we need both rotation and next translation in x, y, z directions

Properties of Rotation Matrix

$$
{}^{A}\mathbf{R}_{B} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}
$$



- An orthogonal (orthonormal) matrix
- Each column is a **unit** length **vector**
- Each column is **orthogonal** to all other columns
- The inverse is the same as the transpose $\mathbf{R}^{-1} = \mathbf{R}^{T}$
- The determinant is +1 $det(\mathbf{R}) = 1$
  - ➡ the length of a vector is unchanged by rotation
- Rotation matrices belong to the Special Orthogonal group of dimension 3 $\mathbf{R} \in SO(3)$

Elementary Rotation matrices

$$\mathbf{R}_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix}$$

$$\mathbf{R}_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}$$

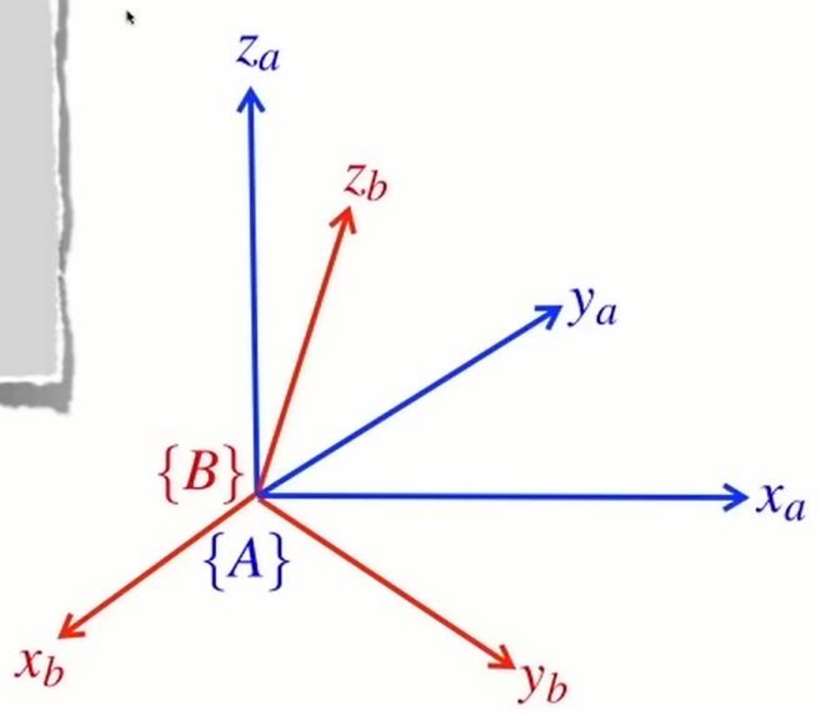
$$\mathbf{R}_z(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{R} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

Euler’s rotation theorem

Euler's rotation theorem

Any two independent orthonormal coordinate frames can be related by a sequence of rotations (not more than 3) about coordinate axes, where no two successive rotations may be about the same axis.

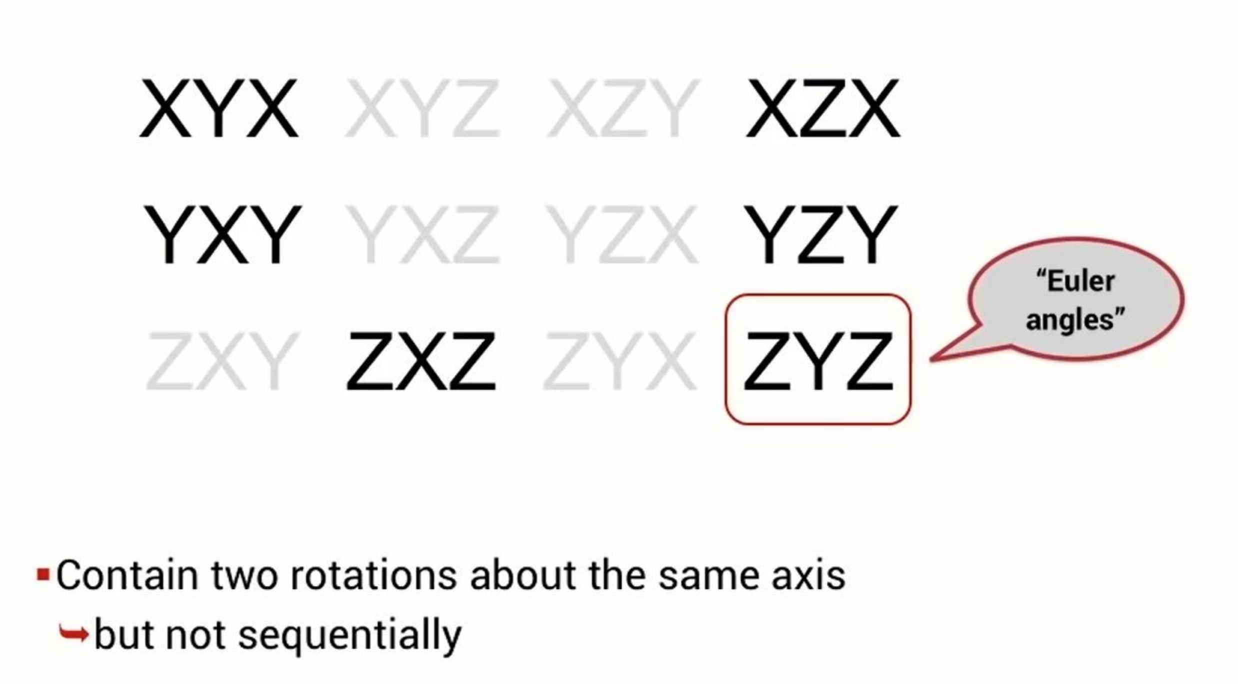


Leonhard Euler
1707–1783

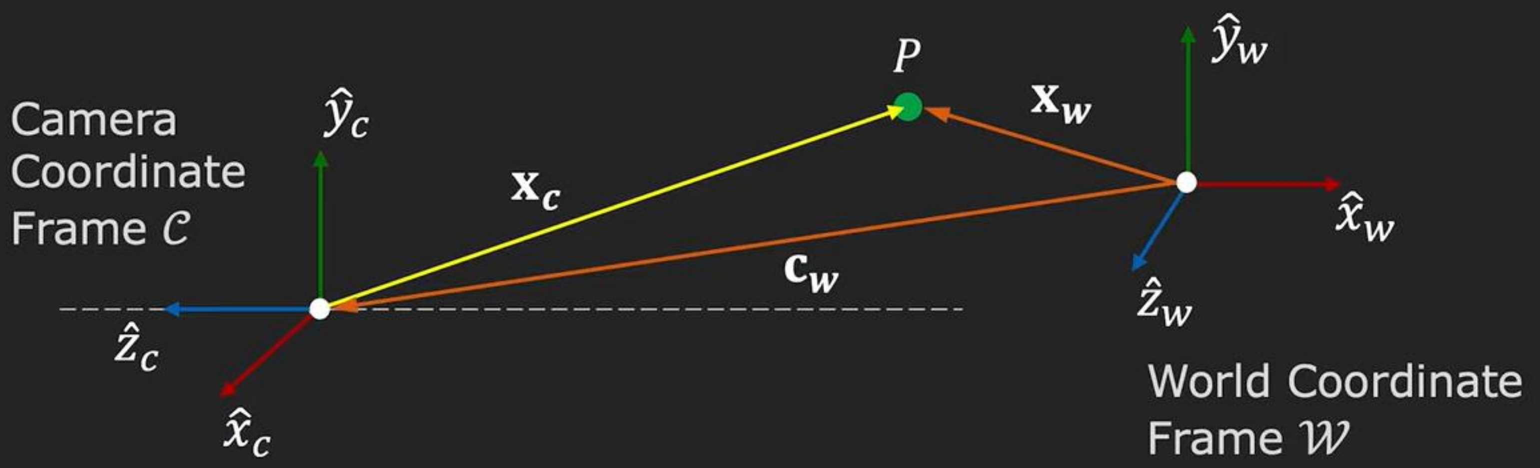
- Swiss mathematician and physicist
- Student of Bernoulli
- Applied calculus to many problems
- His work fills over 70 volumes
- Half produced during his last 17 years when he was completely blind

Euler angles

Euler angles are a set of three angles that describe the orientation of a rigid body in a 3D space, relative to a fixed coordinate system:
•**Roll**: The angle of rotation around the x-axis, represented by the symbol ϕ
•**Pitch**: The angle of rotation around the y-axis, represented by the symbol θ
•**Yaw**: The angle of rotation around the z-axis, represented by the symbol ψ



Extrinsic Parameters



Position $\mathbf{c}_w$ and Orientation R of the camera in the world coordinate frame $\mathcal{W}$ are the camera's Extrinsic Parameters.

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{array}{l} \rightarrow \text{Row 1: Direction of } \hat{x}_c \text{ in world coordinate frame} \\ \rightarrow \text{Row 2: Direction of } \hat{y}_c \text{ in world coordinate frame} \\ \rightarrow \text{Row 3: Direction of } \hat{z}_c \text{ in world coordinate frame} \end{array}$$

Orthonormal Vectors and Matrices

Orthonormal Vectors: Two vectors $\mathbf{u}$ and $\mathbf{v}$ are orthonormal if and only if:

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

(Orthogonality) (Unit length)

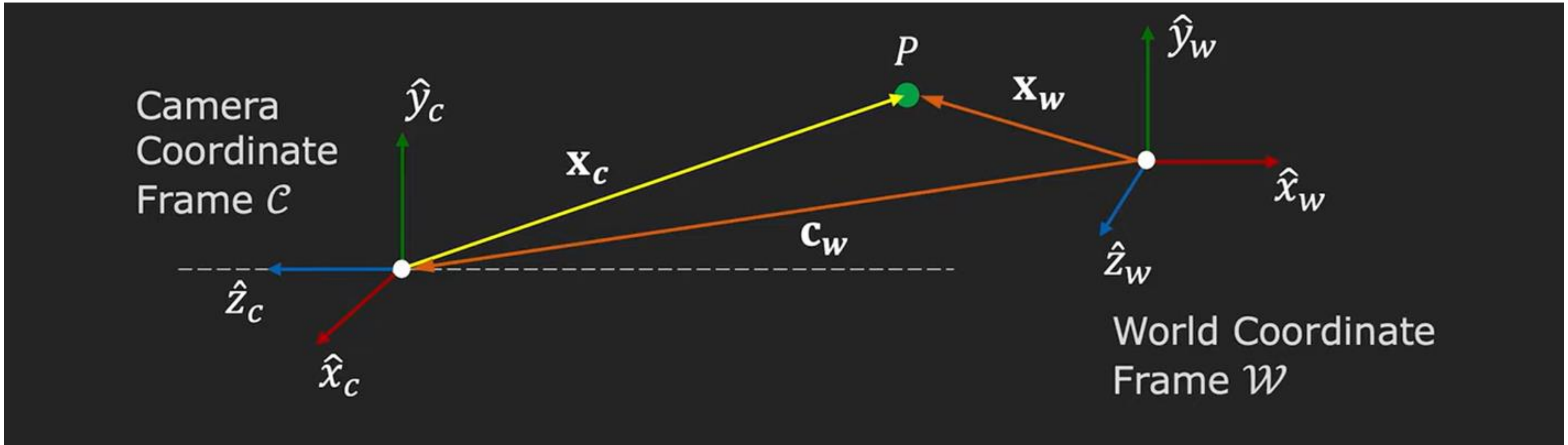
Example: The x-, y- and z-axes of $\mathbb{R}^3$ Euclidean space

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

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

A Rotation Matrix is an Orthonormal Matrix

World-to-Camera Transformation



Given the extrinsic parameters $(R, \mathbf{c}_w)$ of the camera, the camera-centric location of the point P in the world coordinate frame is:

$$\mathbf{x}_c = R(\mathbf{x}_w - \mathbf{c}_w) = R\mathbf{x}_w - R\mathbf{c}_w = R\mathbf{x}_w + \mathbf{t} \qquad \boxed{\mathbf{t} = -R\mathbf{c}_w}$$

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

X_w and C_w are in World Coordinate system. X_c in Camera Coordinate system.

Extrinsic Matrix

Rewriting using homogenous coordinates:

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

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

$$\boxed{\tilde{\mathbf{x}}_c = M_{ext}\tilde{\mathbf{x}}_w}$$

Projection Matrix P

Camera to Pixel

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

$$\tilde{\mathbf{u}} = M_{int}\, \tilde{\mathbf{x}}_c$$

World to Camera

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

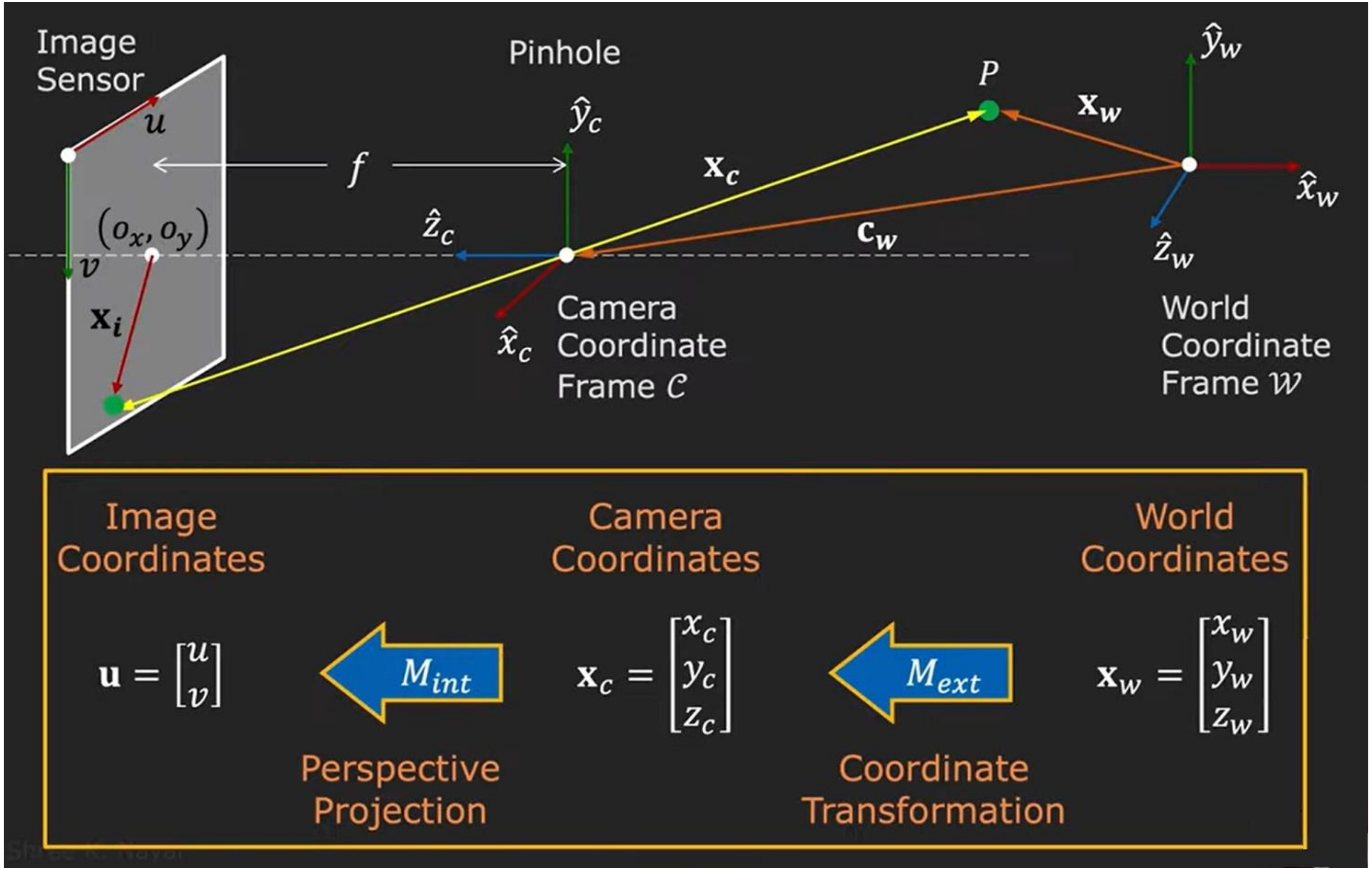
$$\tilde{\mathbf{x}}_c = M_{ext} \tilde{\mathbf{x}}_w$$

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

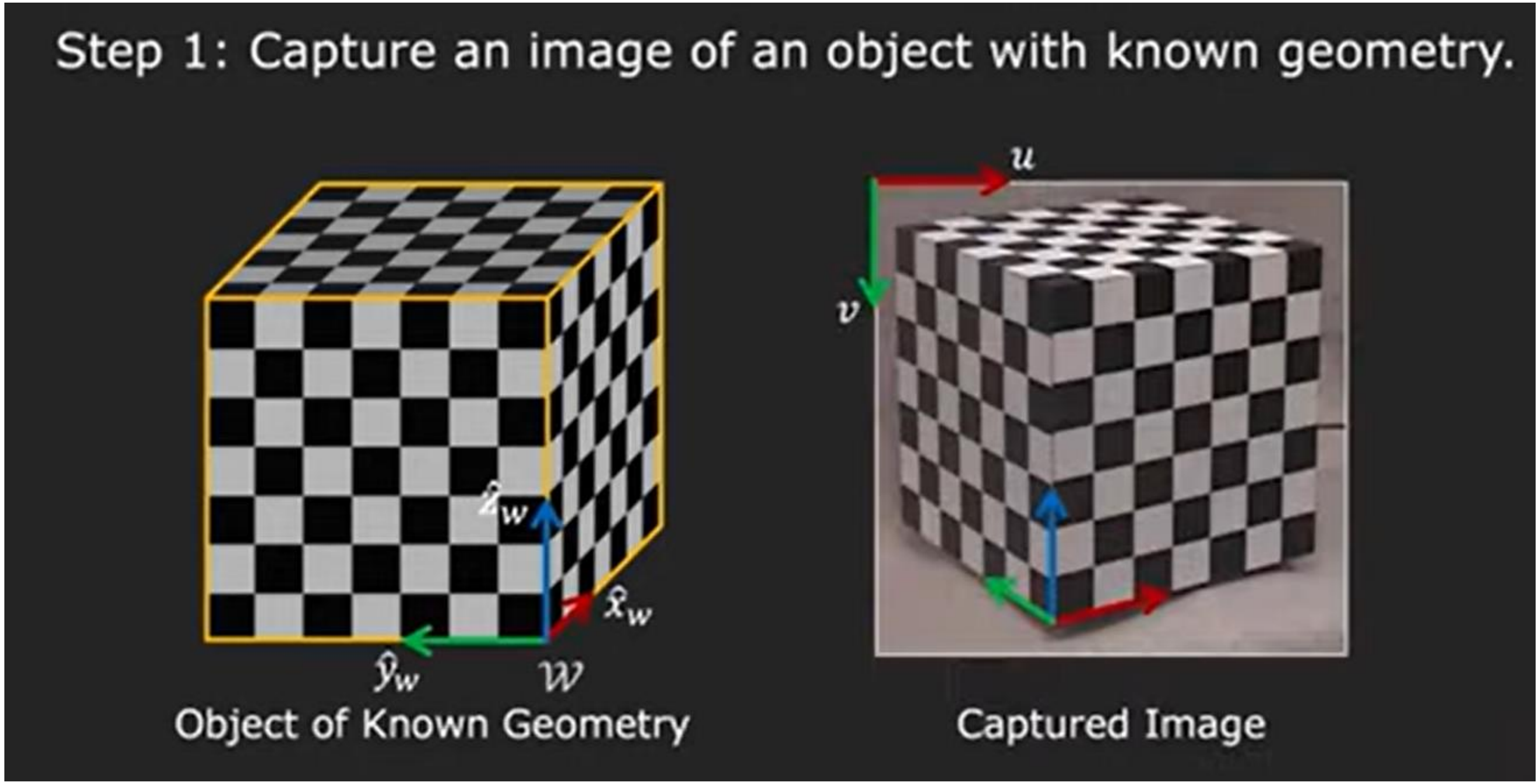
$$\tilde{\mathbf{u}} = M_{int} M_{ext}\, \tilde{\mathbf{x}}_w = P\, \tilde{\mathbf{x}}_w$$

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

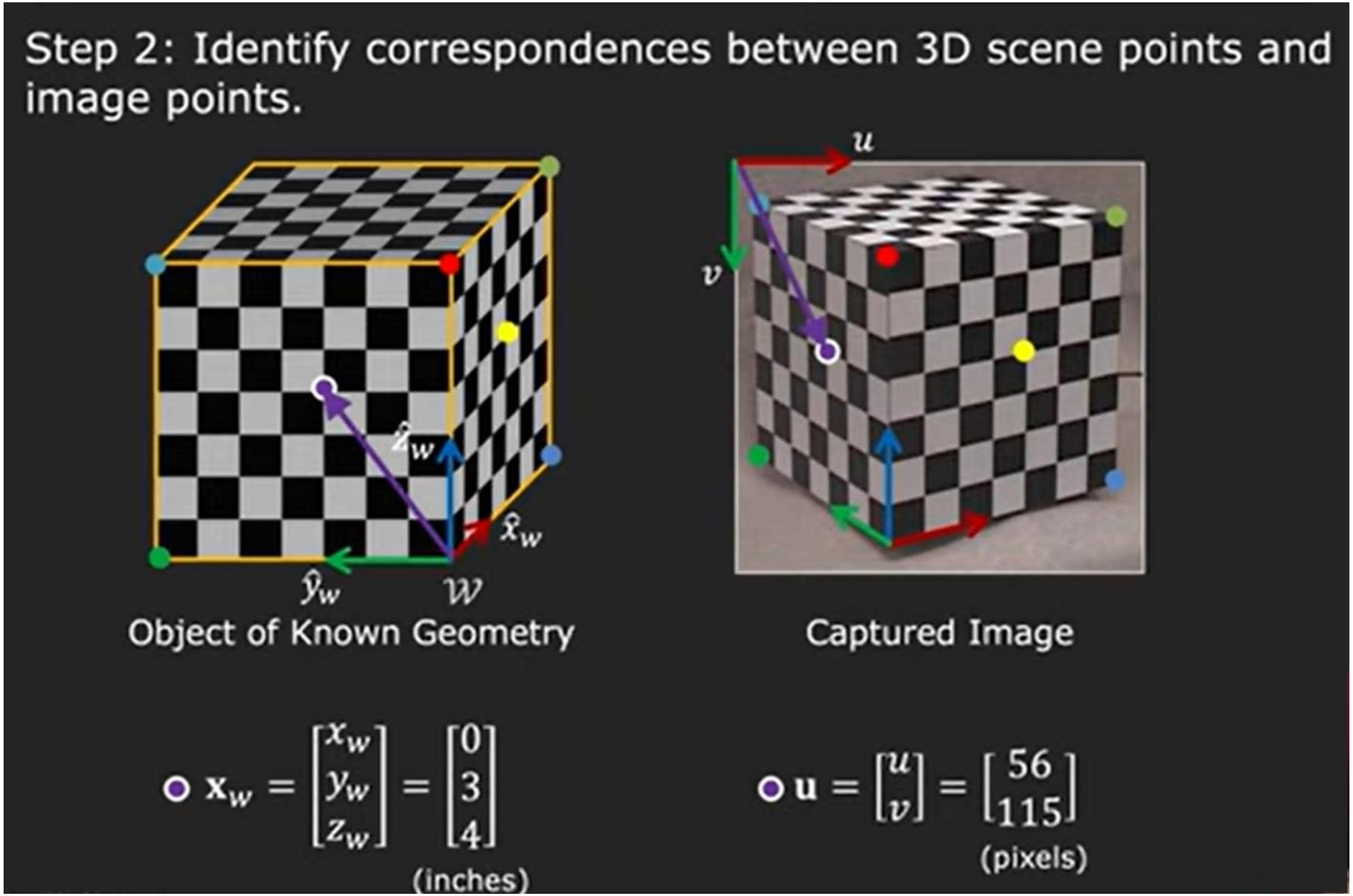
Forward Imaging Model: 3D to 2D



Camera Calibration Procedure (1/7)



Camera Calibration Procedure (2/7)



Camera Calibration Procedure (3/7)

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 (4/7)

Step 4: Rearranging the terms

$$\underbrace{\begin{bmatrix} x_w^{(1)} & y_w^{(1)} & z_w^{(1)} & 1 & 0 & 0 & 0 & 0 & -u_1x_w^{(1)} & -u_1y_w^{(1)} & -u_1z_w^{(1)} & -u_1 \\ 0 & 0 & 0 & 0 & x_w^{(1)} & y_w^{(1)} & z_w^{(1)} & 1 & -v_1x_w^{(1)} & -v_1y_w^{(1)} & -v_1z_w^{(1)} & -v_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_w^{(i)} & y_w^{(i)} & z_w^{(i)} & 1 & 0 & 0 & 0 & 0 & -u_ix_w^{(i)} & -u_iy_w^{(i)} & -u_iz_w^{(i)} & -u_i \\ 0 & 0 & 0 & 0 & x_w^{(i)} & y_w^{(i)} & z_w^{(i)} & 1 & -v_ix_w^{(i)} & -v_iy_w^{(i)} & -v_iz_w^{(i)} & -v_i \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_w^{(n)} & y_w^{(n)} & z_w^{(n)} & 1 & 0 & 0 & 0 & 0 & -u_nx_w^{(n)} & -u_ny_w^{(n)} & -u_nz_w^{(n)} & -u_n \\ 0 & 0 & 0 & 0 & x_w^{(n)} & y_w^{(n)} & z_w^{(n)} & 1 & -v_nx_w^{(n)} & -v_ny_w^{(n)} & -v_nz_w^{(n)} & -v_n \end{bmatrix}}_{\substack{A \\ \text{Known}}} \underbrace{\begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \\ p_{31} \\ p_{32} \\ p_{33} \\ p_{34} \end{bmatrix}}_{\substack{\mathbf{p} \\ \text{Unknown}}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Step 5: Solve for $\mathbf{p}$

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

Camera Calibration Procedure: Scale of Projection Matrix (5/7)

Projection matrix acts on homogenous coordinates.

We know that:

$$\begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} \equiv k \begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} \quad (k \neq 0 \text{ is any constant})$$

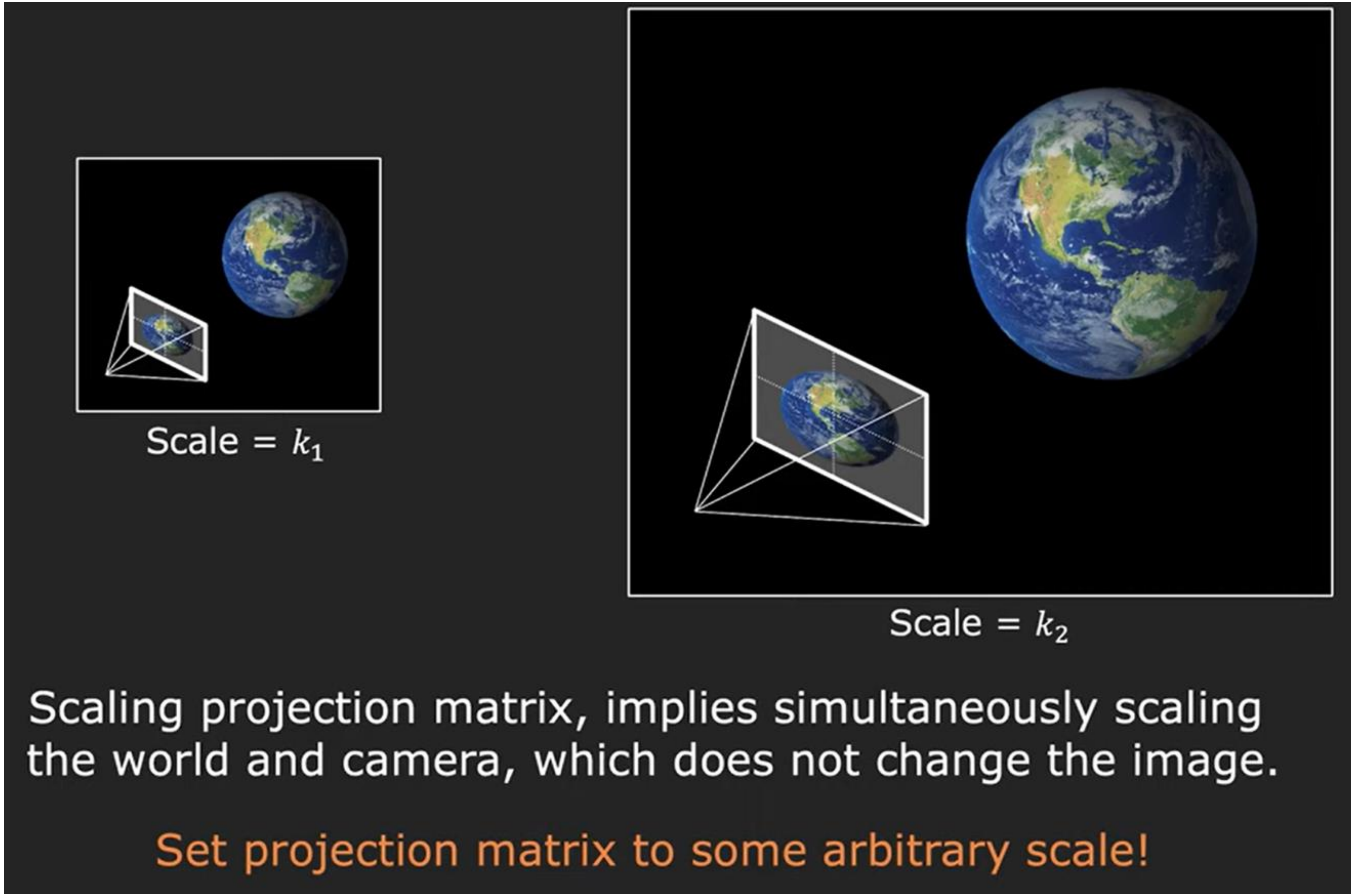
That is:

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

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

Projection Matrix P is defined only up to a scale.

Camera Calibration Procedure: Scale of Projection Matrix (5/7)



Camera Calibration Procedure: Least Squares Solution for *P* (6/7)

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

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

We want $A\mathbf{p}$ as close to 0 as possible and $\|\mathbf{p}\|^2 = 1$:

$$\min_{\mathbf{p}} \|A\mathbf{p}\|^2 \text{ such that } \|\mathbf{p}\|^2 = 1$$

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

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

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

(Similar to Solving Homography in Image Stitching)

Constrained Least Squares Solution

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

$$\boxed{A^T A\mathbf{p} = \lambda\mathbf{p}}$$ Eigenvalue Problem

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

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

Now, We have the Projection Matrix ready.

Extracting Intrinsic / Extrinsic Parameters

We know that:

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

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".

Camera Calibration Procedure: Extracting Intrinsic / Extrinsic Parameters (7/7)

We know that:

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

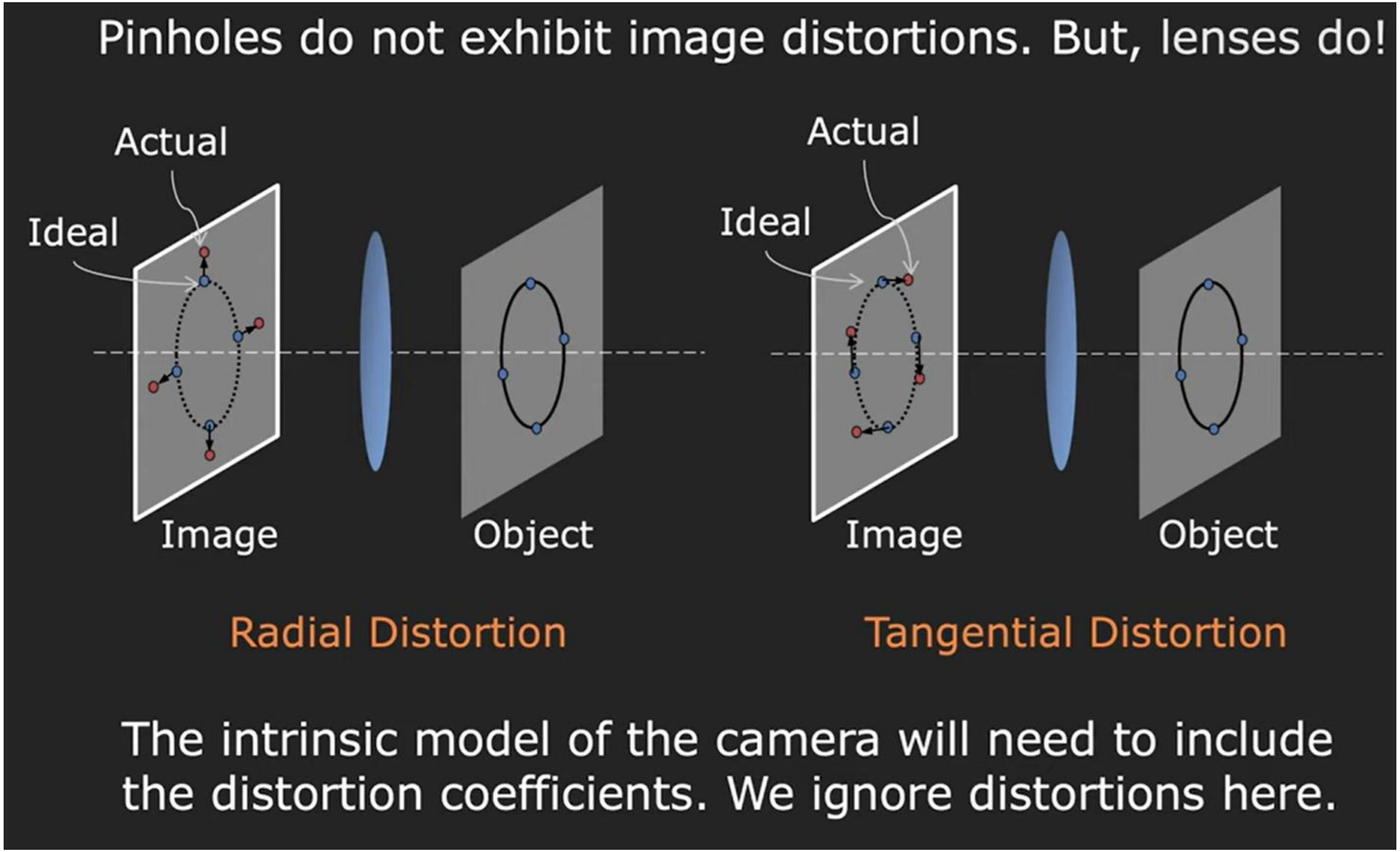
That is:

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

Therefore:

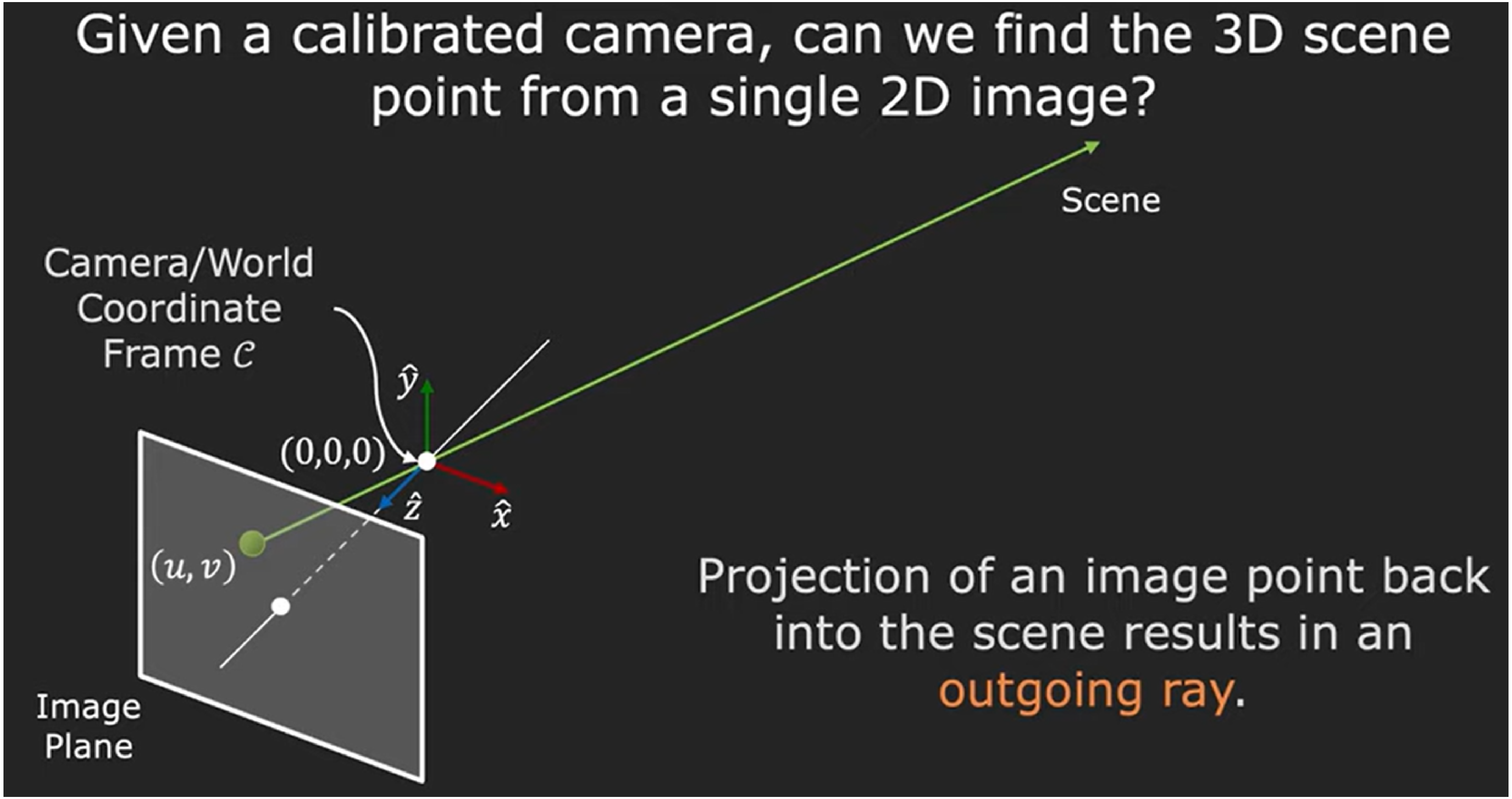
$$\boxed{\mathbf{t} = K^{-1} \begin{bmatrix} p_{14} \\ p_{24} \\ p_{34} \end{bmatrix}}$$

Other intrinsic Parameters

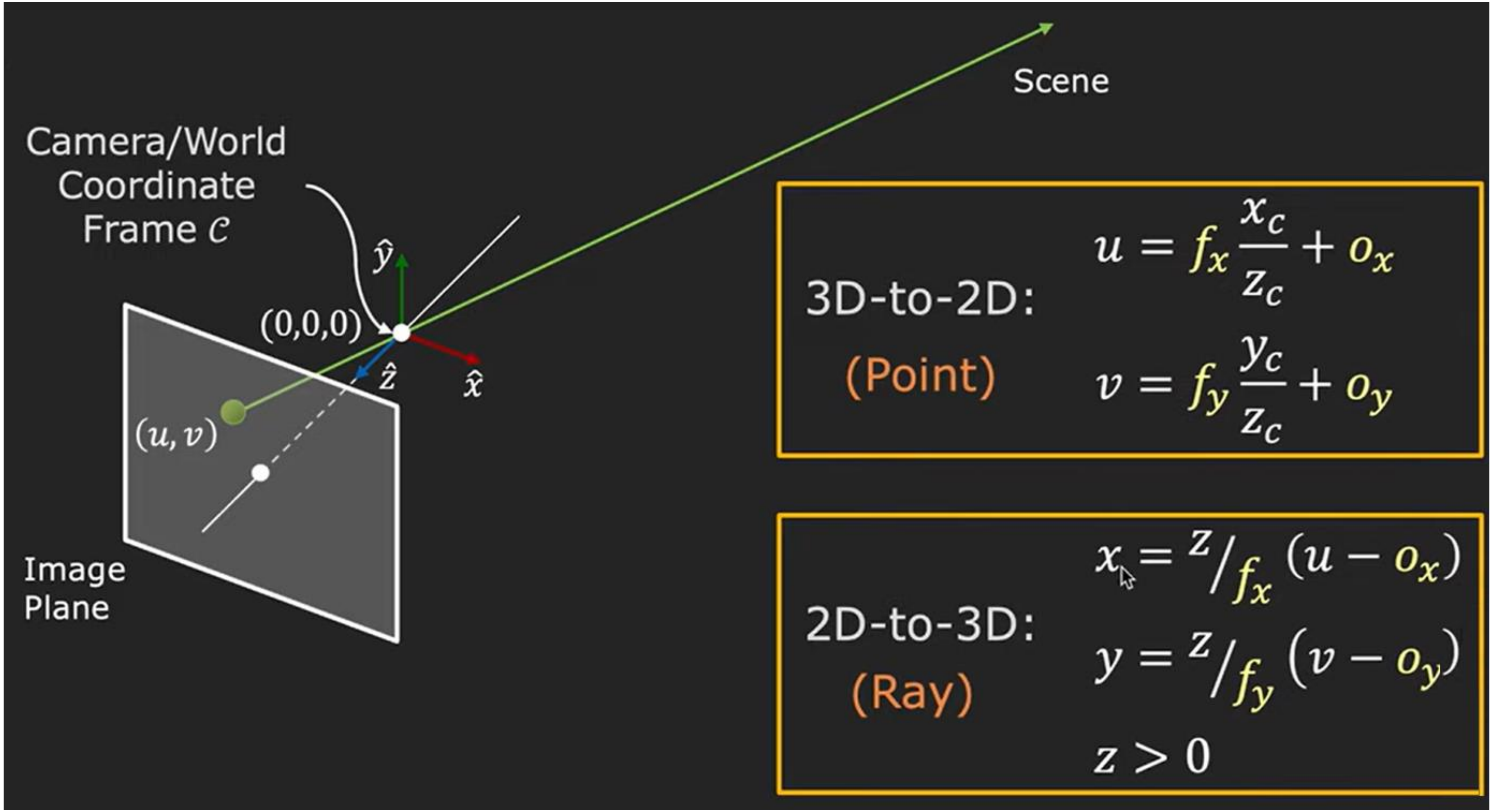


Simple Stereo

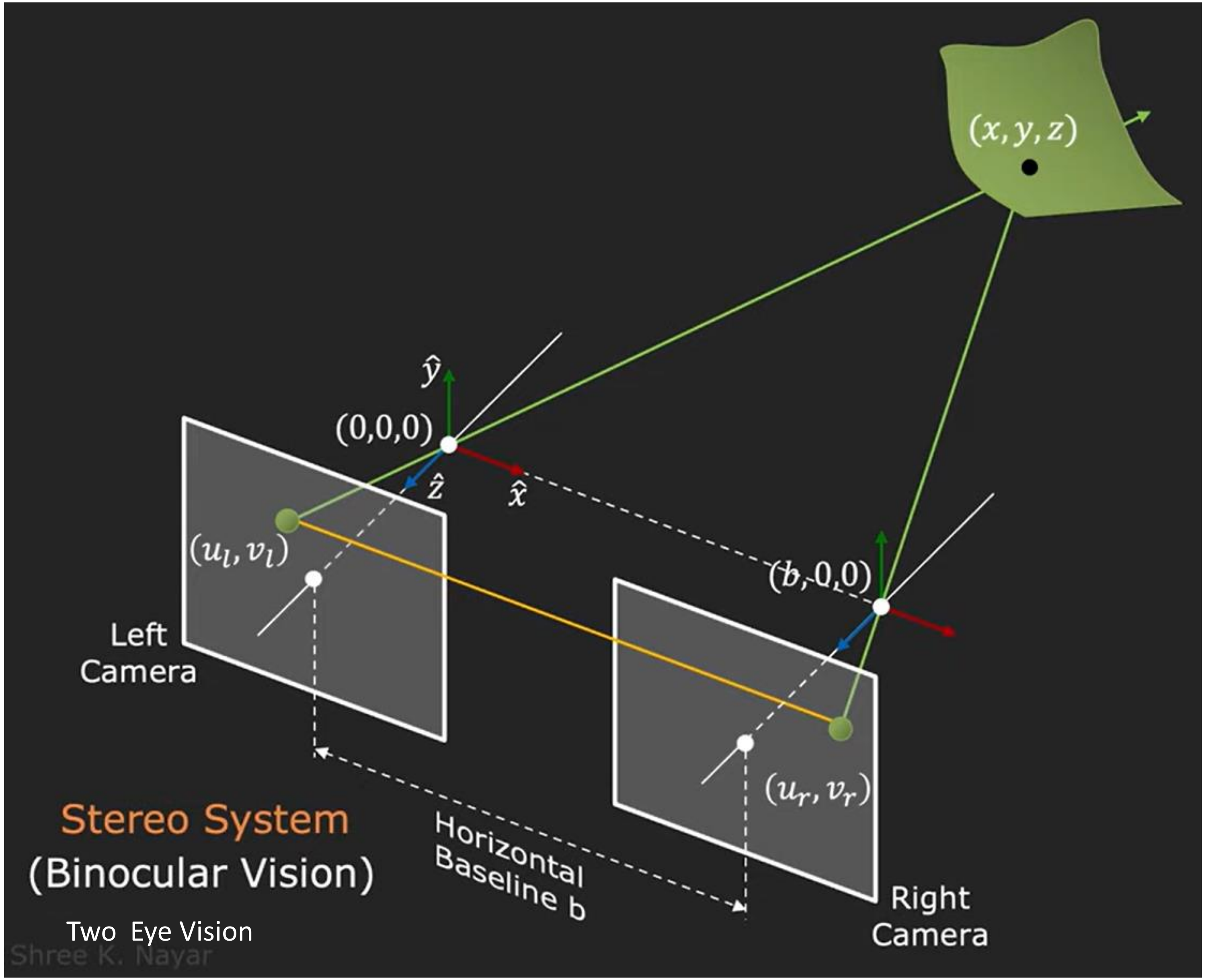
Backward Projection from 2D to 3D



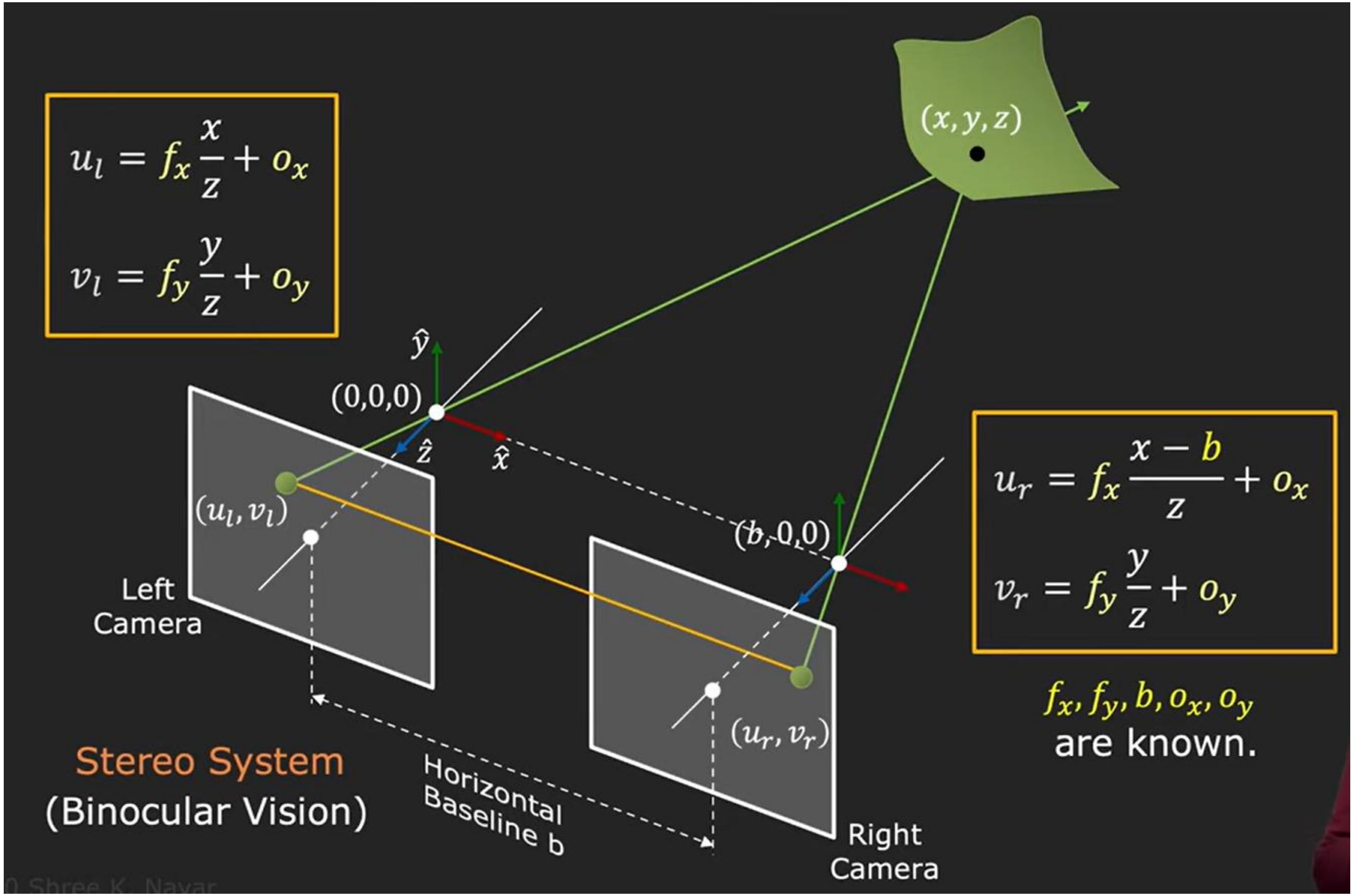
Computing 2D to 3D outgoing ray



Triangulation using two cameras



Triangulation using two cameras



Simple Stereo: Depth and Disparity

From perspective projection:

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

Solving for (x, y, z):

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

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

Depth z is inversely proportional to Disparity.

Disparity is proportional to Baseline.

Stereo Camera Example

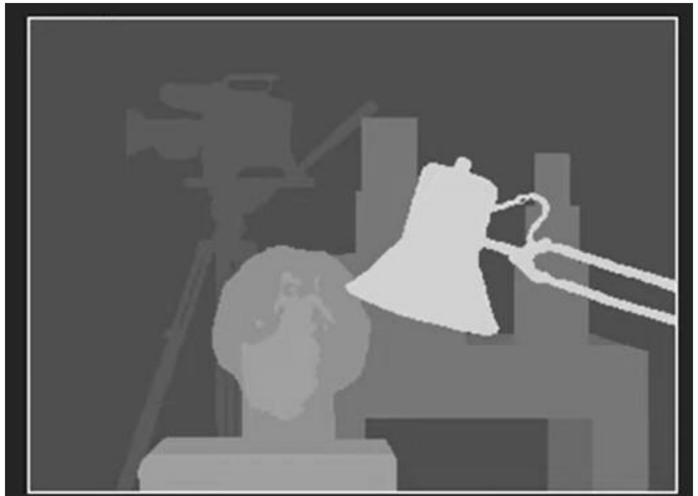
Fujifilm FinePix REAL 3D W3

Stereo Matching: Finding Disparities

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

Left/Right Camera Images

Active Illumination Method



Disparity Map (Ground Truth)

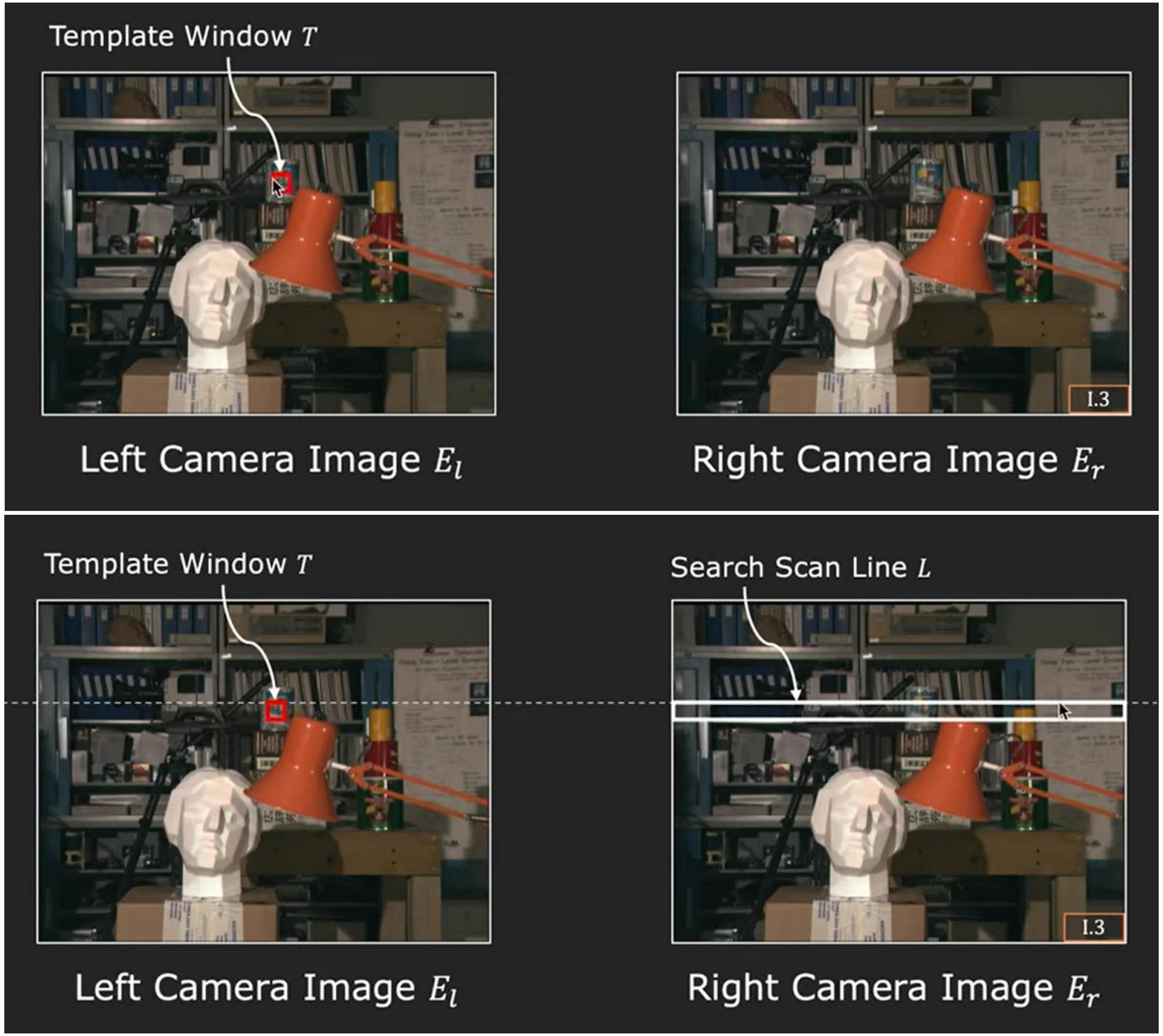
Closer the point, greater the disparity and the brighter it is in Disparity Map.

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

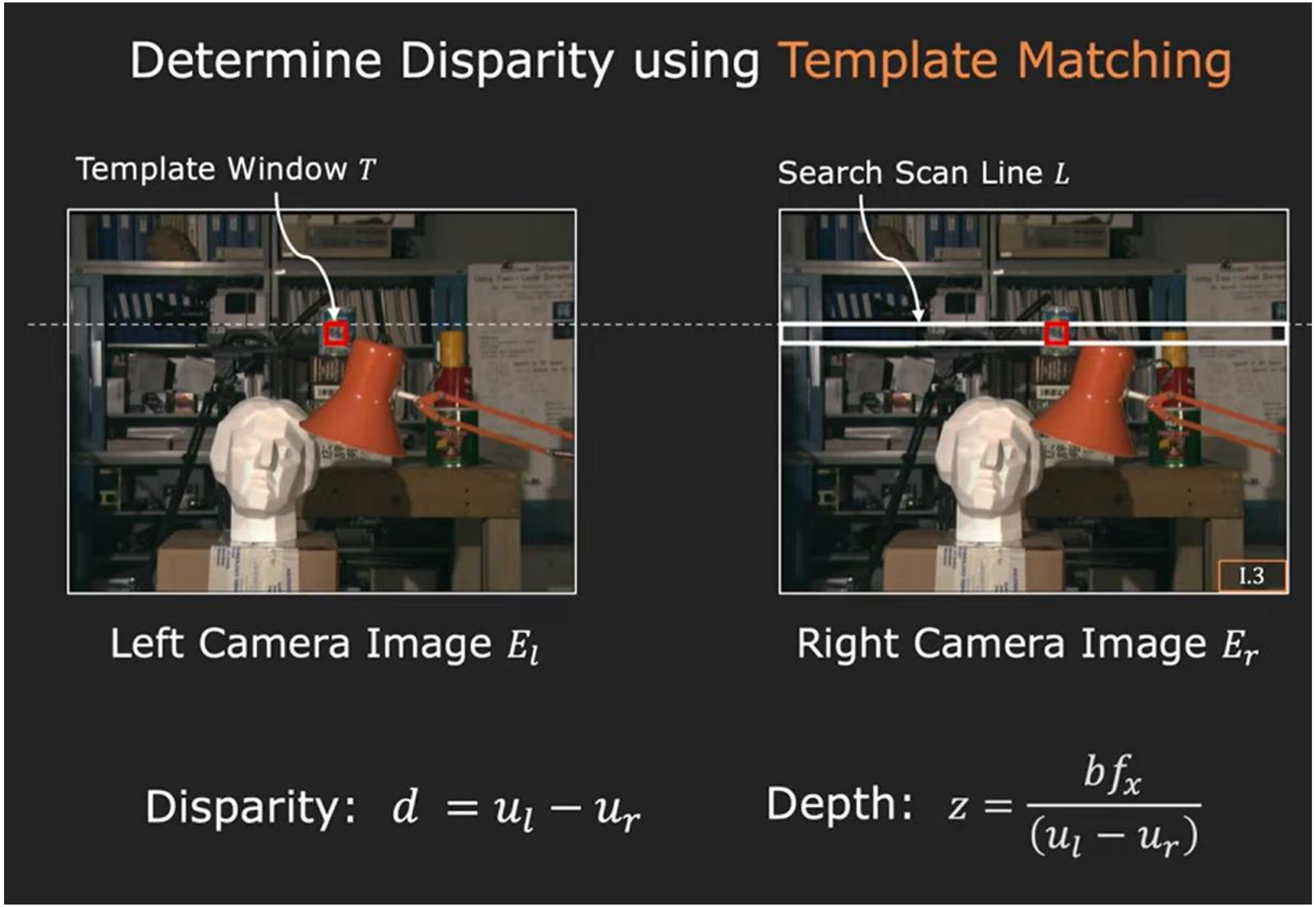
Corresponding scene points lie on the same horizontal scan line.

Window based methods

Determine Disparity using **Template Matching**



Window based methods



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:

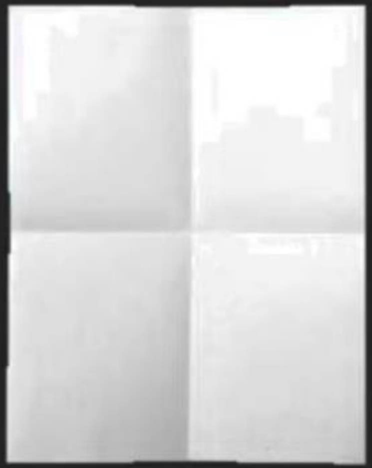
$$SSD(k,l) = \sum_{(i,j)\in T} |E_l(i,j) - E_r(i+k, j+l)|^2$$

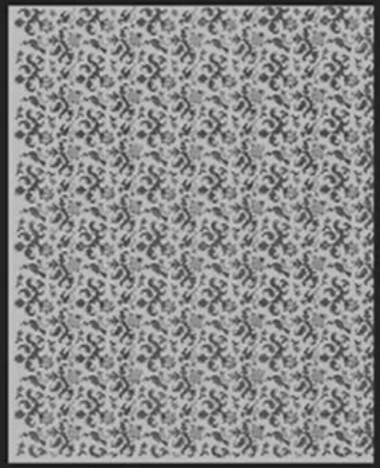
Find pixel $(k,l) \in L$ with Maximum Normalized Cross-Correlation:

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

Issues with Stereo Matching

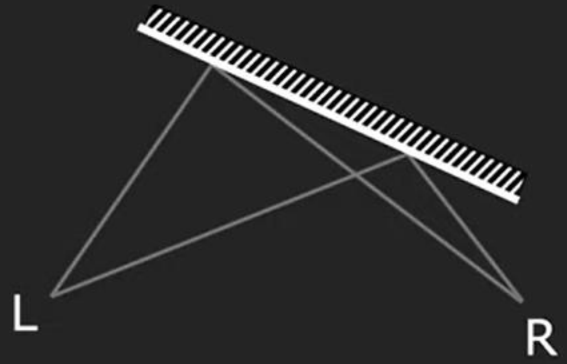
- Surface must have (non-repetitive) texture





- Foreshortening effect makes matching challenging

Projected area is different in the window



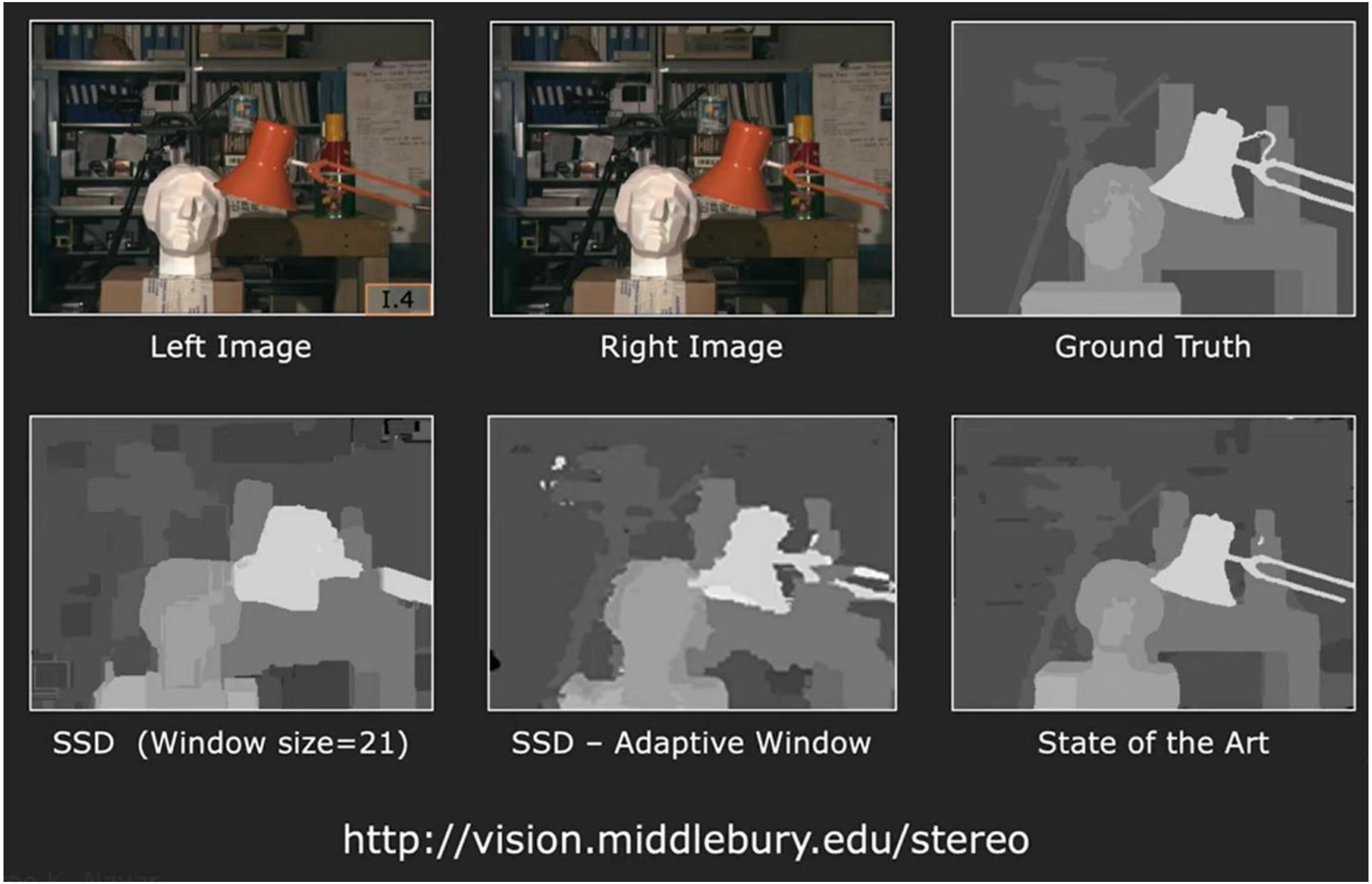
How large the window size be?

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



Uncalibrated Stereo

Uncalibrated Stereo

Method to estimate 3D structure of a static scene from two arbitrary views.

Topics:

(1) Problem of Uncalibrated Stereo

(2) Epipolar Geometry

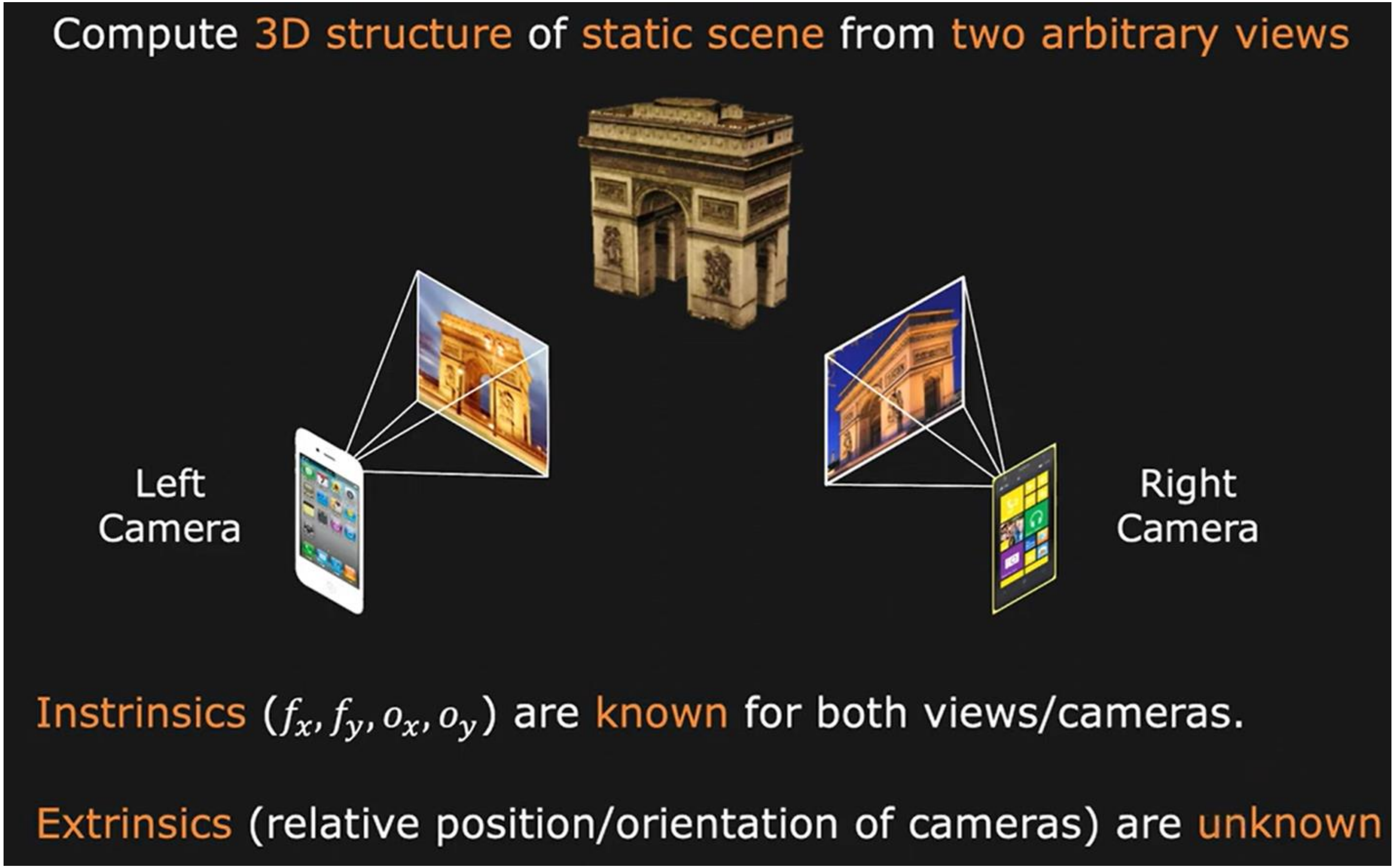
(3) Estimating Fundamental Matrix

(4) Finding Dense Correspondences

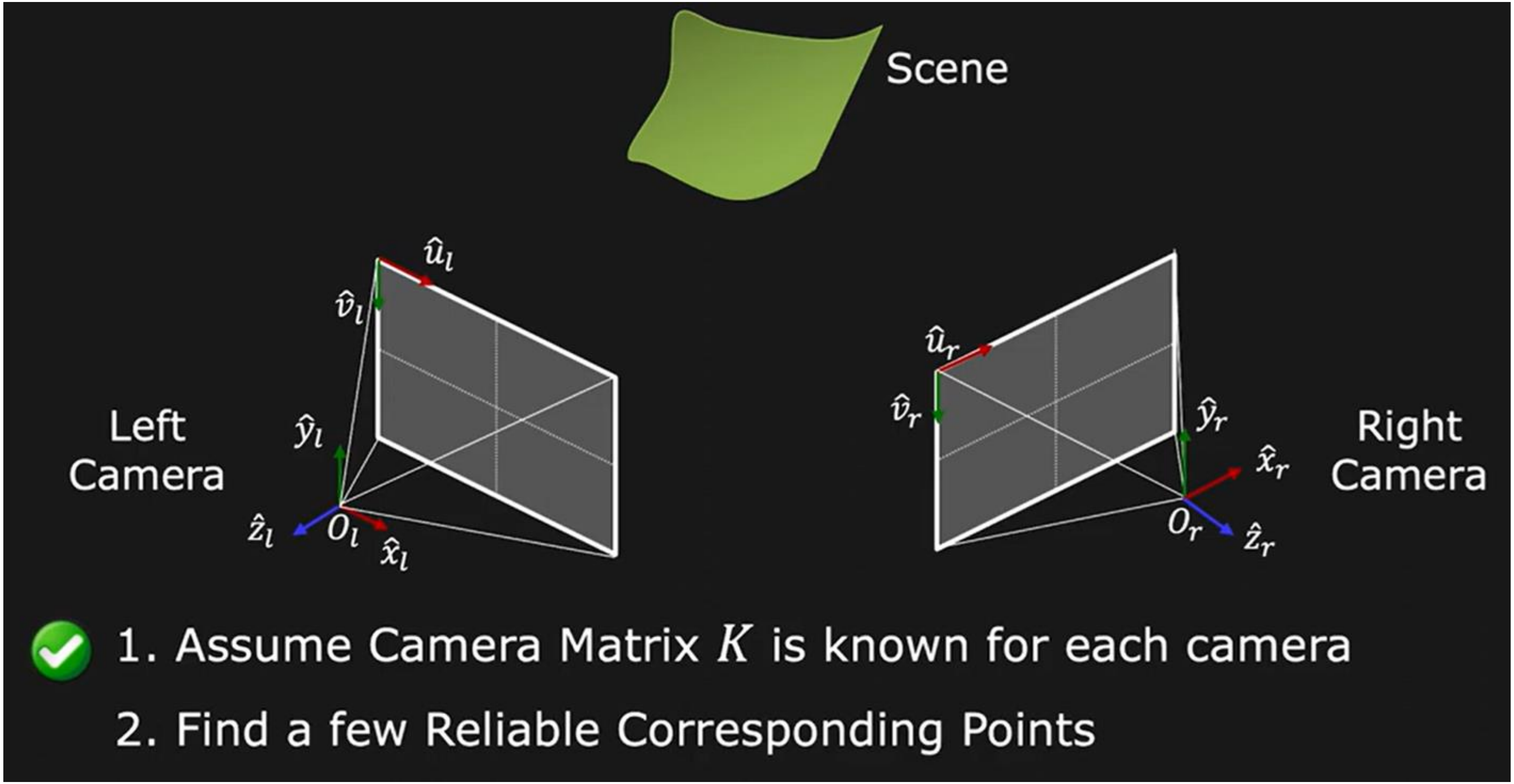
(5) Computing Depth

(6) Stereopsis: Stereo in Nature

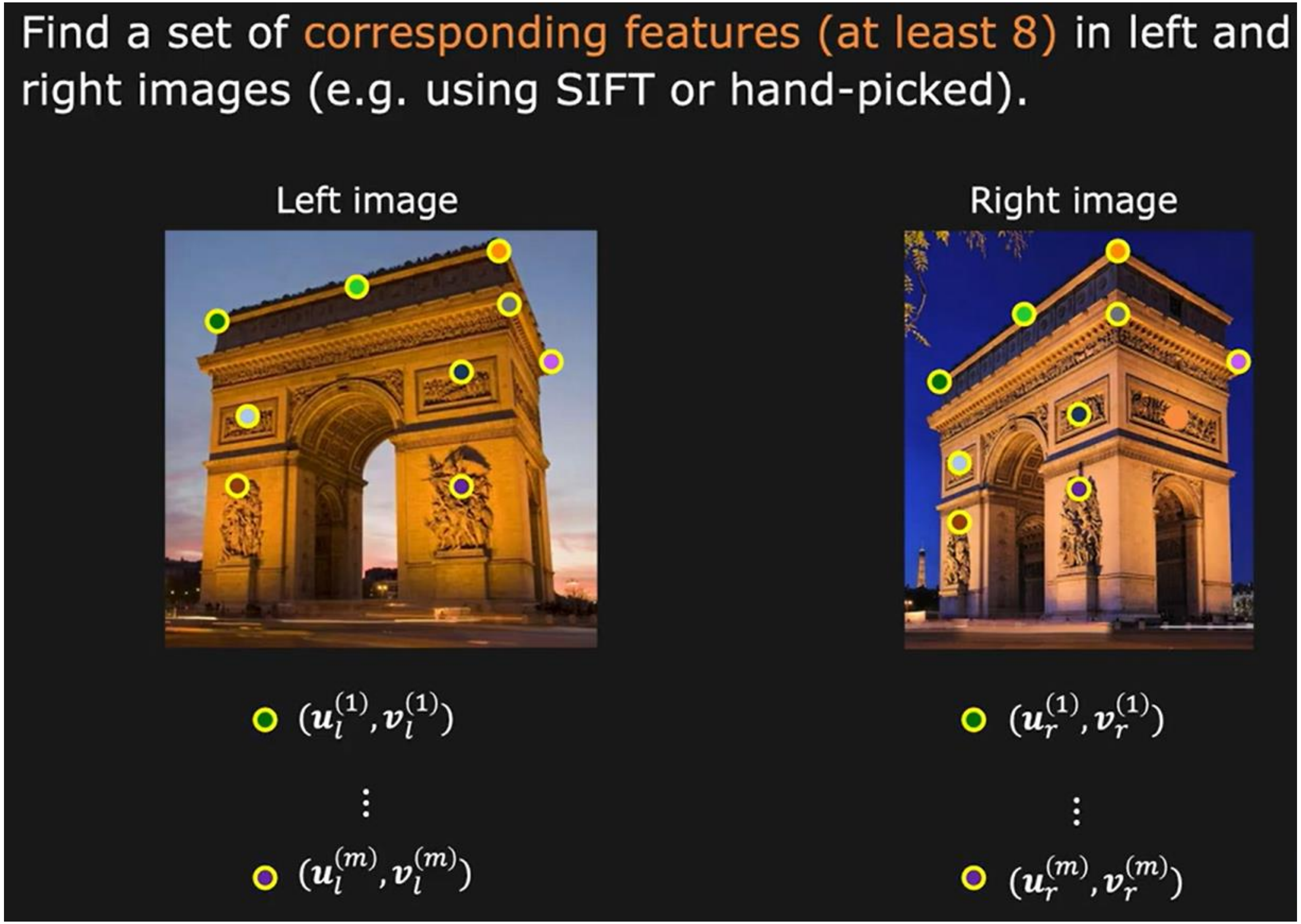
Uncalibrated Stereo



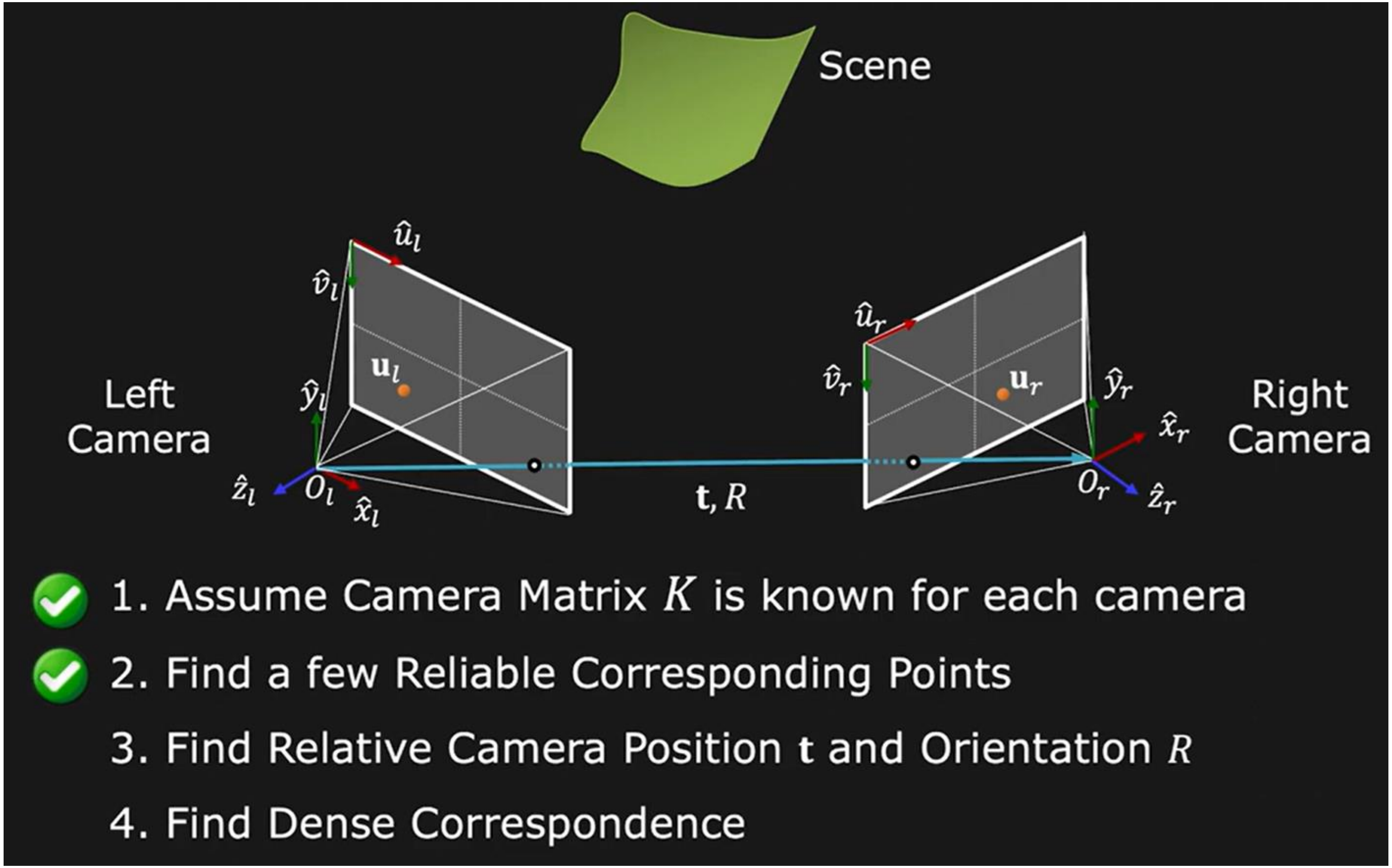
Uncalibrated Stereo



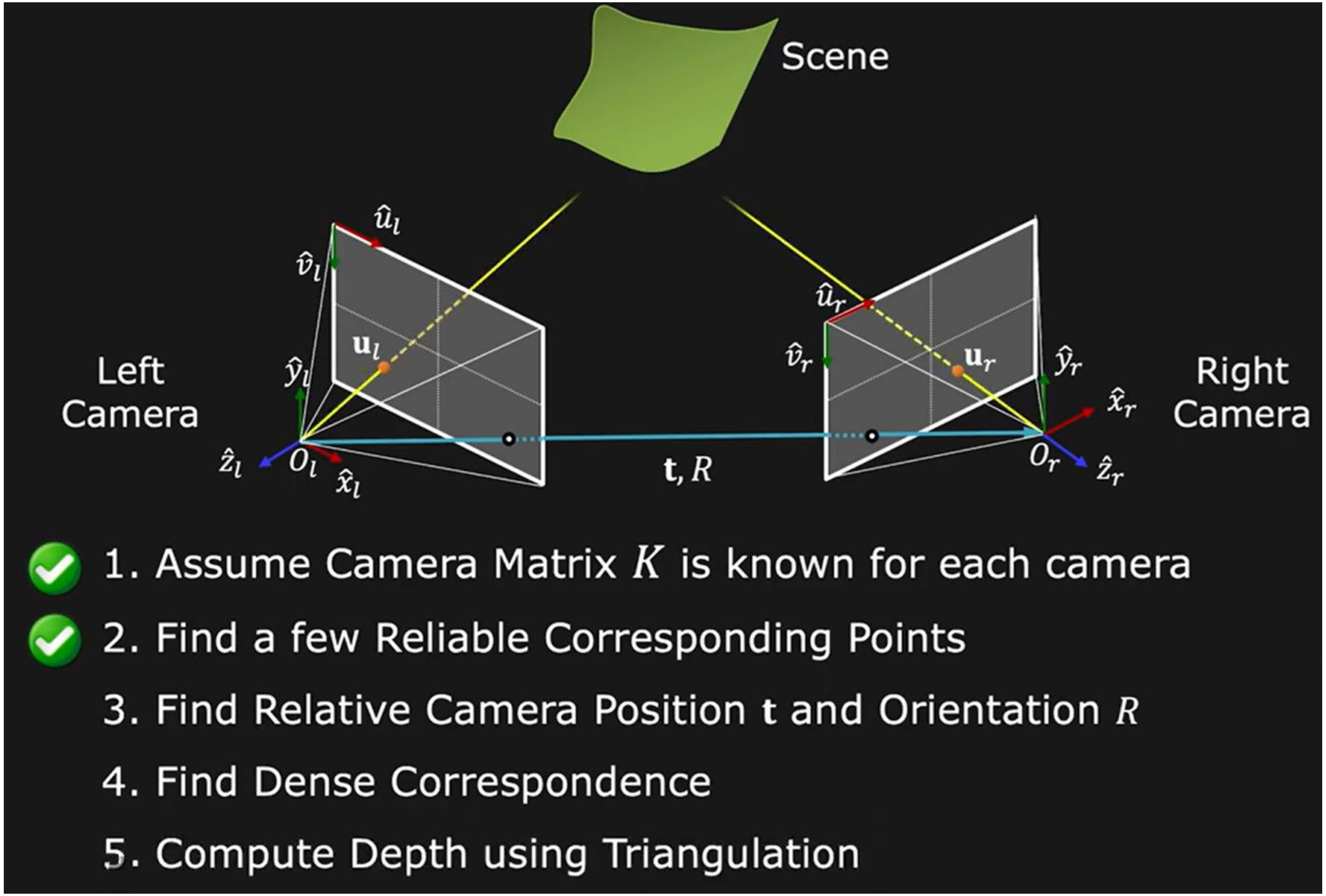
Initial Correspondence points



Uncalibrated Stereo



Uncalibrated Stereo



Epipolar Geometry

Goal: Our goal is to find the relative position and orientation of one camera with respect to other camera. This is the process of calibrating an uncalibrated stereo system

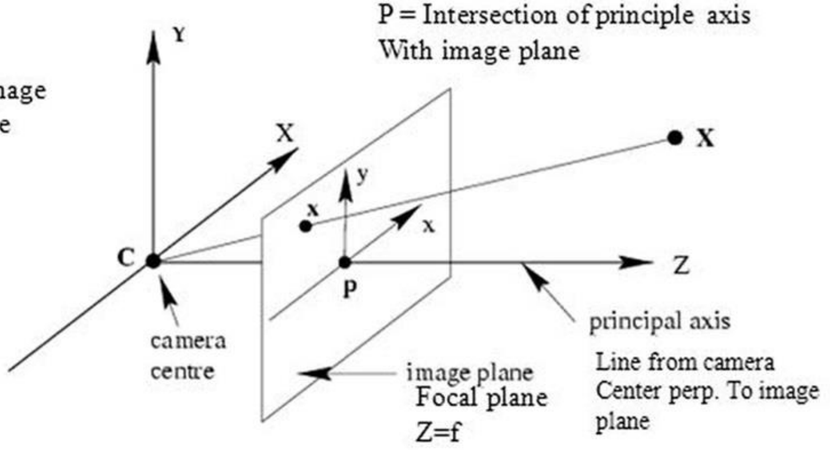
The relative position and orientation between two cameras is completely described by the Epipolar geometry of the stereo system.

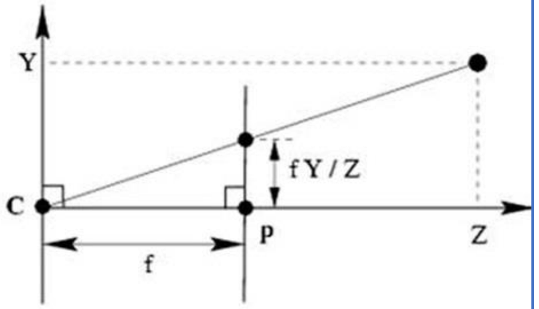
Pin hole camera model

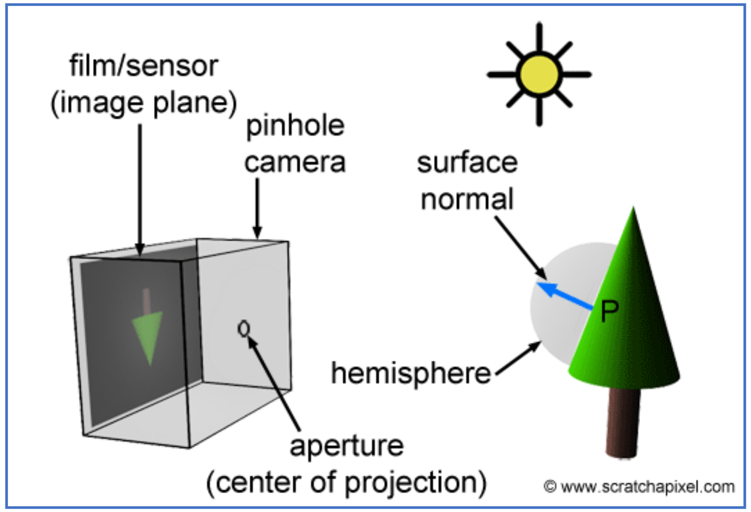
Camera is a mapping between 3D world (object space) and 2D image;

Camera model: matrix representing camera mapping; interested in central projection

Plane through camera Center C, parallel to image Plane is called principle Plane of the camera





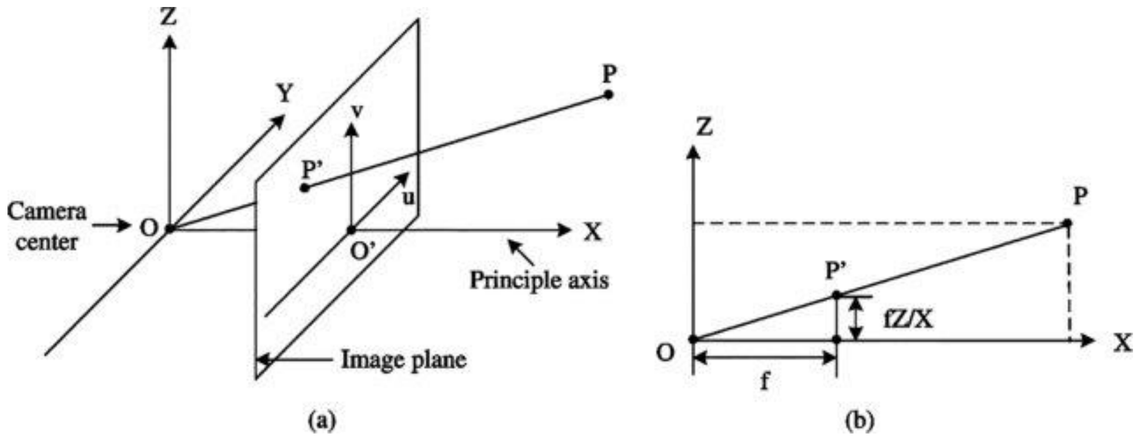


$$(X, Y, Z)^T \mapsto (fX/Z, fY/Z)^T$$

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

linear projection in homogeneous coordinates!

Camera Projection Model



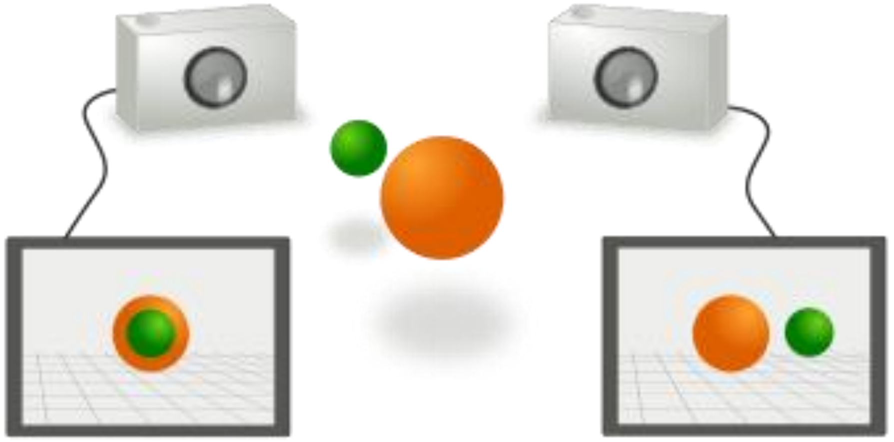
❖ Each camera captures a 2D image of the 3D world. This conversion from 3D to 2D is referred to as a perspective projection and is described by the pinhole camera model.

❖ It is common to model this projection operation by rays that emanate from the camera, passing through its focal center. Each emanating ray corresponds to a single point in the image.

❖ In real cameras, the image plane is actually behind the focal center, and produces an image that is symmetric about the focal center of the lens. Here, *virtual image plane is placed* in front of the focal center i.e. optical center of each camera lens to produce an image.

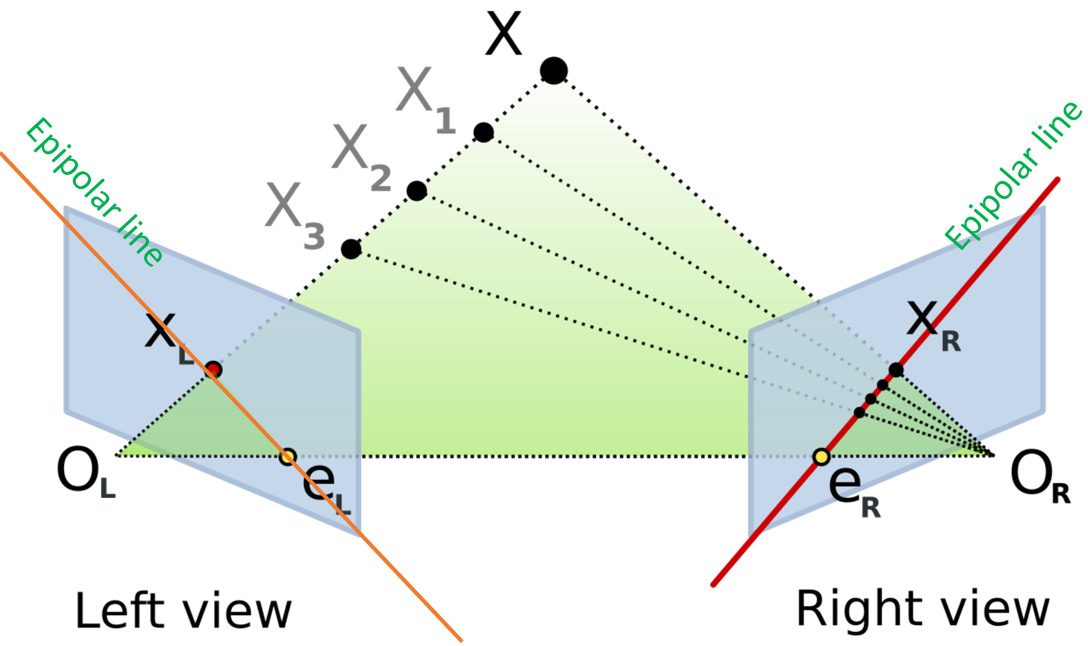
What is Epipolar Geometry?

- Epipolar geometry is the geometry of stereo vision.
- The relative position and orientation between two cameras is completely described by the Epipolar geometry of the stereo system.
- Epipolar geometry is defined as **the geometry that describes the relationship between a pair of images taken by two cameras or different locations of a mobile camera**.
- Epipolar Geometry involves the concept of Epipolar lines and the Fundamental Matrix to represent the projective motion between uncalibrated perspective cameras.
- Epipolar geometry simplifies the search for correspondence points between the image peers.



Typical Usecase:
Two cameras take a picture of the same scene from different points of view. The Epipolar geometry then describes the relation between the two resulting views.

What is Epipolar Geometry?

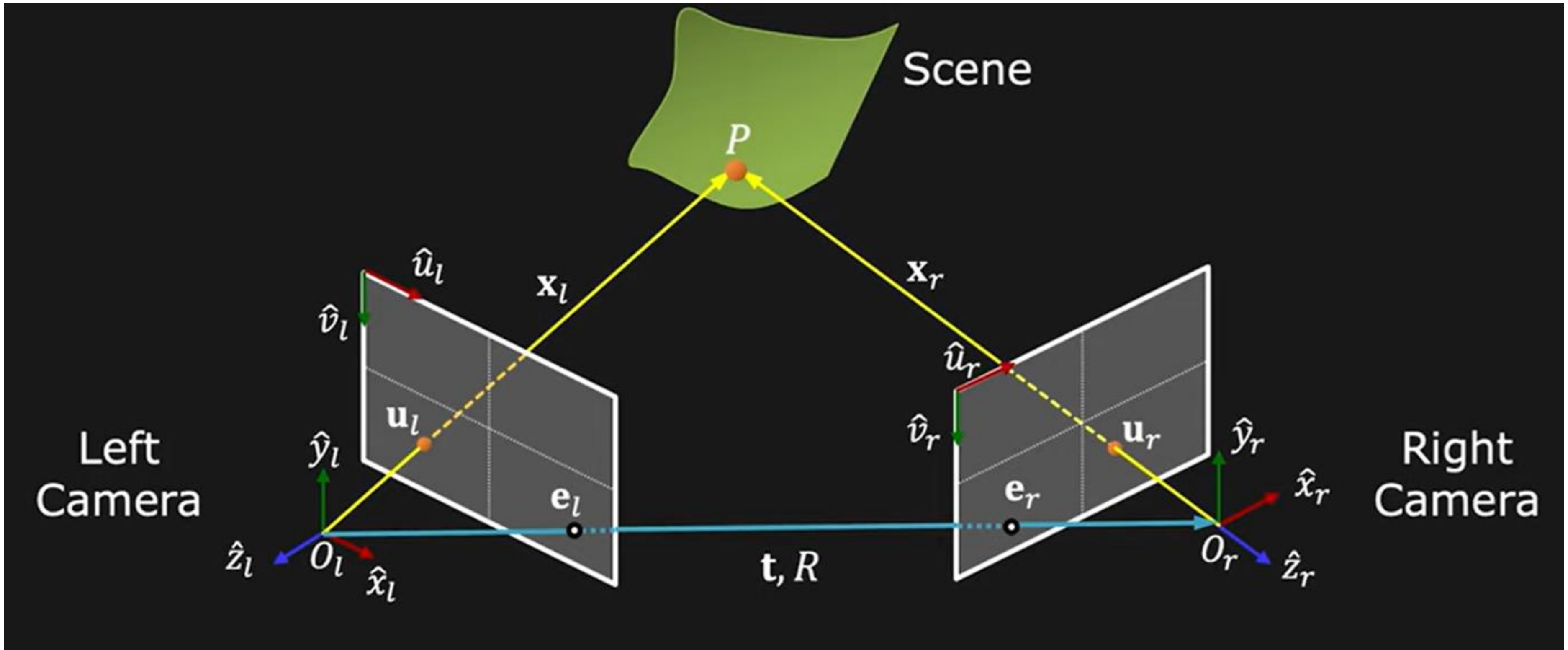


Two pinhole cameras looking at point **X**

- **O_L** and **O_R** represent the centers of symmetry of the two cameras lenses.
- **X** represents the point of interest in both cameras.
- Points **x_L** and **x_R** are the projections of point **X** onto the image plane.
- As the optical centers of the cameras lenses are distinct, each center projects onto a distinct point into the other camera's image plane. These two image points, denoted by **e_L** and **e_R**, are called *epipoles* or *epipolar points*. Both epipoles **e_L** and **e_R** in their respective image planes and both optical centers **O_L** and **O_R** lie on a single 3D line.
- The line **O_L–X** is seen by the left camera as a point because it is directly in line with that camera's lens optical center. However, the right camera sees this line as a line in its image plane. That line (**e_R–x_R**) in the right camera is called an *epipolar line*.
- Symmetrically, the line **O_R–X** is seen by the right camera as a point and is seen as epipolar line **e_L–x_L** by the left camera.

- An epipolar line is a function of the position of point **X** in the 3D space, i.e. as **X** varies, a set of epipolar lines is generated in both images.
- Since the 3D line **O_L–X** passes through the optical center of the lens **O_L**, the corresponding epipolar line in the right image must pass through the epipole **e_R** (and correspondingly for epipolar lines in the left image).
- All epipolar lines in one image contain the epipolar point of that image.
- In fact, any line which contains the Epipolar point is an Epipolar line since it can be derived from some 3D point **X**.

Epipolar Geometry: Epipoles

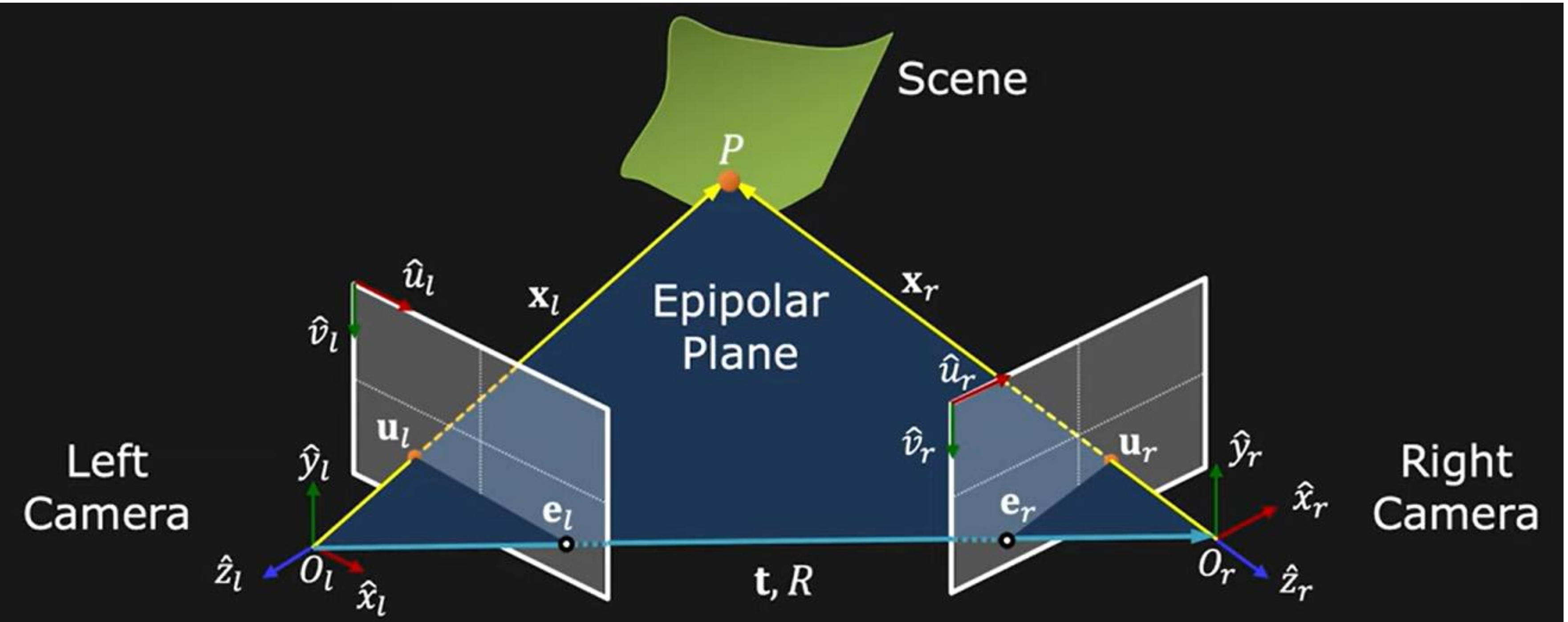


Epipole: Image point of origin/pinhole of one camera as viewed by the other camera.

$\mathbf{e}_l$ and $\mathbf{e}_r$ are the epipoles.

$\mathbf{e}_l$ and $\mathbf{e}_r$ are unique for a given stereo pair.

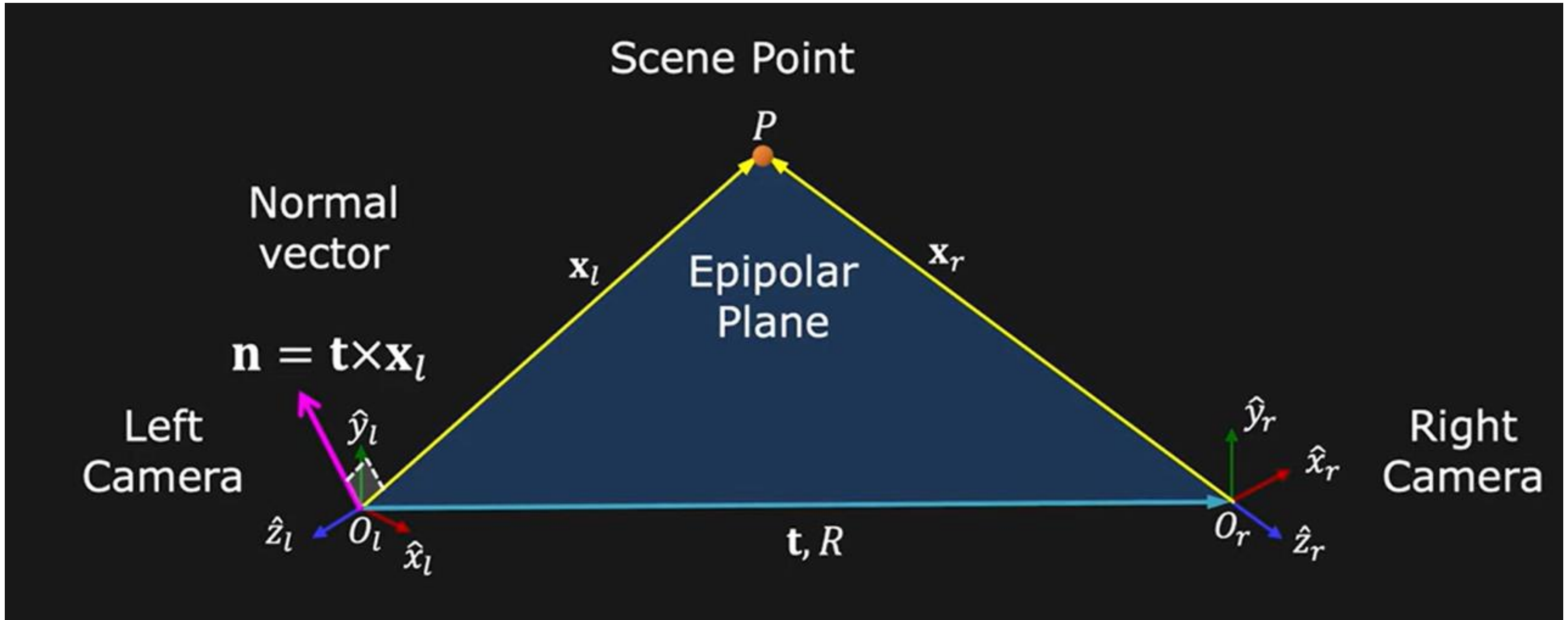
Epipolar Geometry: Epipolar Plane



Epipolar Plane of Scene Point P: The plane formed by camera origins (O_l and O_r), epipoles ($\mathbf{e}_l$ and $\mathbf{e}_r$) and scene point P.

Every scene point lies on a unique epipolar plane.

Epipolar Constraint



Vector normal to the epipolar plane: $\mathbf{n} = \mathbf{t} \times \mathbf{x}_l$

Dot product of $\mathbf{n}$ and $\mathbf{x}_l$ (perpendicular vectors) is zero:

$$\boxed{\mathbf{x}_l \cdot (\mathbf{t} \times \mathbf{x}_l) = 0}$$

$\mathbf{t}_{3\times1}$: Position of Right Camera in Left Camera's Frame

$R_{3\times3}$: Orientation of Left Camera in Right Camera's Frame

Vector Cross product

$$\mathbf{a} = (a_x, a_y, a_z) = a_x\mathbf{i} + a_y\mathbf{j} + a_z\mathbf{k}$$
$$\mathbf{b} = (b_x, b_y, b_z) = b_x\mathbf{i} + b_y\mathbf{j} + b_z\mathbf{k}$$

Multiplying the two vectors, we get:

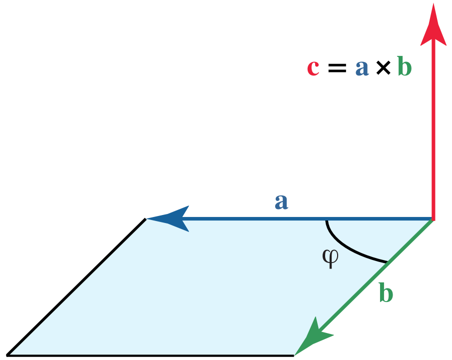
$$\begin{aligned}\mathbf{a} \times \mathbf{b} &= (a_x\mathbf{i} + a_y\mathbf{j} + a_z\mathbf{k}) \times (b_x\mathbf{i} + b_y\mathbf{j} + b_z\mathbf{k}) \\ &= a_xb_x\mathbf{i} \times \mathbf{i} + a_xb_y\mathbf{i} \times \mathbf{j} + a_xb_z\mathbf{i} \times \mathbf{k} \\ &\quad + a_yb_x\mathbf{j} \times \mathbf{i} + a_yb_y\mathbf{j} \times \mathbf{j} + a_yb_z\mathbf{j} \times \mathbf{k} \\ &\quad + a_zb_x\mathbf{k} \times \mathbf{i} + a_zb_y\mathbf{k} \times \mathbf{j} + a_zb_z\mathbf{k} \times \mathbf{k}\end{aligned}$$

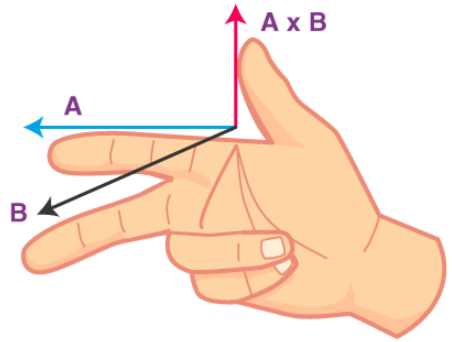
Then, using the unit vector relationships above, this simplifies to:

$$\begin{aligned}\mathbf{a} \times \mathbf{b} &= a_xb_y\mathbf{i} \times \mathbf{j} - a_xb_z\mathbf{k} \times \mathbf{i} - a_yb_x\mathbf{i} \times \mathbf{j} + a_yb_z\mathbf{j} \times \mathbf{k} + a_zb_x\mathbf{k} \times \mathbf{i} - a_zb_y\mathbf{j} \times \mathbf{k} \\ &= (a_xb_y - a_yb_x)\mathbf{i} \times \mathbf{j} + (a_zb_x - a_xb_z)\mathbf{k} \times \mathbf{i} + (a_yb_z - a_zb_y)\mathbf{j} \times \mathbf{k} \\ &= (a_yb_z - a_zb_y)\mathbf{i} + (a_zb_x - a_xb_z)\mathbf{j} + (a_xb_y - a_yb_x)\mathbf{k}\end{aligned}$$

$$\mathbf{a} \times \mathbf{b} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_x & a_y & a_z \\ b_x & b_y & b_z \end{vmatrix}$$
$$= \begin{vmatrix} a_y & a_z \\ b_y & b_z \end{vmatrix}\mathbf{i} - \begin{vmatrix} a_x & a_z \\ b_x & b_z \end{vmatrix}\mathbf{j} + \begin{vmatrix} a_x & a_y \\ b_x & b_y \end{vmatrix}\mathbf{k}$$

Where the determinant $\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$





$$\mathbf{i} \times \mathbf{j} = \mathbf{k}$$
$$\mathbf{j} \times \mathbf{k} = \mathbf{i}$$
$$\mathbf{k} \times \mathbf{i} = \mathbf{j}$$
$$\mathbf{i} \times \mathbf{i} = \mathbf{j} \times \mathbf{j} = \mathbf{k} \times \mathbf{k} = 0$$

$$\mathbf{j} \times \mathbf{i} = -\mathbf{k}$$
$$\mathbf{k} \times \mathbf{j} = -\mathbf{i}$$
$$\mathbf{i} \times \mathbf{k} = -\mathbf{j}$$

The following are several properties of the vector cross product:

#1. If $\mathbf{a}$ and $\mathbf{b}$ are parallel, then $\mathbf{a} \times \mathbf{b} = 0$

#2. $\mathbf{a} \times \mathbf{b} = -\mathbf{b} \times \mathbf{a}$

#3. $\mathbf{a} \times (\mathbf{b} + \mathbf{c}) = \mathbf{a} \times \mathbf{b} + \mathbf{a} \times \mathbf{c}$

#4. $(c\mathbf{a}) \times \mathbf{b} = c(\mathbf{a} \times \mathbf{b})$

#5. $\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c}) = (\mathbf{a} \times \mathbf{b}) \cdot \mathbf{c}$

Where $\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c}) = \begin{vmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \end{vmatrix}$

Epipolar Constraint

Writing the epipolar constraint in matrix form:

$$\mathbf{x}_l \cdot (\mathbf{t} \times \mathbf{x}_l) = 0$$

$$\begin{bmatrix} x_l & y_l & z_l \end{bmatrix} \begin{bmatrix} t_y z_l - t_z y_l \\ t_z x_l - t_x z_l \\ t_x y_l - t_y x_l \end{bmatrix} = 0$$

Cross-product definition

$$\boxed{\begin{bmatrix} x_l & y_l & z_l \end{bmatrix} \underbrace{\begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}}_{T_\times} \begin{bmatrix} x_l \\ y_l \\ z_l \end{bmatrix} = 0}$$

Matrix-vector form

$\mathbf{t}_{3\times1}$: Position of Right Camera in Left Camera's Frame

$R_{3\times3}$: Orientation of Left Camera in Right Camera's Frame

$$\boxed{\mathbf{x}_l = R\mathbf{x}_r + \mathbf{t}}$$

$$\boxed{\begin{bmatrix} x_l \\ y_l \\ z_l \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}}$$

Epipolar Constraint

Substituting into the epipolar constraint gives:

$$\begin{bmatrix} x_l & y_l & z_l \end{bmatrix} \left(\boxed{\begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}} \begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} + \underbrace{\boxed{\begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}}}_{\boldsymbol{t} \times \boldsymbol{t} = \boldsymbol{0}} \right) = 0$$

$$\begin{bmatrix} x_l & y_l & z_l \end{bmatrix} \boxed{\begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix}} \begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} = 0$$

Essential Matrix E

$$\boxed{E = T_{\times} R}$$

[Longuet-Higgins 1981]

Essential Matrix E: Decomposition

$$\boxed{E = T_\times R}$$

$$\begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

Given that $T_\times$ is a Skew-Symmetric matrix (a_{ij} = - a_{ji}) and R is an Orthonormal matrix, it is possible to "decouple" $T_\times$ and R from their product using "Singular Value Decomposition".

Take Away: If E is known, we can calculate $\boldsymbol{t}$ and R.

How do we find the Essential Matrix E?

Relates 3D position (x_l, y_l, z_l) of scene point w.r.t left camera to its 3D position (x_r, y_r, z_r) w.r.t. right camera

$$\boxed{\mathbf{x}_l^T E \mathbf{x}_r \;= 0}$$

$$\begin{bmatrix} x_l & y_l & z_l \end{bmatrix} \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} = 0$$

3D position in left camera coordinates

3x3 Essential Matrix

3D position in right camera coordinates

How do we find the Essential Matrix E?

Relates 3D position (x_l, y_l, z_l) of scene point w.r.t left camera to its 3D position (x_r, y_r, z_r) w.r.t. right camera

$$\boxed{\mathbf{x}_l^T E \mathbf{x}_r = 0}$$

$$\begin{bmatrix} x_l & y_l & z_l \end{bmatrix} \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} = 0$$

3D position in left camera coordinates

3x3 Essential Matrix

3D position in right camera coordinates

Unfortunately, we don't have $\mathbf{x}_l$ and $\mathbf{x}_r$.

But we do know corresponding points in image coordinates.

Incorporating the Image Coordinates

Perspective projection equations for left camera:

$$u_l = f_x^{(l)} \frac{x_l}{z_l} + o_x^{(l)} \qquad v_l = f_y^{(l)} \frac{y_l}{z_l} + o_y^{(l)}$$

$$z_l u_l = f_x^{(l)} x_l + z_l o_x^{(l)} \qquad z_l v_l = f_y^{(l)} y_l + z_l o_y^{(l)}$$

Representing in matrix form:

$$z_l \begin{bmatrix} u_l \\ v_l \\ 1 \end{bmatrix} = \begin{bmatrix} z_l u_l \\ z_l v_l \\ z_l \end{bmatrix} = \begin{bmatrix} f_x^{(l)} x_l + z_l o_x^{(l)} \\ f_y^{(l)} y_l + z_l o_y^{(l)} \\ z_l \end{bmatrix} = \underbrace{\begin{bmatrix} f_x^{(l)} & 0 & o_x^{(l)} \\ 0 & f_y^{(l)} & o_y^{(l)} \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Known Camera Matrix } K_l} \begin{bmatrix} x_l \\ y_l \\ z_l \end{bmatrix}$$

Incorporating Image Coordinates

Left camera

$$z_l \begin{bmatrix} u_l \\ v_l \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f_x^{(l)} & 0 & o_x^{(l)} \\ 0 & f_y^{(l)} & o_y^{(l)} \\ 0 & 0 & 1 \end{bmatrix}}_{K_l} \begin{bmatrix} x_l \\ y_l \\ z_l \end{bmatrix}$$

(Calibration Matrix)

Right camera

$$z_r \begin{bmatrix} u_r \\ v_r \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f_x^{(r)} & 0 & o_x^{(r)} \\ 0 & f_y^{(r)} & o_y^{(r)} \\ 0 & 0 & 1 \end{bmatrix}}_{K_r} \begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix}$$

$$\boxed{\mathbf{x}_l{}^T = [u_l \quad v_l \quad 1] z_l\, {K_l^{-1}}^T}$$

$$\boxed{\mathbf{x}_r = K_r^{-1} z_r \begin{bmatrix} u_r \\ v_r \\ 1 \end{bmatrix}}$$

Incorporating the Image Coordinates

Epipolar constraint:

$$\begin{bmatrix} x_l & y_l & z_l \end{bmatrix} \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} = 0$$

Rewriting in terms of image coordinates:

$$\begin{bmatrix} u_l & v_l & 1 \end{bmatrix} \cancel{z_l}\, K_l^{-1^T} \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} K_r^{-1}\, \cancel{z_r} \begin{bmatrix} u_r \\ v_r \\ 1 \end{bmatrix} = 0$$

$$\boxed{\begin{matrix} z_l \neq 0 \\ z_r \neq 0 \end{matrix}}$$

z_l and z_r are the depths of the same scene point in the two cameras. The depths are not equal to zero.
Note: The center of the camera coordinate frame is placed at the effective pinhole of the camera itself.

Incorporating the Image Coordinates

Epipolar constraint:

$$[x_l \quad y_l \quad z_l] \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} = 0$$

Rewriting in terms of image coordinates:

$$\boxed{[u_l \quad v_l \quad 1]\, {K_l^{-1}}^T \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} K_r^{-1} \begin{bmatrix} u_r \\ v_r \\ 1 \end{bmatrix} = 0}$$

Fundamental Matrix F

Epipolar constraint:

$$[x_l \quad y_l \quad z_l] \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} = 0$$

Rewriting in terms of image coordinates:

$$[u_l \quad v_l \quad 1] \boxed{\begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}} \begin{bmatrix} u_r \\ v_r \\ 1 \end{bmatrix} = 0$$

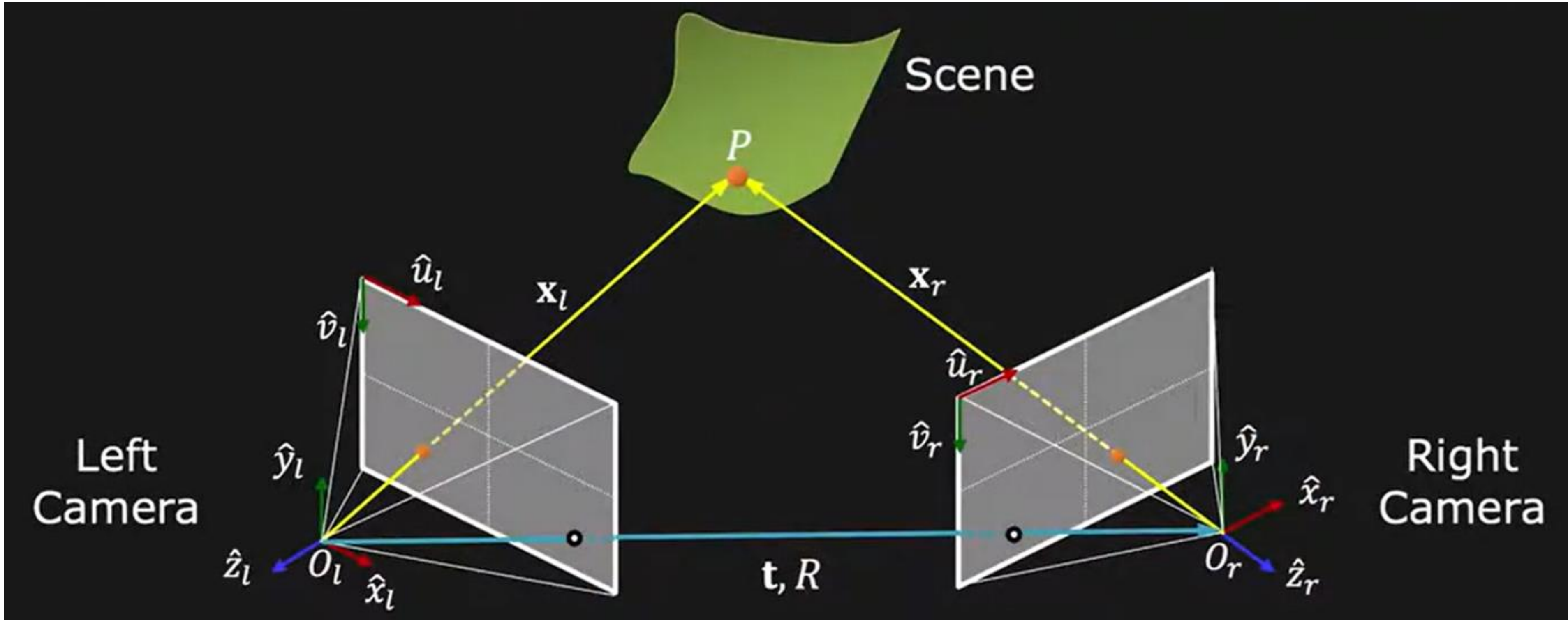
Fundamental Matrix F

$$\boxed{E = K_l^T F K_r} \qquad \boxed{E = T_\times R}$$

[Fagueras 1992, Luong 1992]

- We know that these image coordinate points (u_l , v_l) and (u_r , v_r) are from Left camera image right camera image respectively.
- From this equation, we can calculate the Fundamental Matrix
- Once we have Fundamental matrix, we can calculate the Essential Matrix as we know left and right Camera calibration matrices K_l and K_r
- Essential Matrix can be decomposed into Translation Matrix (T_x) and R (Rotation Matrix) using Singular Value Decomposition as T_x is a skew symmetric matrix and R is a orthonormal matrix.
- Now we know the relative position and orientation of both the cameras.
- Finally, We can search for corresponding points between two cameras

Computing Depth



Given the intrinsic parameters, the projections of scene point on the two image sensors are:

$$\begin{bmatrix} u_l \\ v_l \\ 1 \end{bmatrix} \equiv \begin{bmatrix} f_x^{(l)} & 0 & o_x^{(l)} & 0 \\ 0 & f_y^{(l)} & o_y^{(l)} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_l \\ y_l \\ z_l \\ 1 \end{bmatrix} \qquad \begin{bmatrix} u_r \\ v_r \\ 1 \end{bmatrix} \equiv \begin{bmatrix} f_x^{(r)} & 0 & o_x^{(r)} & 0 \\ 0 & f_y^{(r)} & o_y^{(r)} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ z_r \\ 1 \end{bmatrix}$$

Computing Depth

Left Camera Imaging Equation

$$\begin{bmatrix} u_l \\ v_l \\ 1 \end{bmatrix} \equiv \begin{bmatrix} f_x^{(l)} & 0 & o_x^{(l)} & 0 \\ 0 & f_y^{(l)} & o_y^{(l)} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_l \\ y_l \\ z_l \\ 1 \end{bmatrix}$$

Right Camera Imaging Equation

$$\begin{bmatrix} u_r \\ v_r \\ 1 \end{bmatrix} \equiv \begin{bmatrix} f_x^{(r)} & 0 & o_x^{(r)} & 0 \\ 0 & f_y^{(r)} & o_y^{(r)} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ z_r \\ 1 \end{bmatrix}$$

We also know the relative position and orientation between the two cameras.

$$\begin{bmatrix} x_l \\ y_l \\ z_l \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ z_r \\ 1 \end{bmatrix}$$

Computing Depth

Left Camera Imaging Equation:

$$\begin{bmatrix} u_l \\ v_l \\ 1 \end{bmatrix} \equiv \begin{bmatrix} f_x^{(l)} & 0 & o_x^{(l)} & 0 \\ 0 & f_y^{(l)} & o_y^{(l)} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ z_r \\ 1 \end{bmatrix}$$

$$\boxed{\tilde{\mathbf{u}}_l = P_l\, \tilde{\mathbf{x}}_r}$$

Right Camera Imaging Equation:

$$\begin{bmatrix} u_r \\ v_r \\ 1 \end{bmatrix} \equiv \begin{bmatrix} f_x^{(r)} & 0 & o_x^{(r)} & 0 \\ 0 & f_y^{(r)} & o_y^{(r)} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ z_r \\ 1 \end{bmatrix}$$

$$\boxed{\tilde{\mathbf{u}}_r = M_{int_r}\, \tilde{\mathbf{x}}_r}$$

Computing Depth

The imaging equations:

$$\tilde{\mathbf{u}}_r = M_r \, \tilde{\mathbf{x}}_r \qquad\qquad \tilde{\mathbf{u}}_l = P_l \, \tilde{\mathbf{x}}_r$$

$$\underbrace{\begin{bmatrix} u_r \\ v_r \\ 1 \end{bmatrix} \equiv \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix}}_{\text{Known}} \underbrace{\begin{bmatrix} x_r \\ y_r \\ z_r \\ 1 \end{bmatrix}}_{\text{Unknown}} \qquad \underbrace{\begin{bmatrix} u_l \\ v_l \\ 1 \end{bmatrix} \equiv \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}}_{\text{Known}} \underbrace{\begin{bmatrix} x_r \\ y_r \\ z_r \\ 1 \end{bmatrix}}_{\text{Unknown}}$$

Rearranging the terms:

$$\begin{bmatrix} u_r m_{31} - m_{11} & u_r m_{32} - m_{12} & u_r m_{33} - m_{13} \\ v_r m_{31} - m_{21} & v_r m_{32} - m_{22} & v_r m_{33} - m_{23} \\ u_l p_{31} - p_{11} & u_l p_{32} - p_{12} & u_l p_{33} - p_{13} \\ v_l p_{31} - p_{21} & v_l p_{32} - p_{22} & v_l p_{33} - p_{23} \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} = \begin{bmatrix} m_{14} - m_{34} \\ m_{24} - m_{34} \\ p_{14} - p_{34} \\ p_{24} - p_{34} \end{bmatrix}$$

Computing Depth: Least Squares Solution

$$\underbrace{\begin{bmatrix} u_r m_{31} - m_{11} & u_r m_{32} - m_{12} & u_r m_{33} - m_{13} \\ v_r m_{31} - m_{21} & v_r m_{32} - m_{22} & v_r m_{33} - m_{23} \\ u_l p_{31} - p_{11} & u_l p_{32} - p_{12} & u_l p_{33} - p_{13} \\ v_l p_{31} - p_{21} & v_l p_{32} - p_{22} & v_l p_{33} - p_{23} \end{bmatrix}}_{\substack{A_{4\times3} \\ \text{(Known)}}} \underbrace{\begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix}}_{\substack{\mathbf{x}_r \\ \text{(Unknown)}}} = \underbrace{\begin{bmatrix} m_{14} - m_{34} \\ m_{24} - m_{34} \\ p_{14} - p_{34} \\ p_{24} - p_{34} \end{bmatrix}}_{\substack{\mathbf{b}_{4\times1} \\ \text{(Known)}}}$$

We have a set of Over determined system of linear equations.

Find least squares solution using pseudo-inverse:

$$A\mathbf{x}_r = \mathbf{b}$$

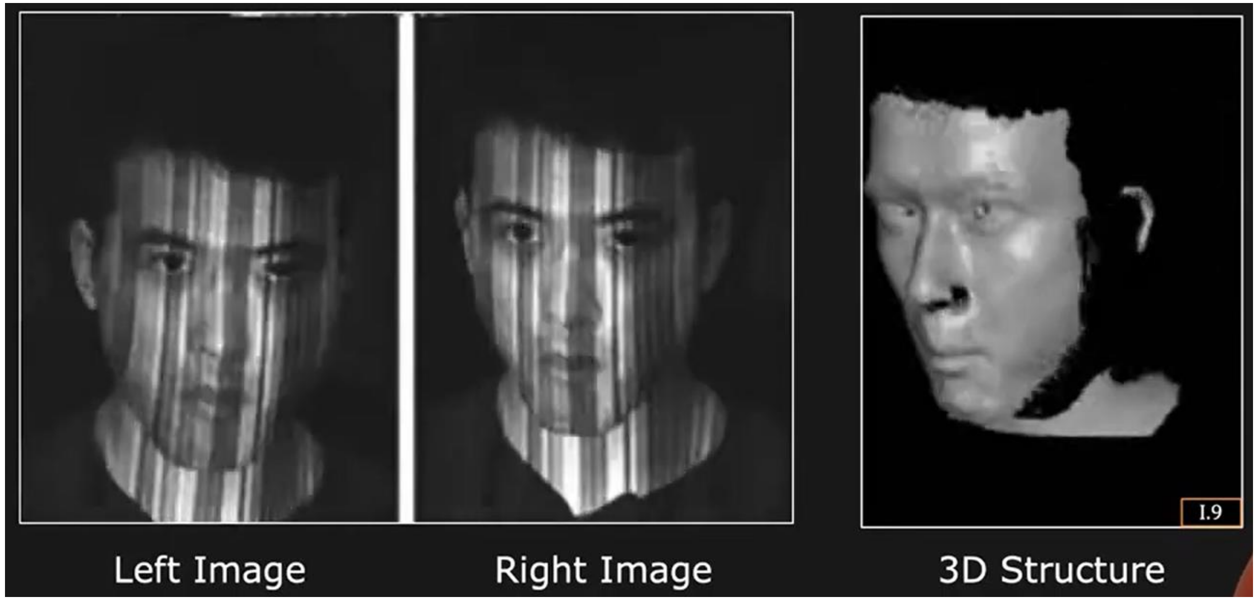
$$A^T A\mathbf{x}_r = A^T\mathbf{b}$$

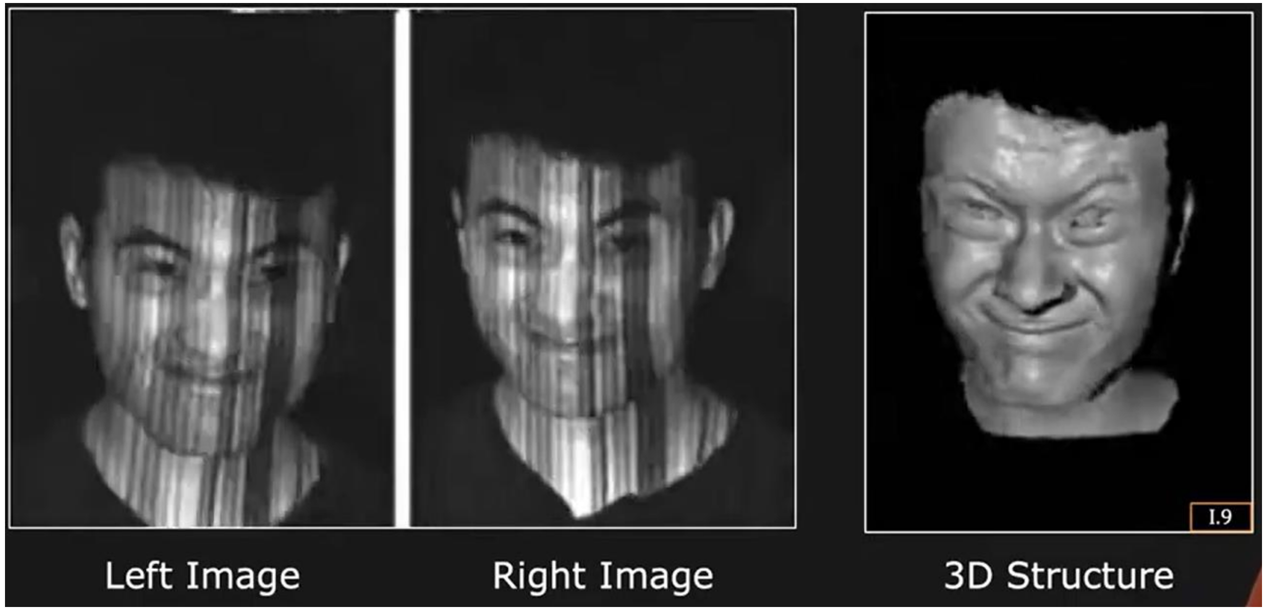
$$\boxed{\mathbf{x}_r = (A^T A)^{-1} A^T\mathbf{b}}$$

Now, we know how to calculate depth maps of the scene from arbitrary views through this calibration process

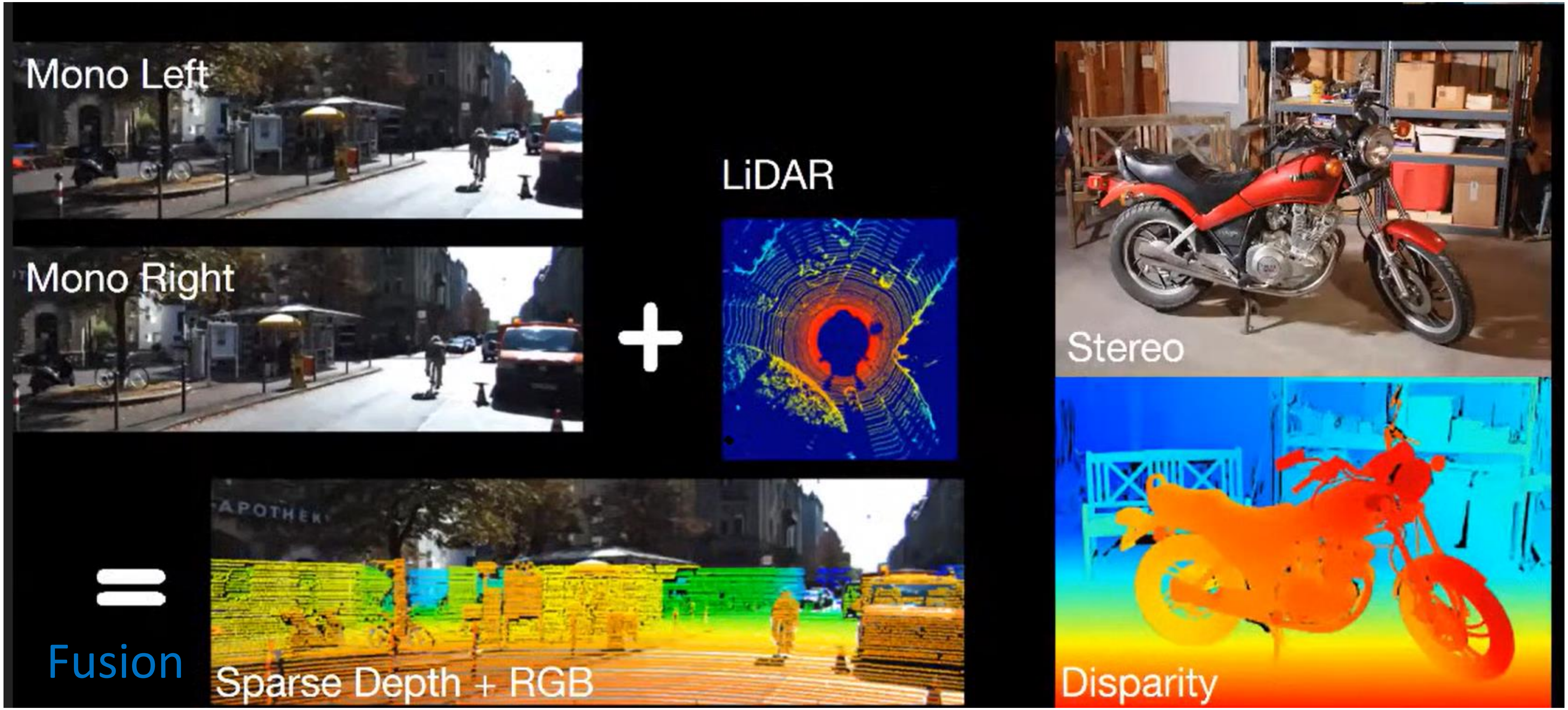
3D reconstruction with Internet images

Real time reconstruction





Mono-Stereo-Time modalities



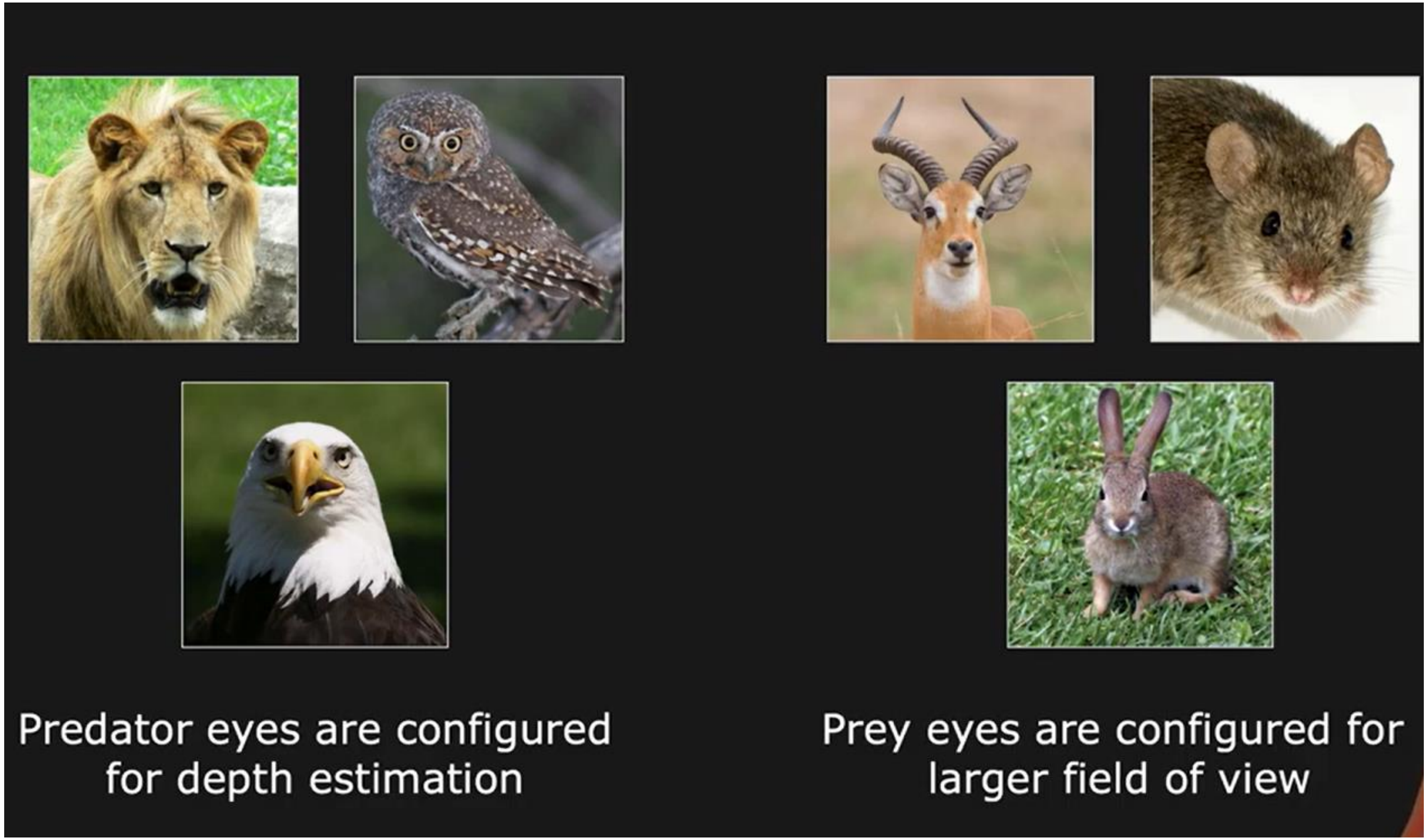
Disparity Map: Coding depth information in an image

Closers pixels: reddish

Far pixels: bluish

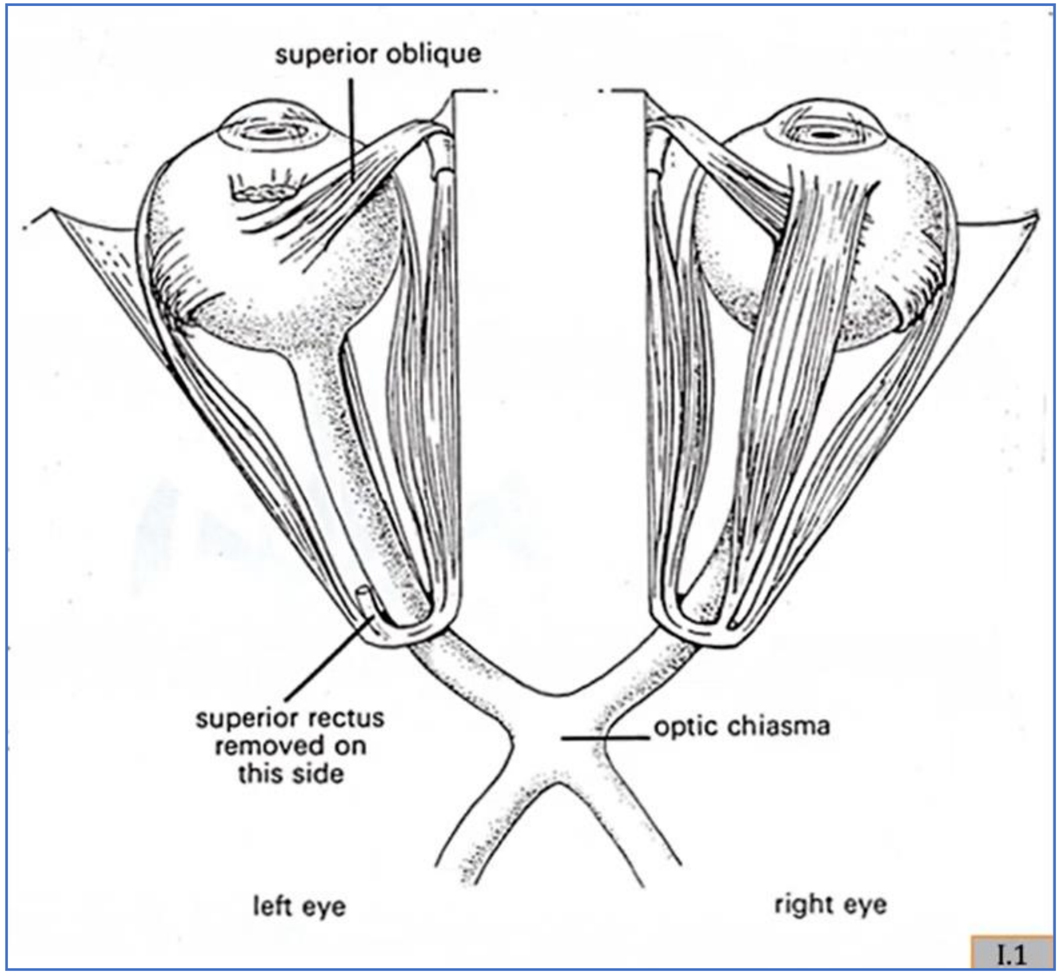
Stereopsis: Stereo in Nature

Stereopsis

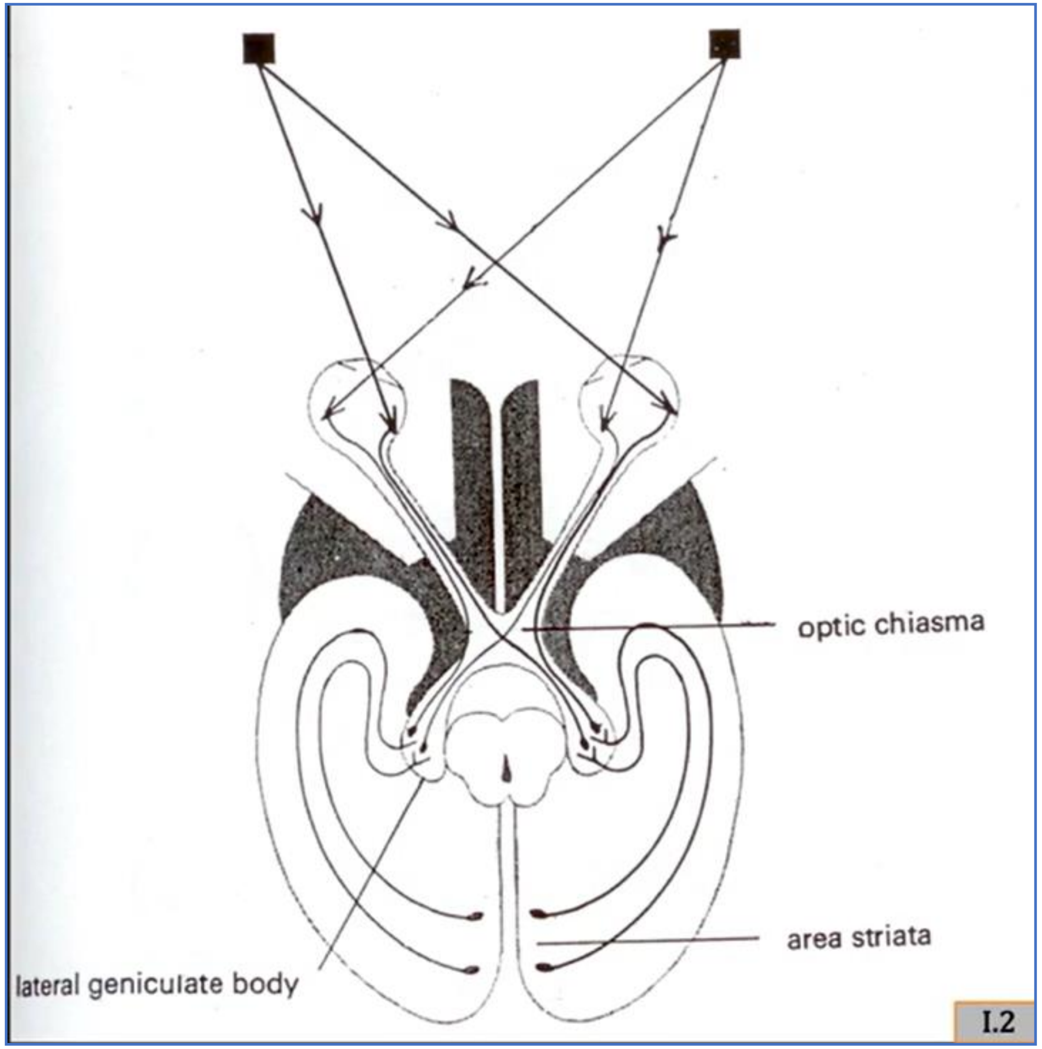


Predators: The reason is you want the overlap in the fields of view to be able to do stereo and then depth perception

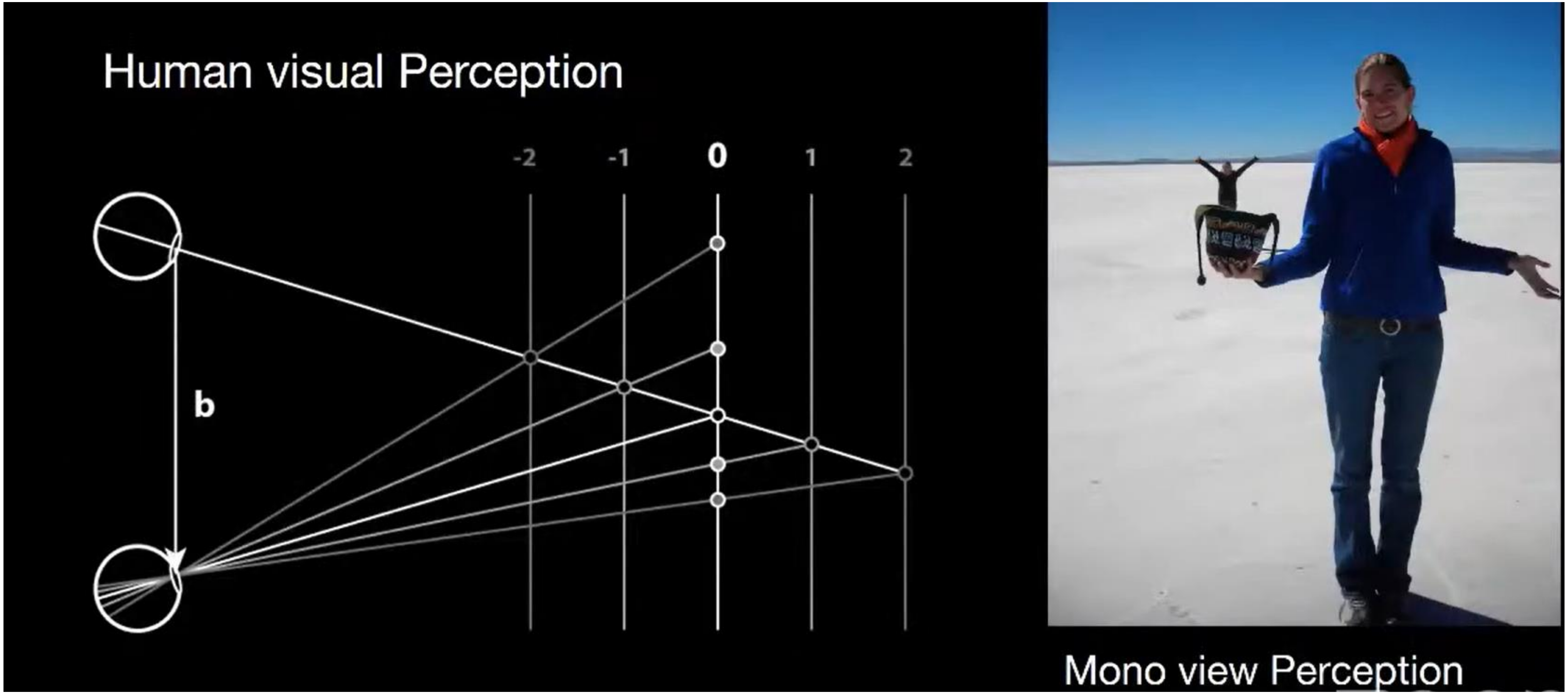
Stereopsis in Humans



Inter ocular distance : 64 mm (adults)



Human Visual Perception: Image is a 2D projection of a 3D scene



References

- Camera Calibration | Uncalibrated Stereo (Entire Playlist): by Shree K Nayar Columbia University
- Camera Calibration: Uncalibrated Stereo: by Shree K Nayar Columbia University
- Epipolar Geometry | Uncalibrated Stereo: by Shree K Nayar Columbia University
- Computing Depth | Uncalibrated Stereo: by Shree K Nayar Columbia University
- https://en.wikipedia.org/wiki/Epipolar_geometry
- Sensing Depth with 3D Computer Vision - Dr. Benjamin Busam

- https://robotacademy.net.au/masterclass/2d-geometry/?lesson=69   (Good one)
- https://robotacademy.net.au/masterclass/3d-geometry/ (Good one)