Computer Vision

Kumar Bipin

BE, MS, PhD (MMTU, IISc, IIIT-Hyderabad) (Robotics Control and Computer Vision)

Motorola, STMicroeletronics, Tata Elxsi (Technical Manager)

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

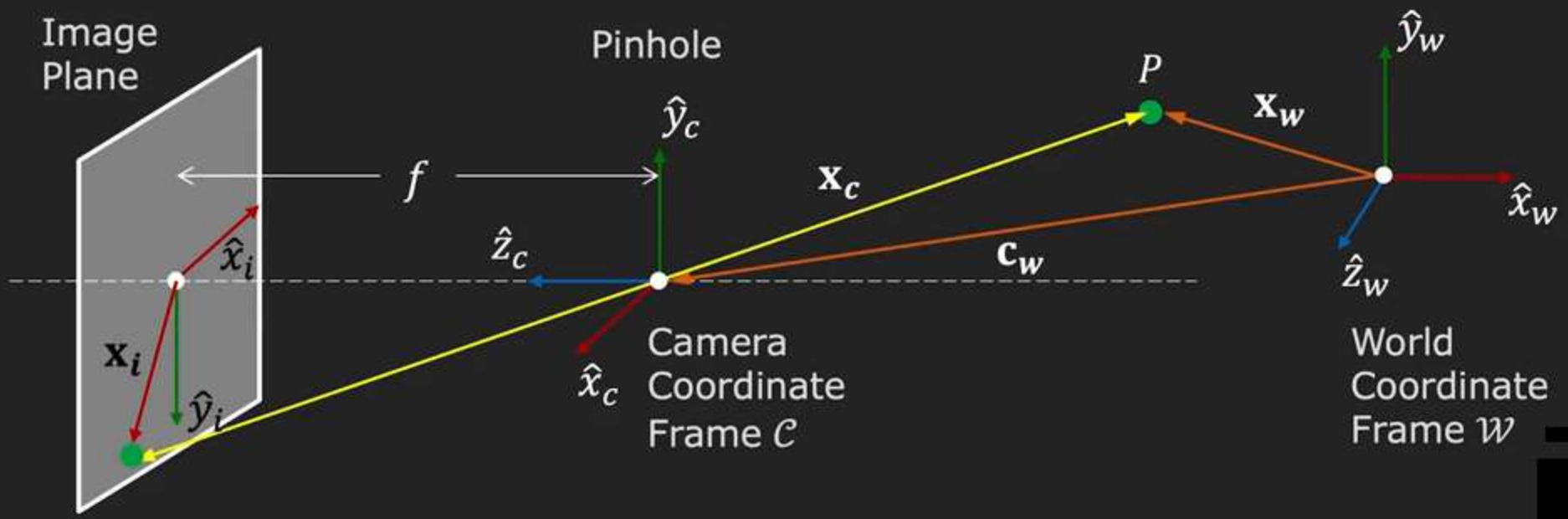


Image Coordinates

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

Perspective Projection

Camera Coordinates

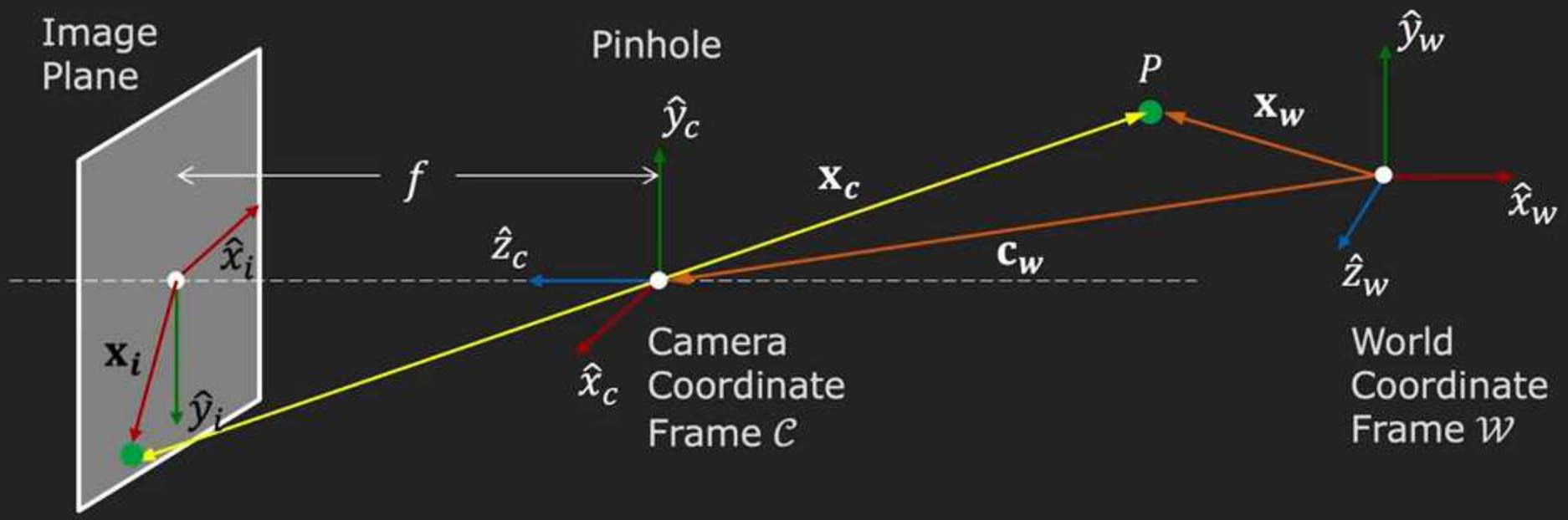
$$\mathbf{x}_c = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$$

Coordinate Transformation

World Coordinates

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

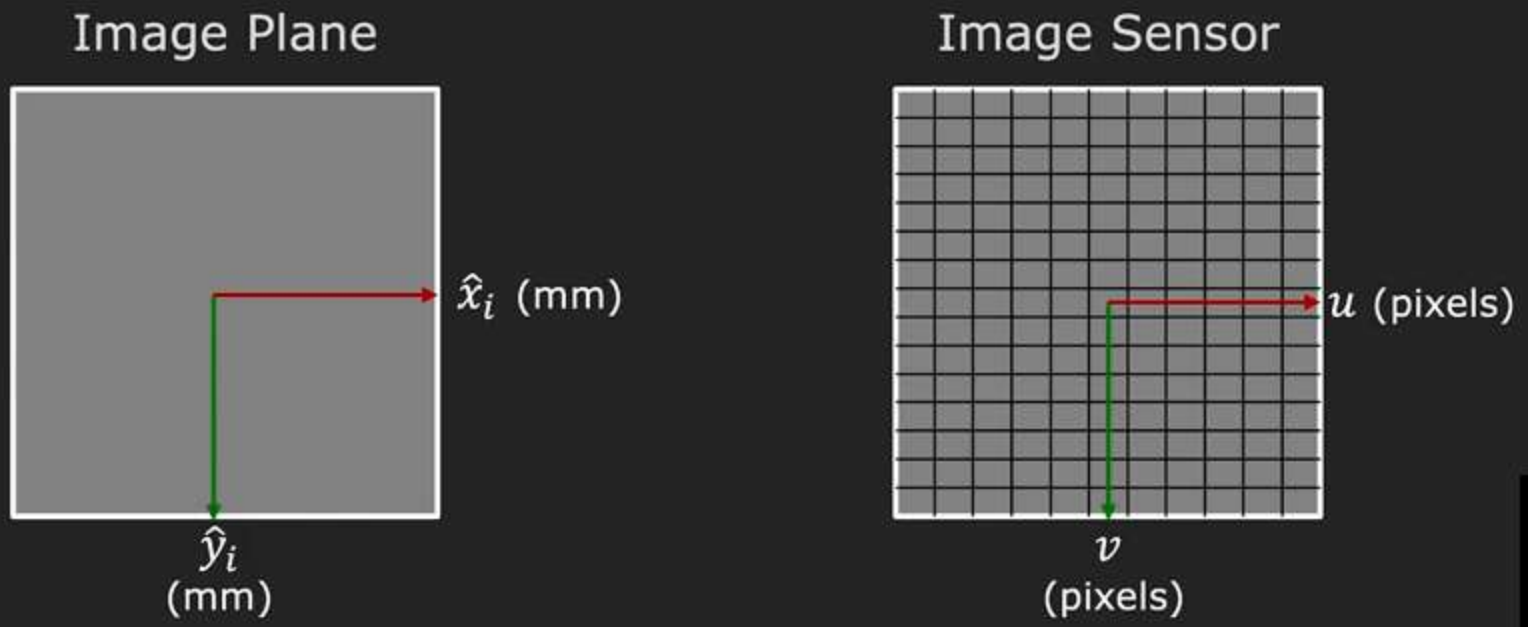
Perspective Projection



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

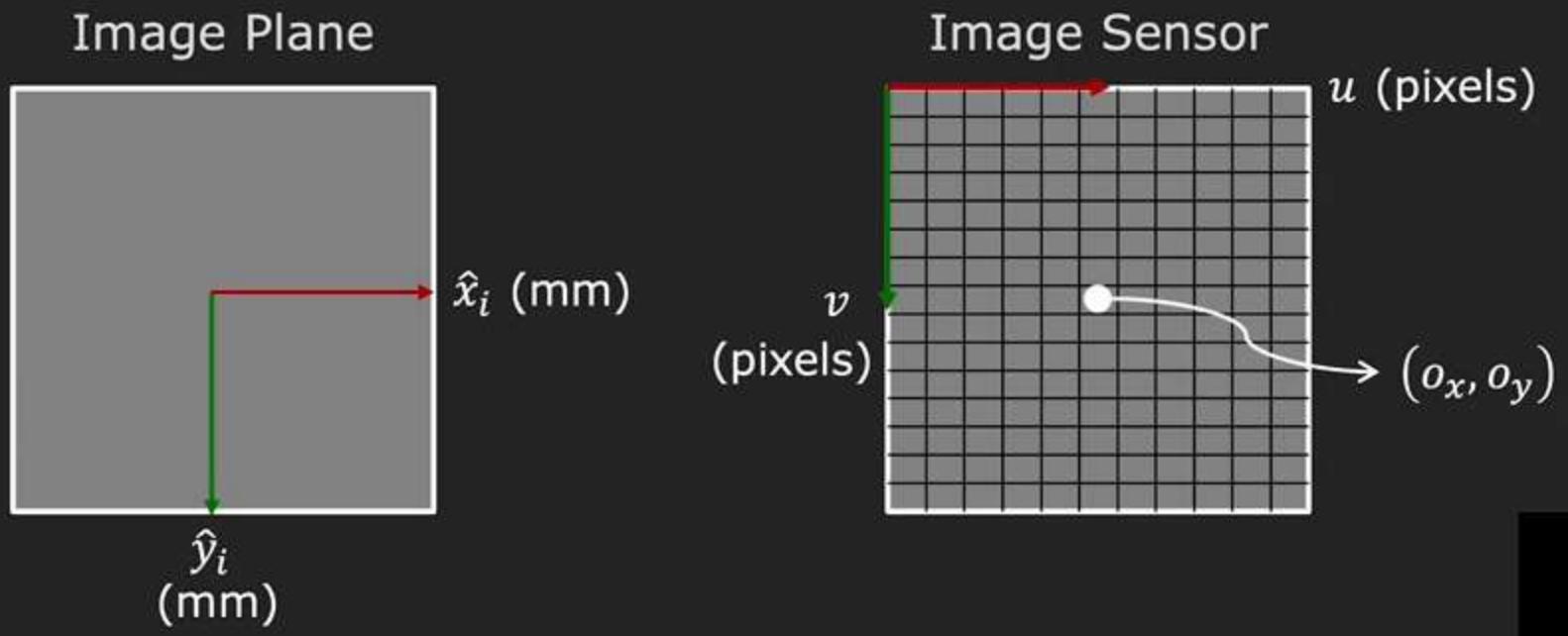


Pixels may be rectangular.

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

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

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

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

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

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

Equations for perspective projection are Non-Linear.

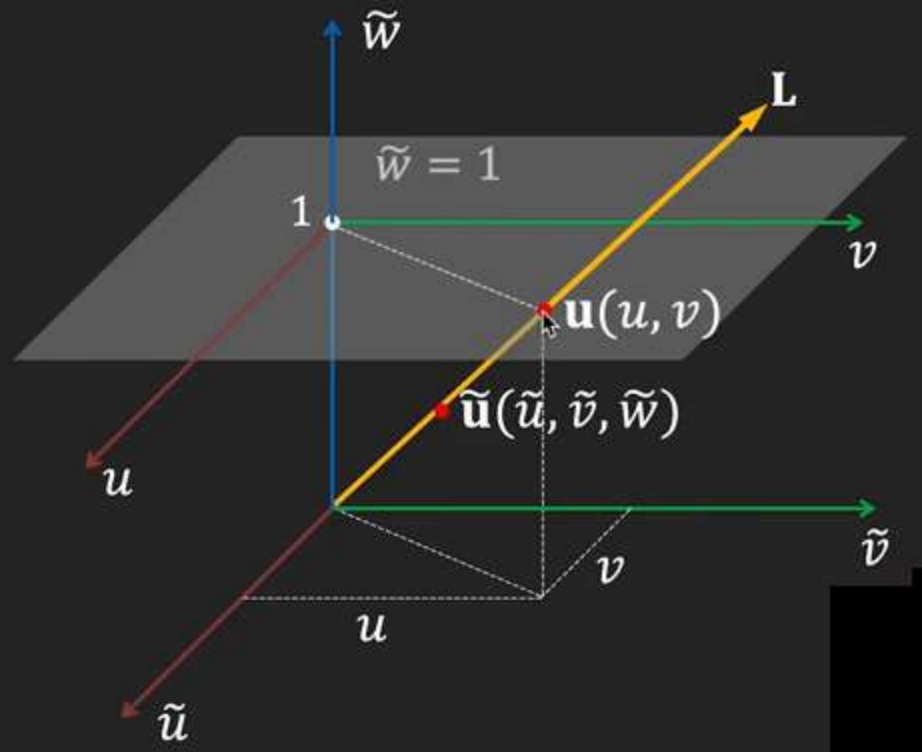
It is convenient to express them as linear equations.

Homogenous Coordinates

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

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

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



Every point on line $\mathbf{L}$ (except origin) repres
the homogenous coordinate of $\mathbf{u}(u, v)$

Homogenous Coordinates

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:

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

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

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):

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

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

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:

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

$$\tilde{\mathbf{u}} = [K|0]\,\tilde{\mathbf{x}}_c = M_{int}\,\tilde{\mathbf{x}}_c$$

Forward Imaging Model: 3D to 2D

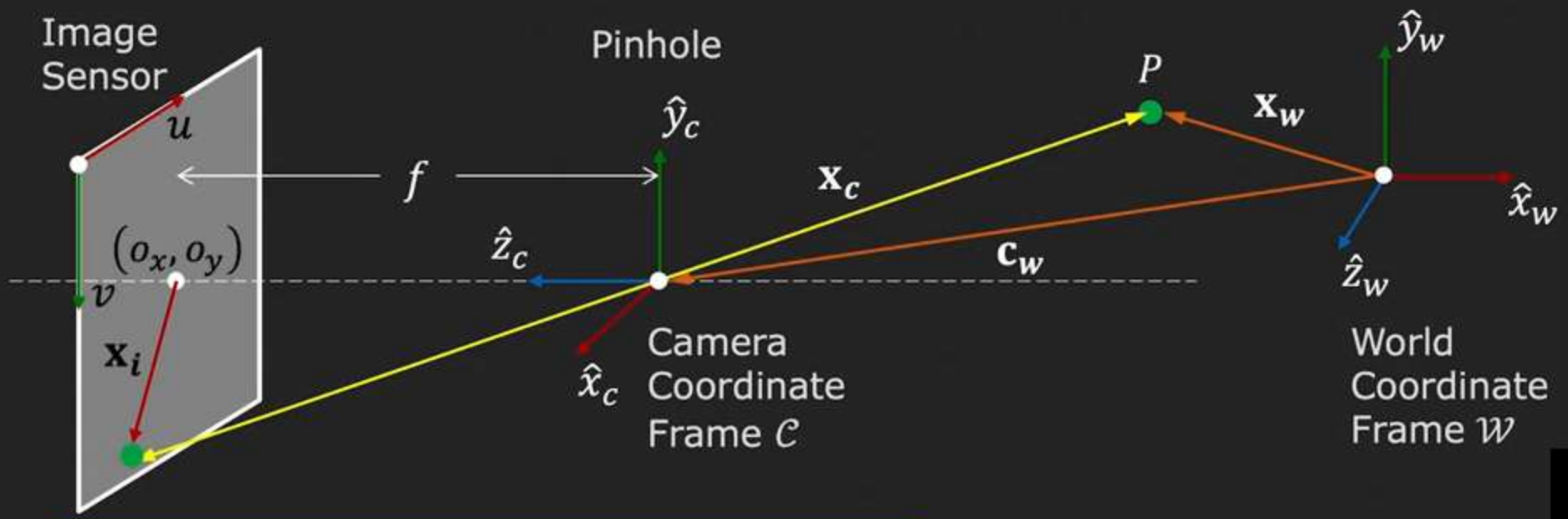


Image Coordinates

$$\mathbf{u} = \begin{bmatrix} u \\ v \end{bmatrix}$$

M_{int}

Perspective Projection

Camera Coordinates

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

Coordinate Transformation

World Coordinates

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

Forward Imaging Model: 3D to 2D

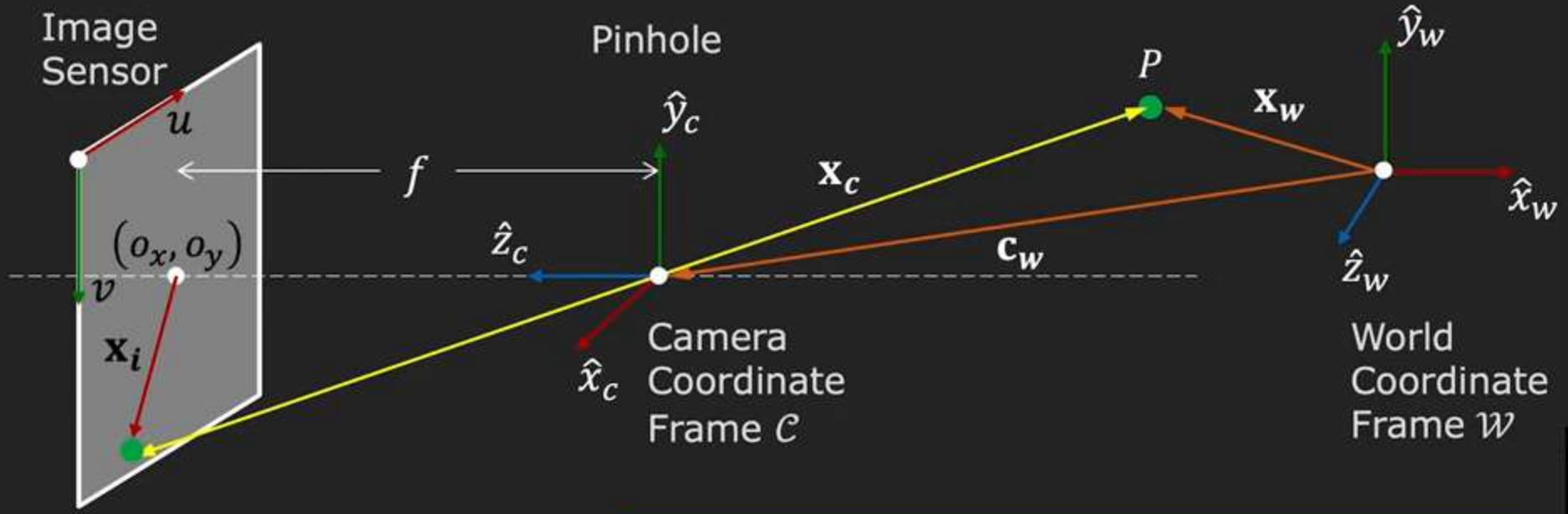


Image Coordinates

$$\mathbf{u} = \begin{bmatrix} u \\ v \end{bmatrix}$$

M_{int}

Perspective Projection

Camera Coordinates

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

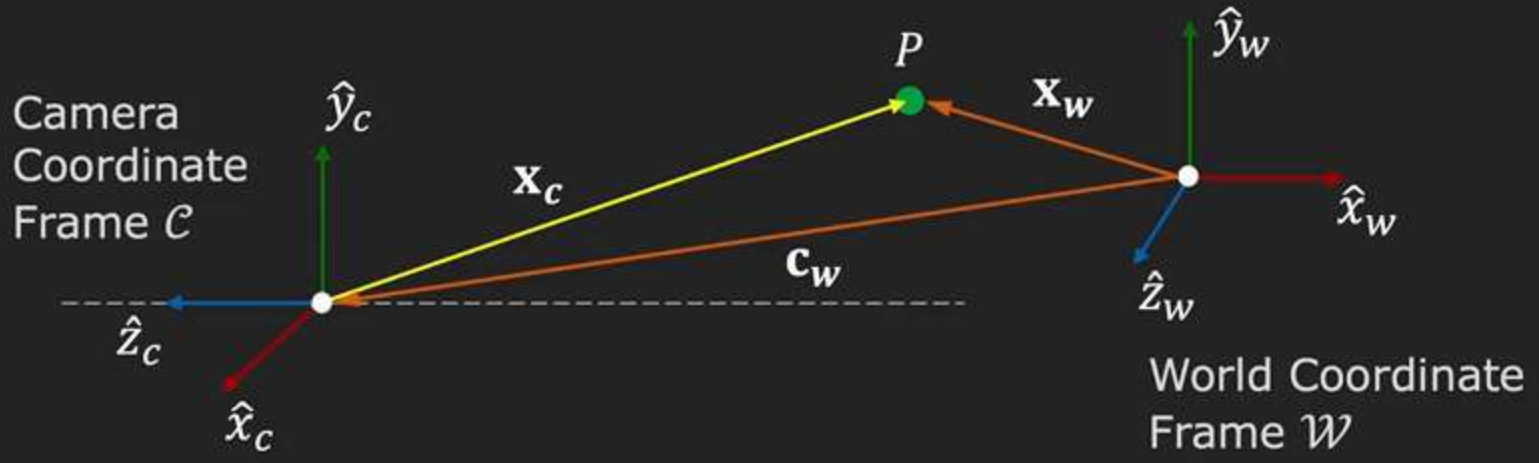
?

Coordinate Transformation

World Coordinates

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

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.

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

→ 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 $\mathbf{u}$ and $\mathbf{v}$ are orthonormal if and only if:

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

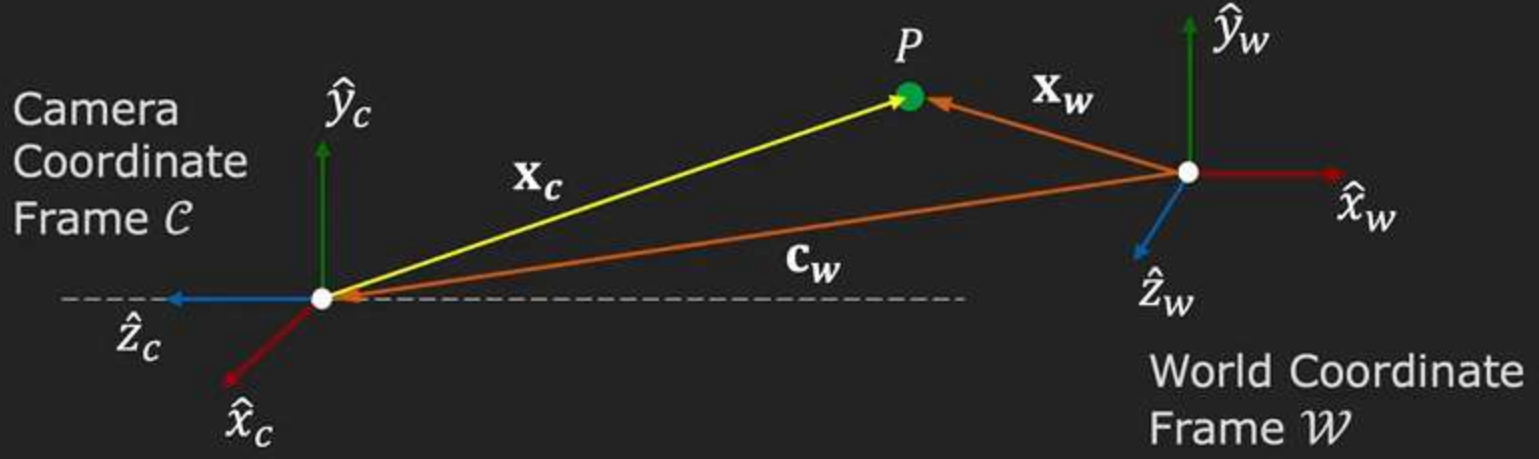
(Orthogonality) (Unit length)

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:

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

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:

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

$$\mathbf{t} = -R\mathbf{c}_w$$

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

Extrinsic Matrix

Rewriting using homogenous coordinates:

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

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

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

Forward Imaging Model: 3D to 2D

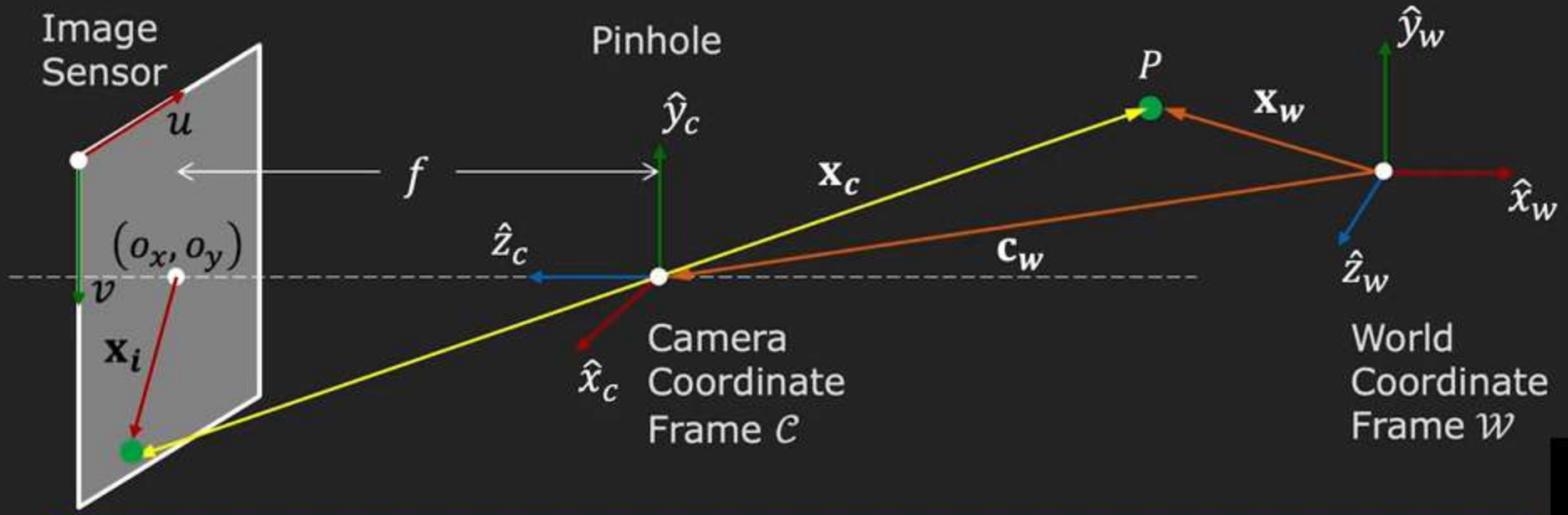


Image Coordinates		Camera Coordinates		World Coordinates
$\mathbf{u} = \begin{bmatrix} u \\ v \end{bmatrix}$	$\xleftarrow{M_{int}}$ Perspective Projection	$\mathbf{x}_c = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$	$\xleftarrow{M_{ext}}$ Coordinate Transformation	$\mathbf{x}_w = \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix}$

Projection Matrix P

Camera to Pixel

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

$$\tilde{\mathbf{u}} = M_{int}\, \tilde{\mathbf{x}}_{\boldsymbol{c}}$$

World to Camera

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

$$\tilde{\mathbf{x}}_{\boldsymbol{c}} = M_{ext} \tilde{\mathbf{x}}_{\boldsymbol{w}}$$

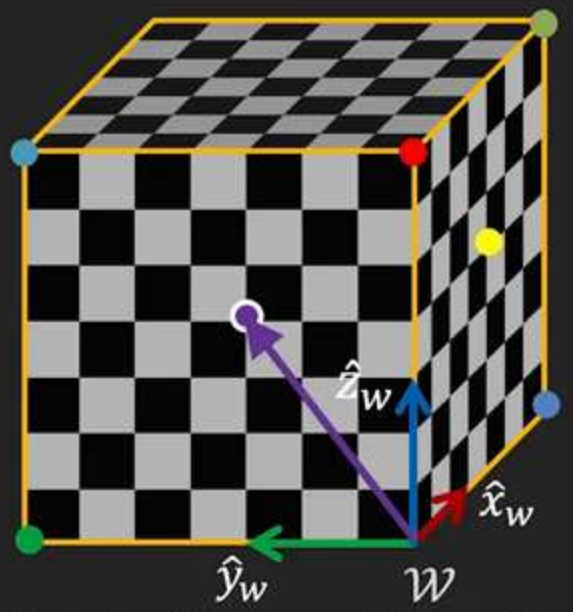
Combining the above two equations, we get the full projection matrix P:

$$\tilde{\mathbf{u}} = M_{int} M_{ext}\, \tilde{\mathbf{x}}_{\boldsymbol{w}} = P\, \tilde{\mathbf{x}}_{\boldsymbol{w}}$$

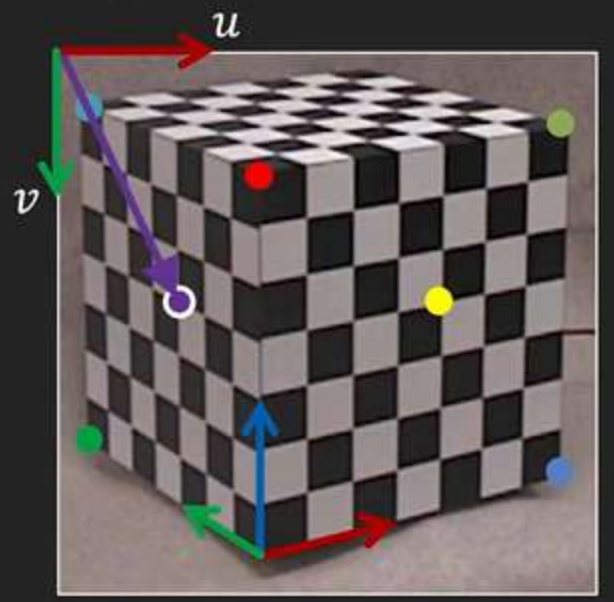
$$\begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

Camera Calibration Procedure

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



Object of Known Geometry



Captured Image

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

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

Step 4: Rearranging the terms

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

Known

Unknown

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

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

Scale of Projection Matrix

Projection matrix acts on homogenous coordinates.

We know that:

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

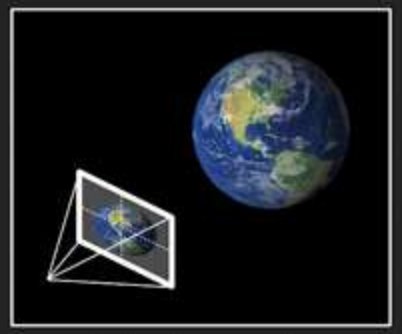
That is:

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

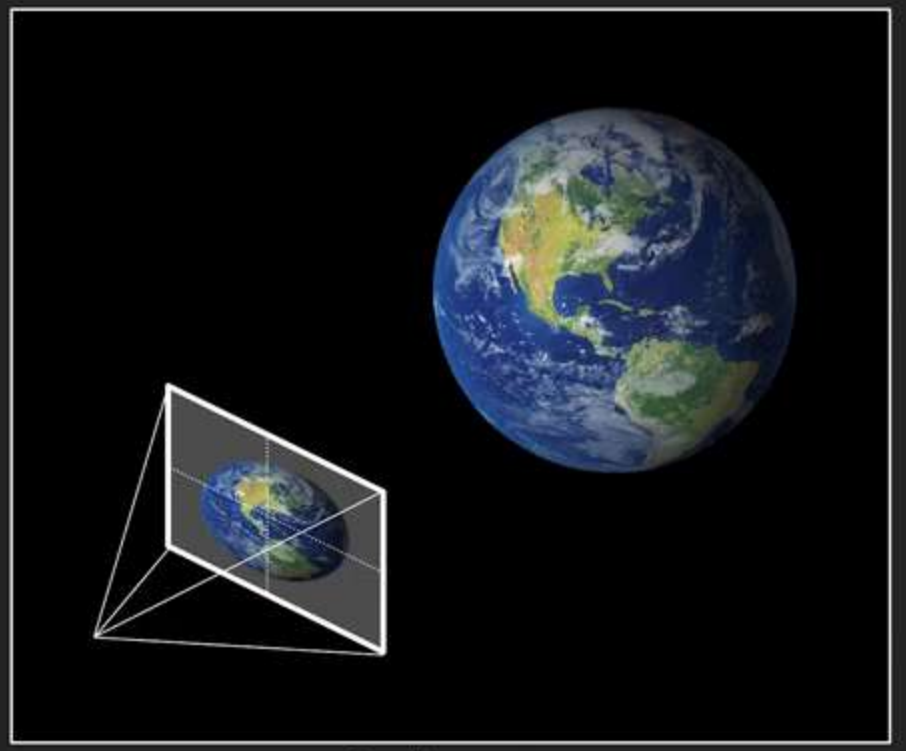
Therefore, Projection Matrices P and kP produce the same homogenous pixel coordinates.

Projection Matrix P is defined only up to a scale.

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: $\boxed{\|\mathbf{p}\|^2 = 1}$

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

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

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

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 $\mathbf{p}$: $2A^TA\mathbf{p} - 2\lambda\mathbf{p} = \mathbf{0}$

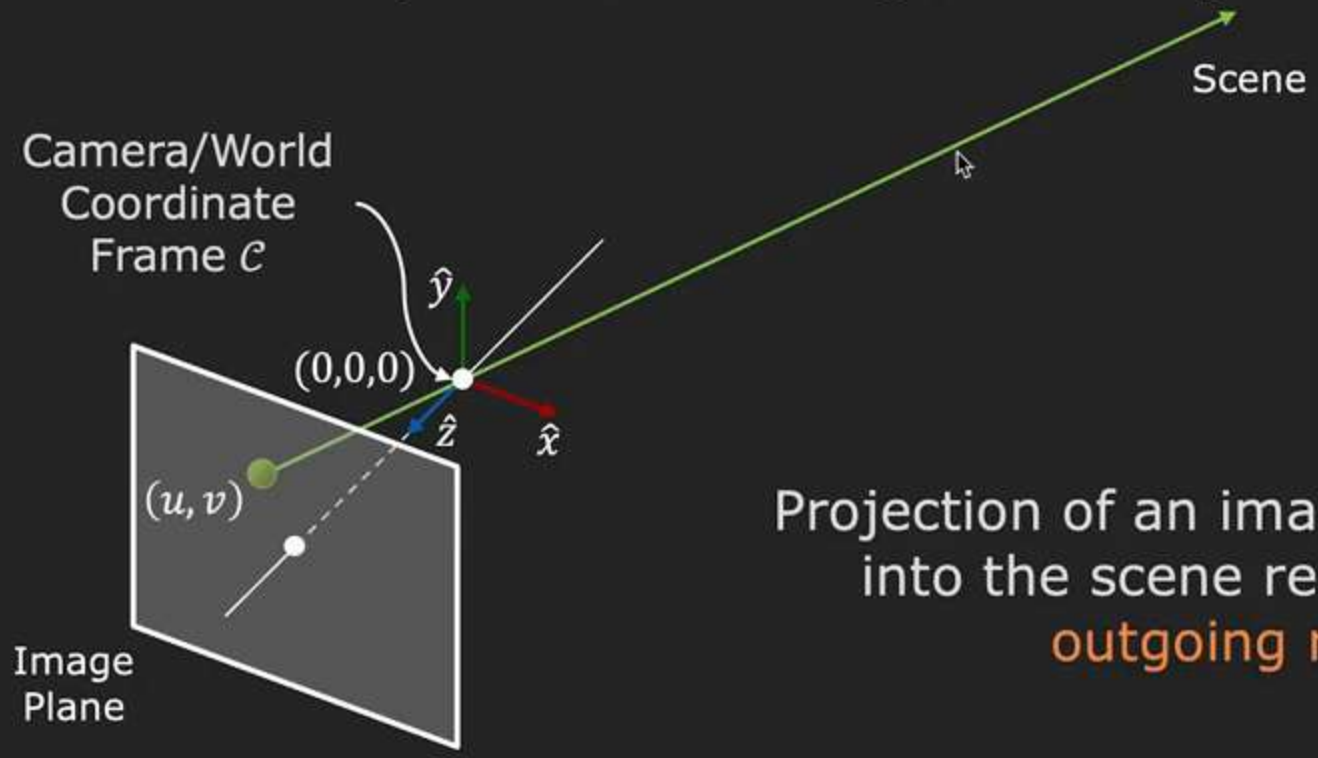
$$\boxed{A^TA\mathbf{p} = \lambda\mathbf{p}}$$ Eigenvalue Problem

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

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

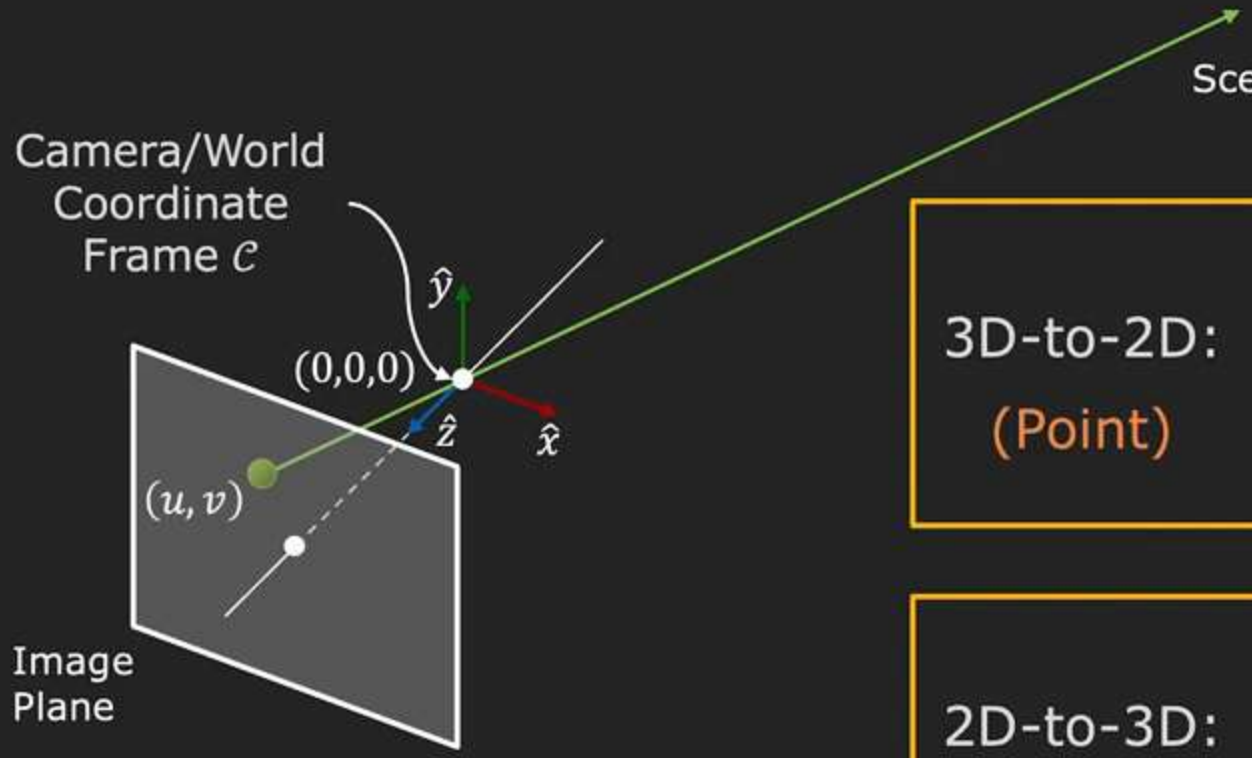
Backward Projection: From 2D to 3D

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)

$$x = {}^{z}/_{f_x}\,(u - o_x)$$

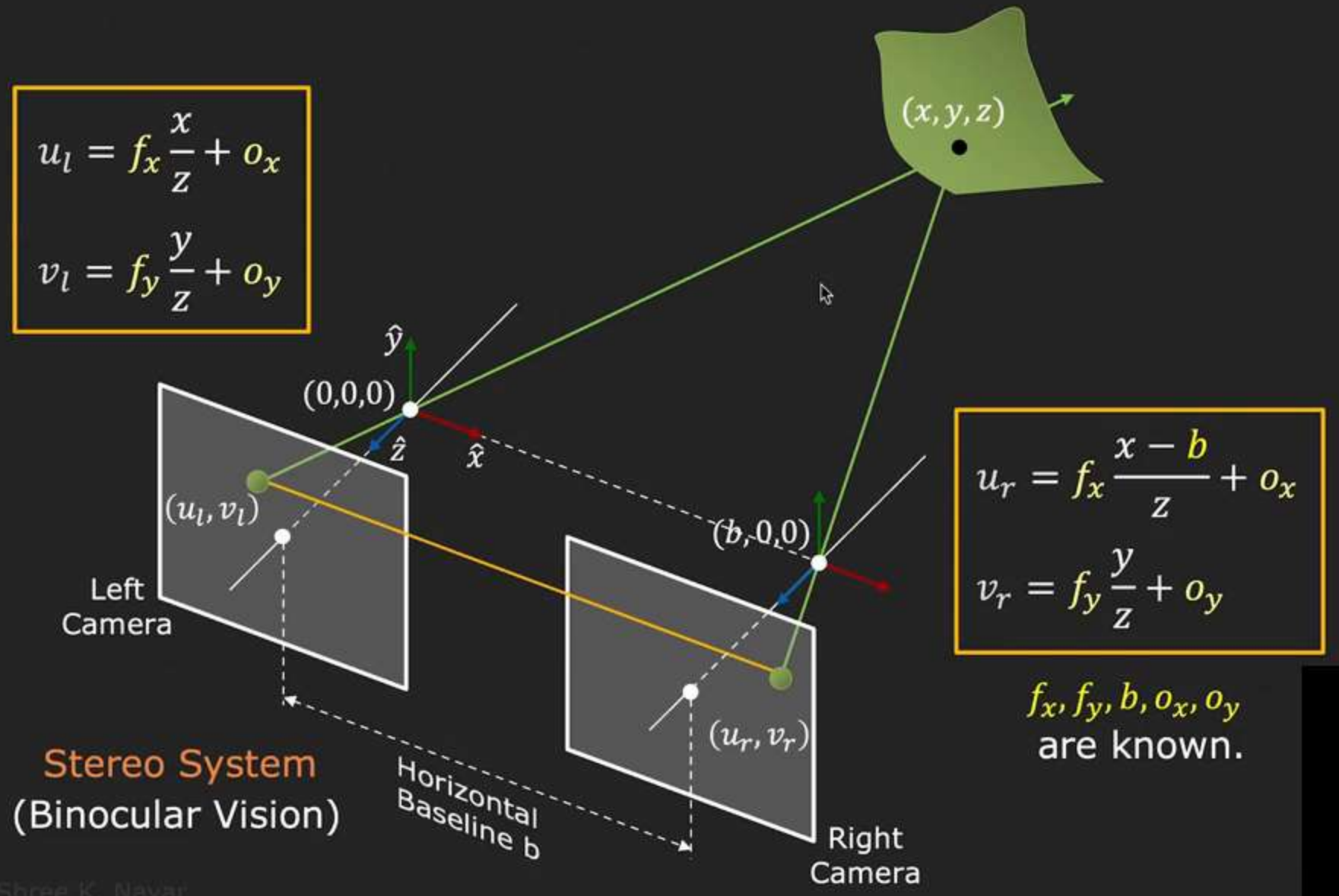
$$y = {}^{z}/_{f_y}\,(v - o_y$$

$$z > 0$$

Triangulation using Two Cameras

$$u_l = f_x \frac{x}{z} + o_x$$

$$v_l = f_y \frac{y}{z} + o_y$$



$$u_r = f_x \frac{x - b}{z} + o_x$$

$$v_r = f_y \frac{y}{z} + o_y$$

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

Stereo System
(Binocular Vision)

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):

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

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

Depth z is inversely proportional to Disparity.

Disparity is proportional to Baseline.

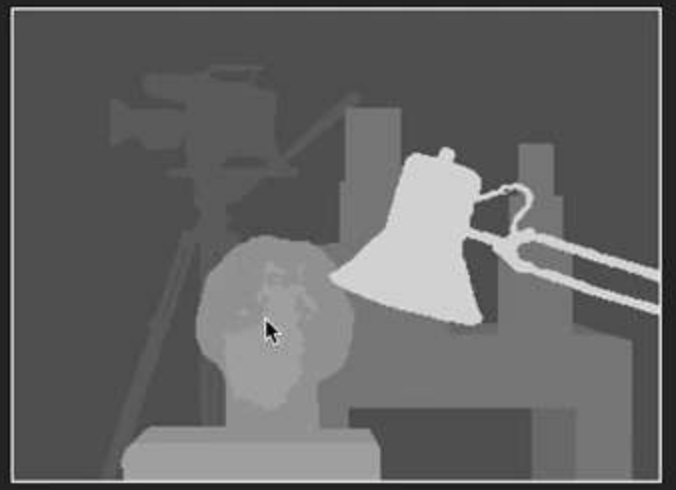
A Simple Stereo Camera

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)

Stereo Matching: Finding Disparities

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

Left/Right Camera Images

Disparity Map (Ground Truth)

Stereo Matching: Finding Disparities

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

Left/Right Camera Images

Disparity Map (Ground Truth)

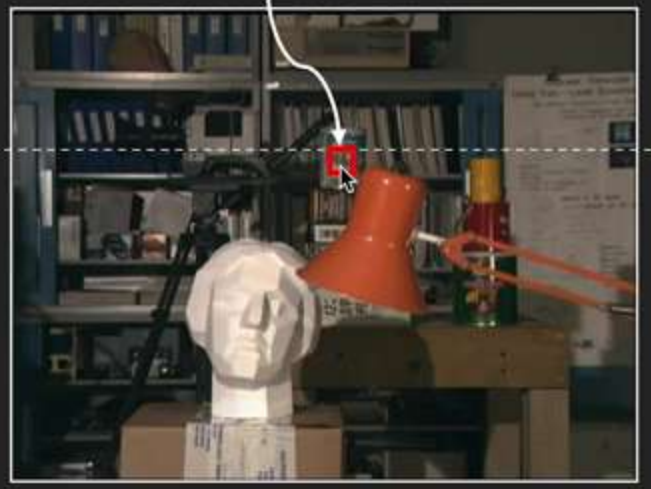
From perspective projection: $v_l = v_r = f_y \frac{y}{z} + o_y$

Corresponding scene points lie on the same horizontal scan line.

Window Based Methods

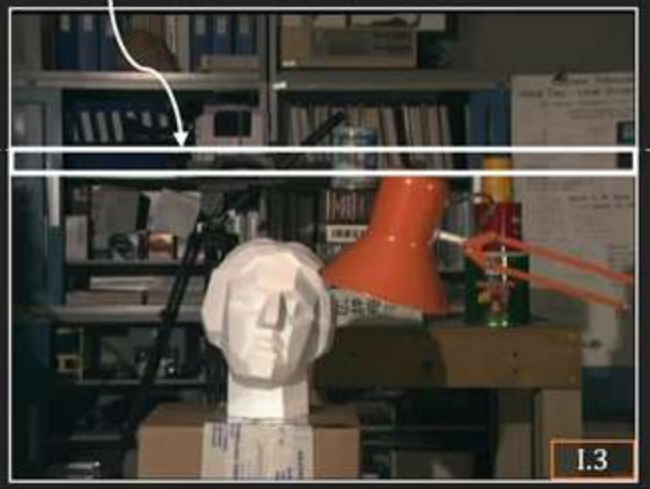
Determine Disparity using Template Matching

Template Window T



Left Camera Image E_l

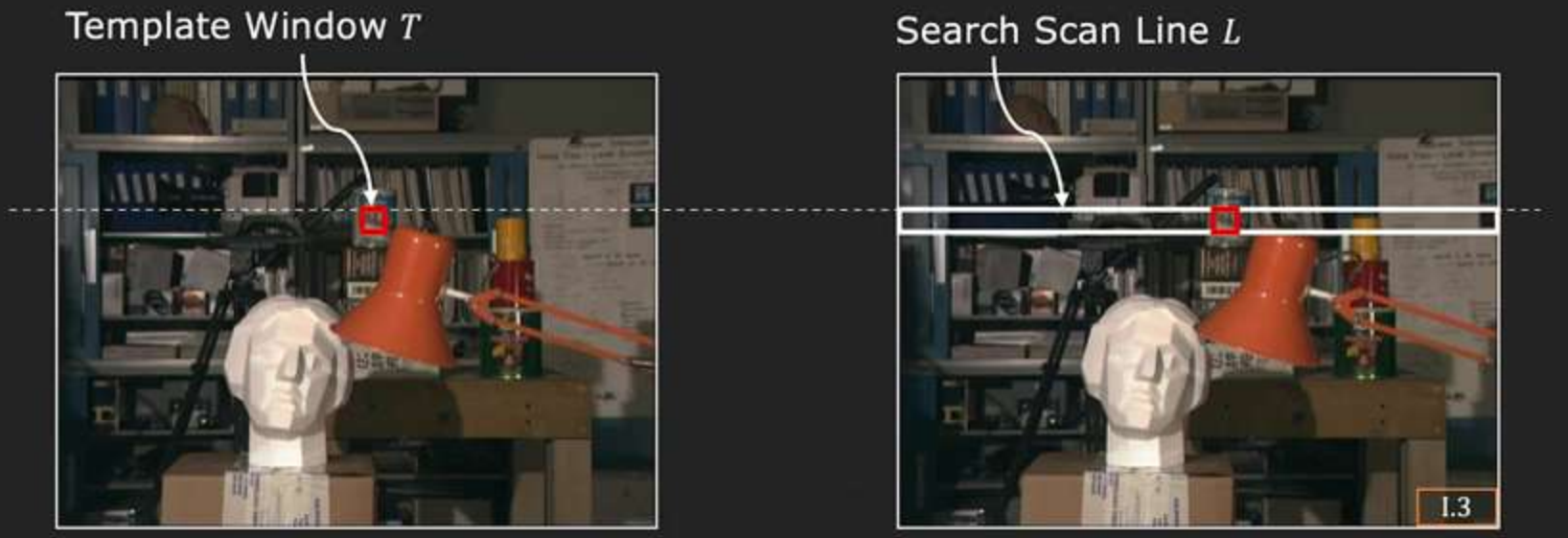
Search Scan Line L



Right Camera Image E_r

Window Based Methods

Determine Disparity using Template Matching



Left Camera Image E_l Right Camera Image E_r

Disparity: $d = u_l - u_r$ Depth: $z = \frac{bf_x}{(u_l - u_r)}$

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:

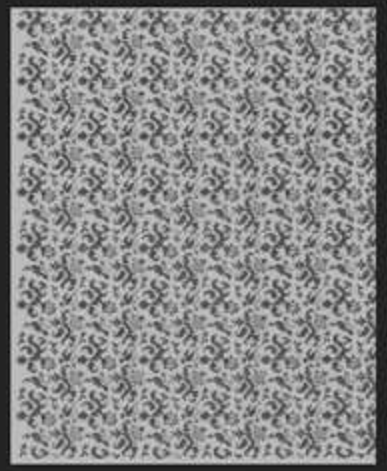
$$SSD(k,l) = \sum_{(i,j)\in T} |E_l(i,j) - E_r(i+k, j+l)|^2$$

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

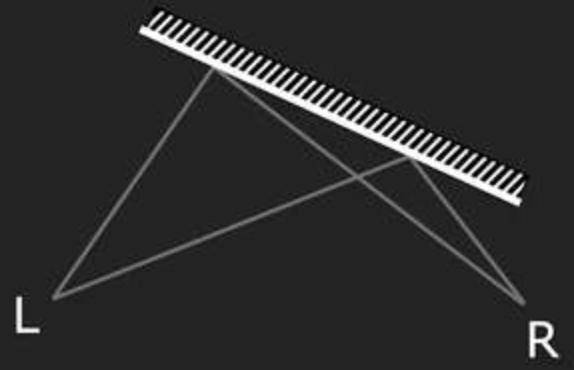
$$NCC(k,l) = \frac{\sum_{(i,j)\in T} E_l(i,j)\, E_r(i+k, j+l)}{\sqrt{\sum_{(i,j)\in T} E_l(i,j)^2 \sum_{(i,j)\in T} E_r(i+k, j+l)^2}}$$

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 Truth

SSD (Window size=21)

SSD – Adaptive Window

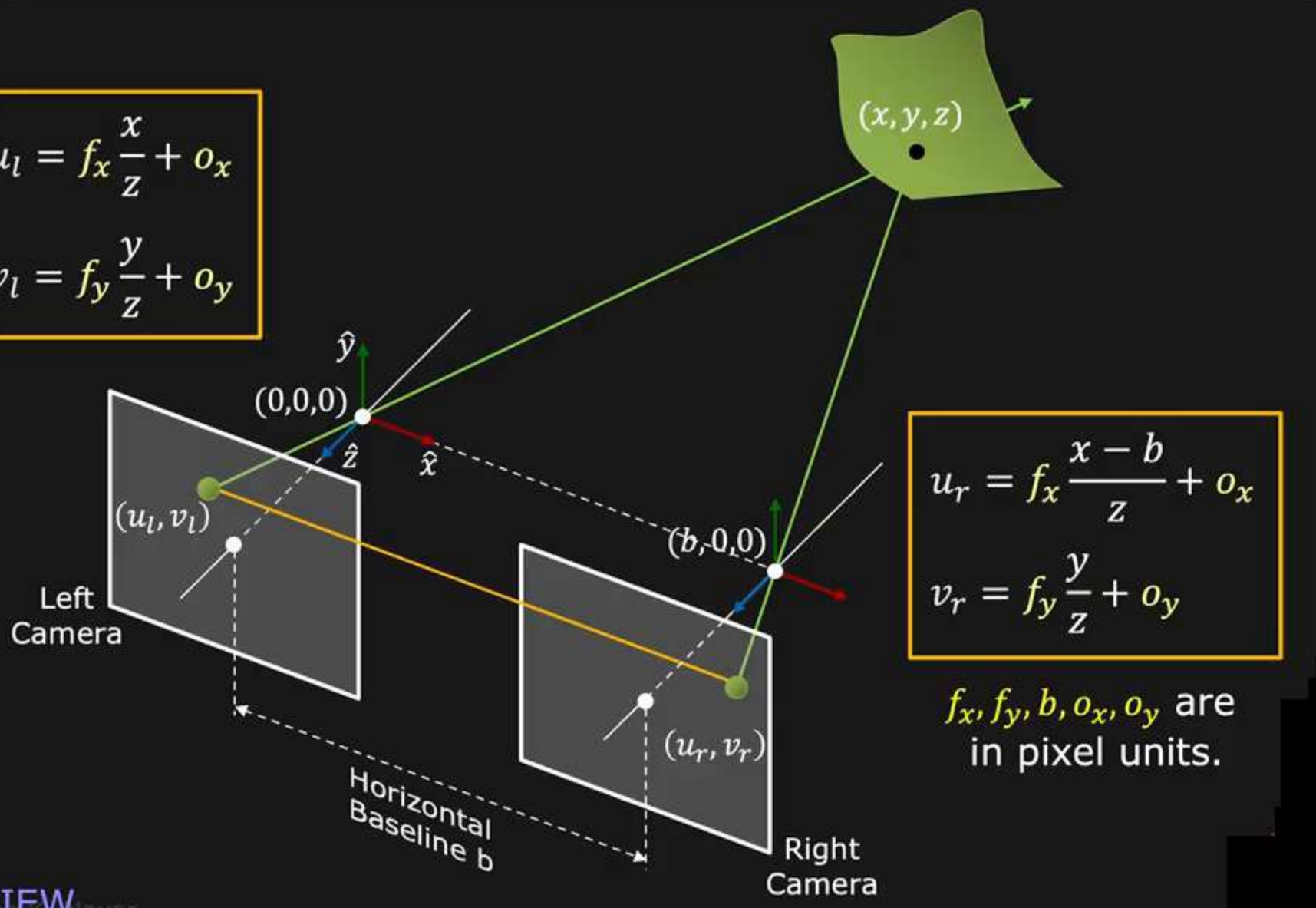
State of the Art

http://vision.middlebury.edu/stereo

Simple (Calibrated) Stereo

$$u_l = f_x \frac{x}{z} + o_x$$

$$v_l = f_y \frac{y}{z} + o_y$$



$$u_r = f_x \frac{x - b}{z} + o_x$$

$$v_r = f_y \frac{y}{z} + o_y$$

f_x, f_y, b, o_x, o_y are in pixel units.

REVIEW

Depth and Disparity

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 the Disparity.

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

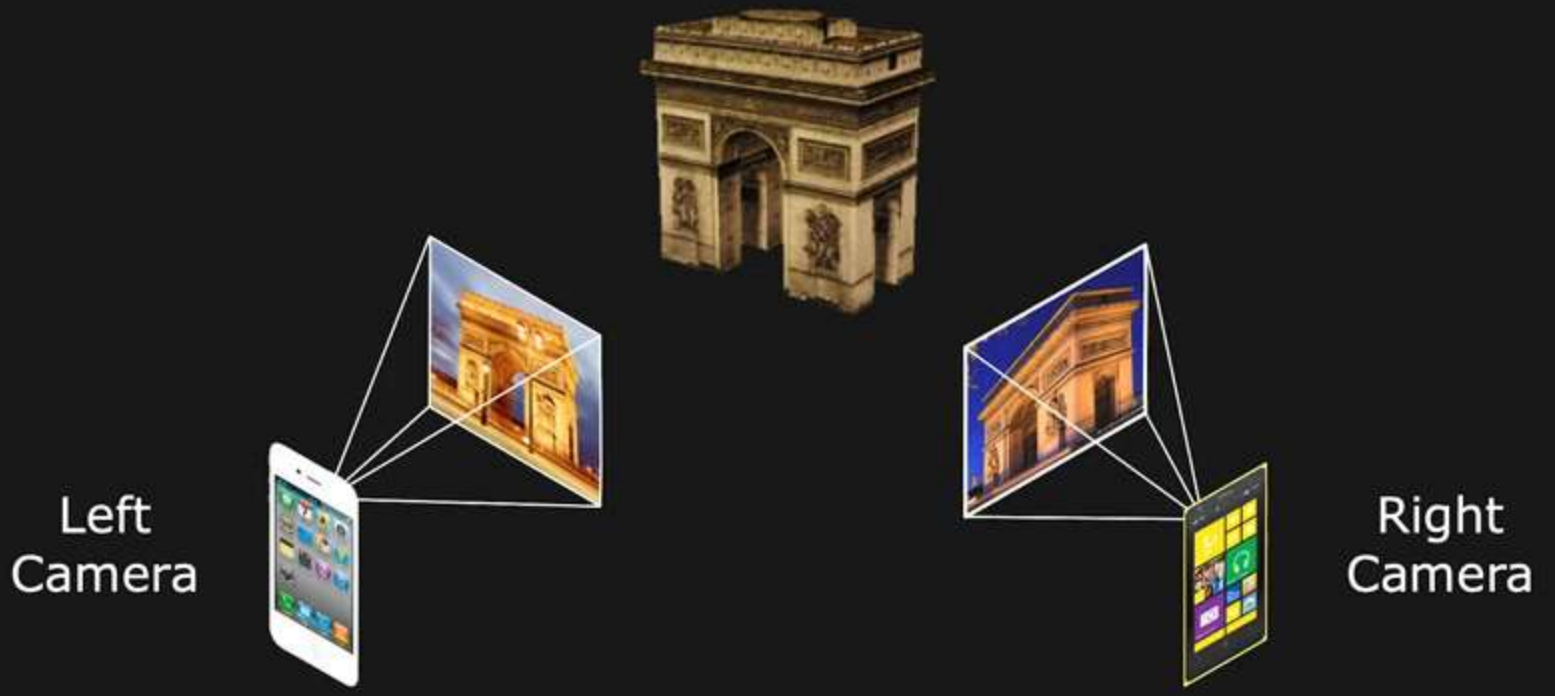
(4) Finding Dense Correspondences

(5) Computing Depth

(6) Stereopsis: Stereo in Nature

Uncalibrated Stereo

Compute 3D structure of static scene from two arbitrary views

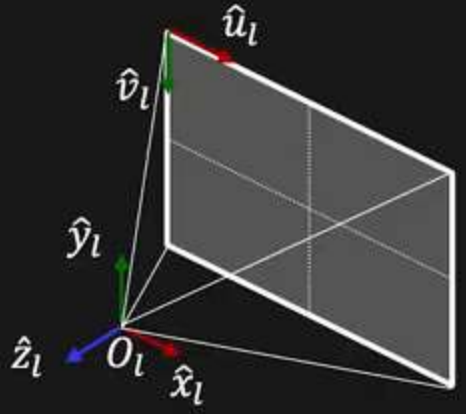


Instrinsics (f_x, f_y, o_x, o_y) are known for both views/cameras.

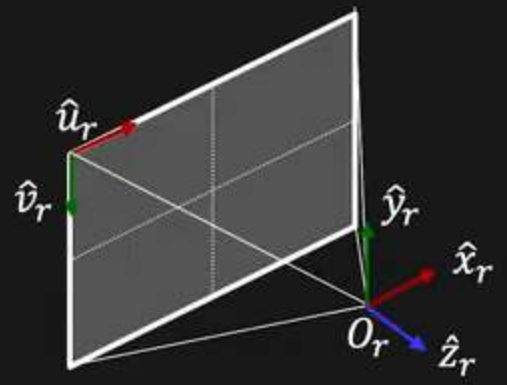
Extrinsics (relative position/orientation of cameras) are unkno

Uncalibrated Stereo

Scene



Left Camera



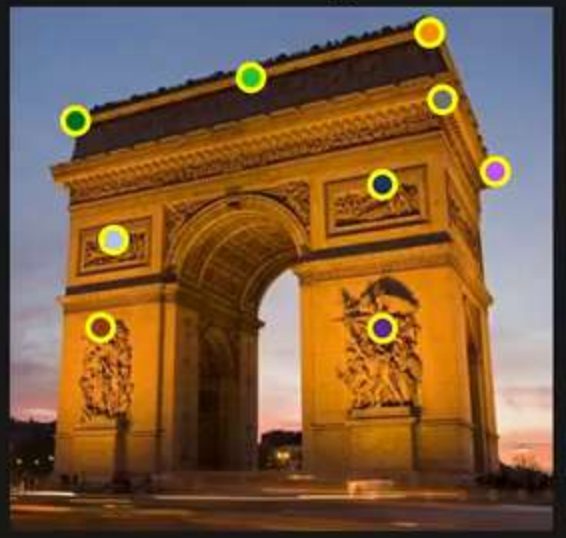
Right Camera

1. Assume Camera Matrix K is known for each camera
2. Find a few Reliable Corresponding Points

Initial Correspondence

Find a set of corresponding features (at least 8) in left and right images (e.g. using SIFT or hand-picked).

Left image



Right image

- $(u_l^{(1)}, v_l^{(1)})$

$\vdots$

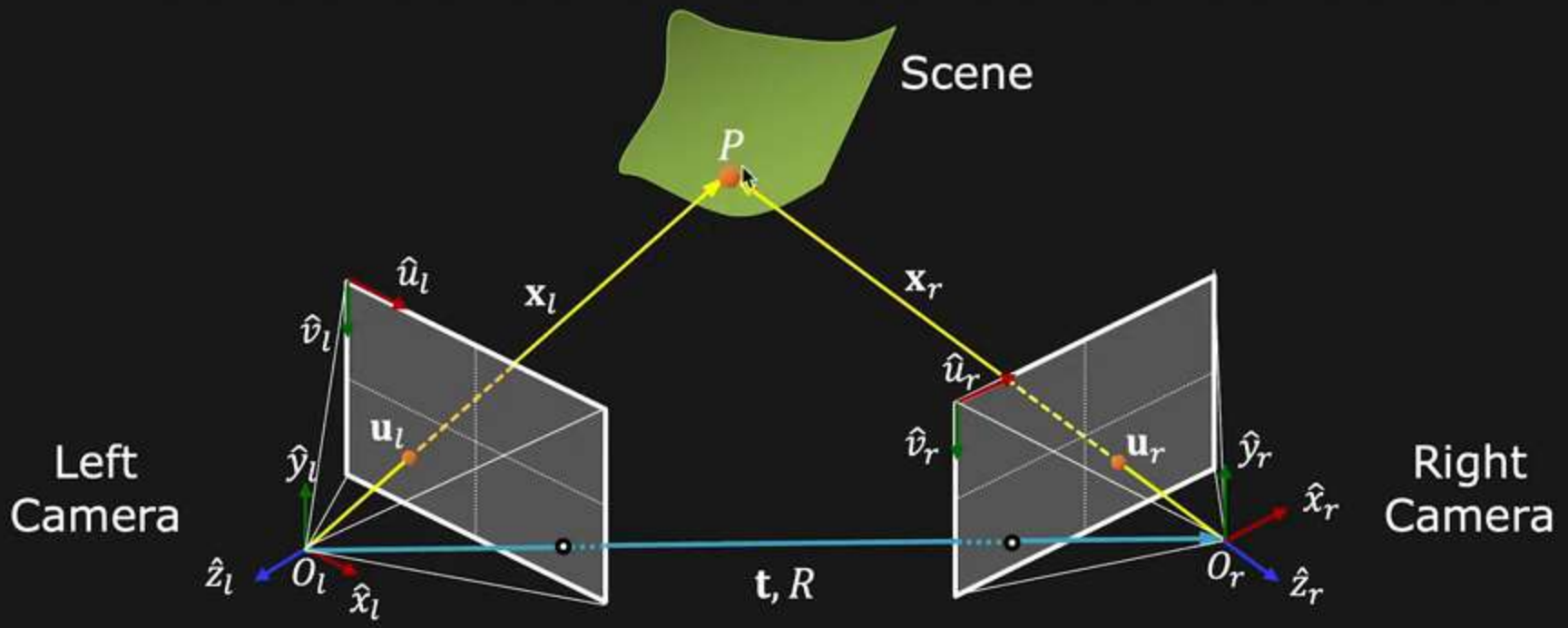
- $(u_l^{(m)}, v_l^{(m)})$

- $(u_r^{(1)}, v_r^{(1)})$

$\vdots$

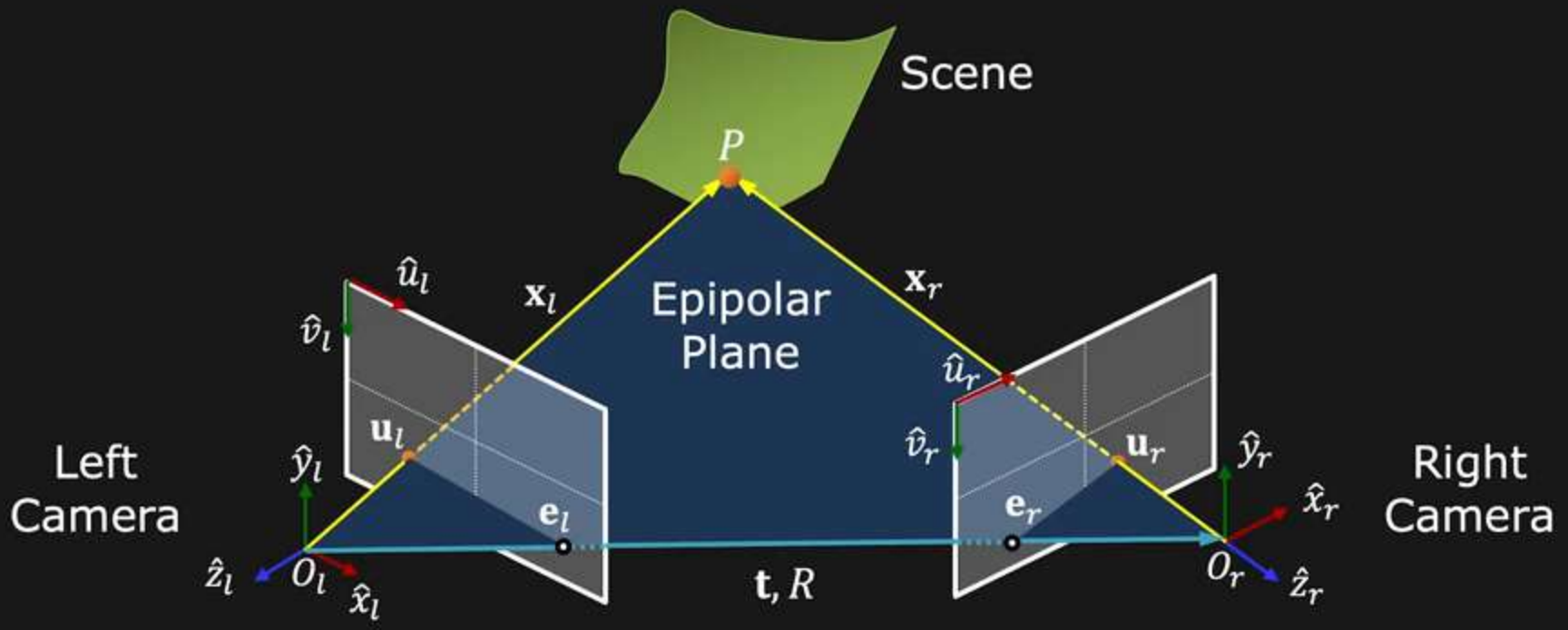
- $(u_r^{(m)}, v_r^{(m)})$

Uncalibrated Stereo



- 1. Assume Camera Matrix K is known for each camera
- 2. Find a few Reliable Corresponding Points
- 3. Find Relative Camera Position $\mathbf{t}$ and Orientation R
- 4. Find Dense Correspondence
- 5. Compute Depth using Triangulation

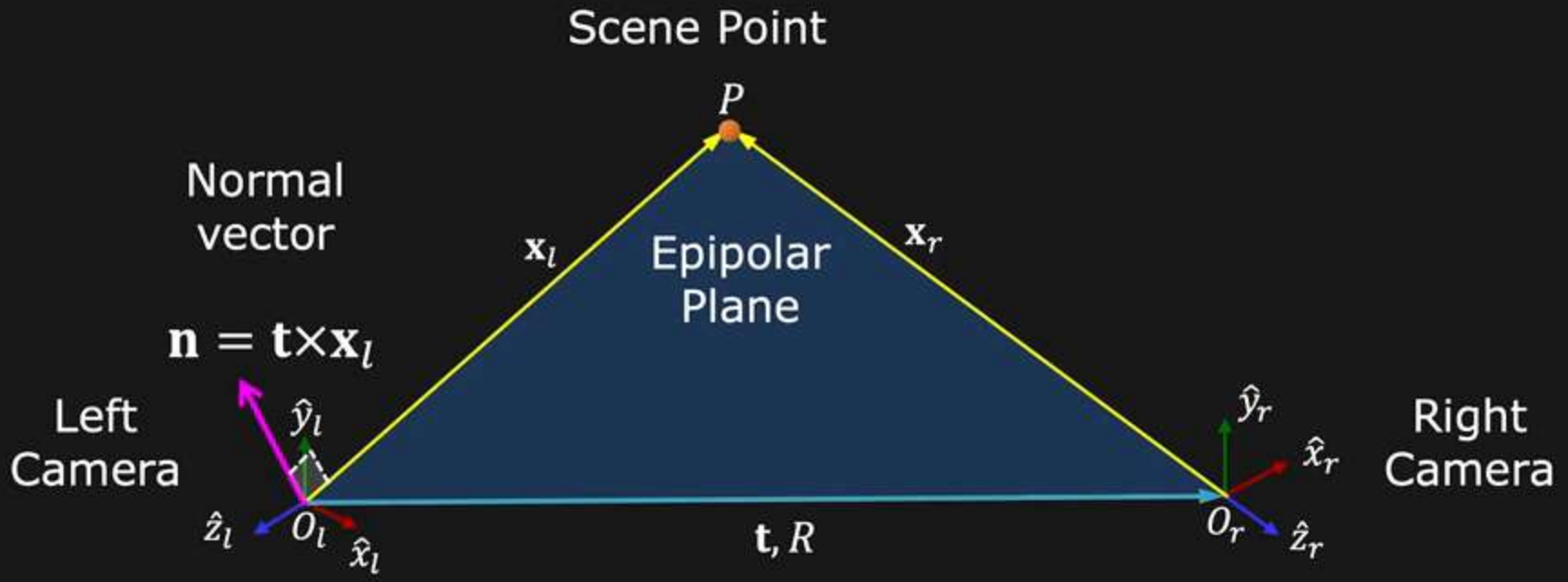
Epipolar Geometry: Epipolar Plane



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:

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

Epipolar Constraint

Writing the epipolar constraint in matrix form:

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

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

Cross-product definition

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

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

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

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

Epipolar Constraint

Substituting into the epipolar constraint gives:

$$[x_l \quad y_l \quad z_l]\left(\begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}\begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} + \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}\begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}\right) = 0$$

$$\boldsymbol{t \times t = 0}$$

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

Essential Matrix E

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

Essential Matrix E: Decomposition

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

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

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 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

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

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

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 the Image Coordinates

Left camera

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

Right camera

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

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

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

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:

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

$$z_l \neq 0$$
$$z_r \neq 0$$

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:

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

Fundamental Matrix F

Epipolar constraint:

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

Rewriting in terms of image coordinates:

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

Fundamental Matrix F

$$E = K_l^T F K_r$$

Stereopsis

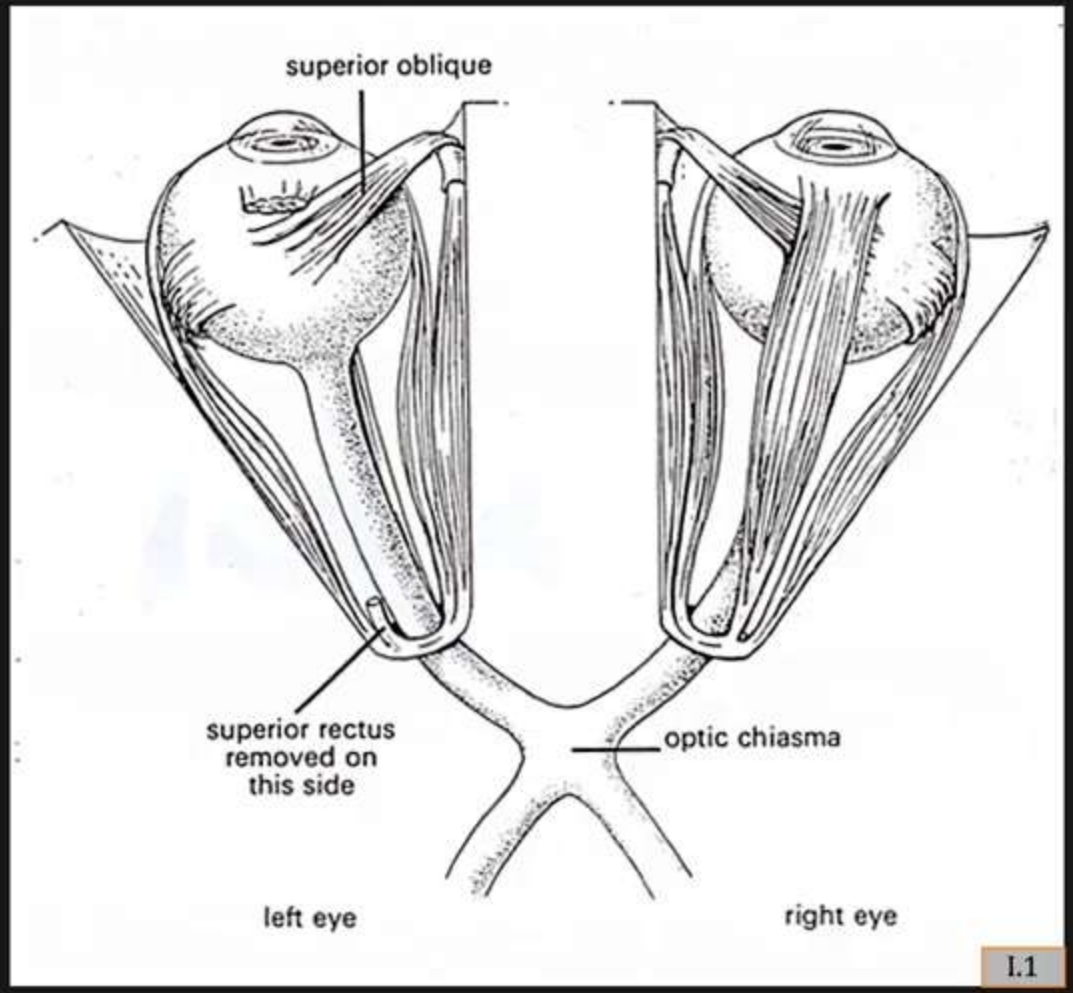


Predator eyes are configured for depth estimation

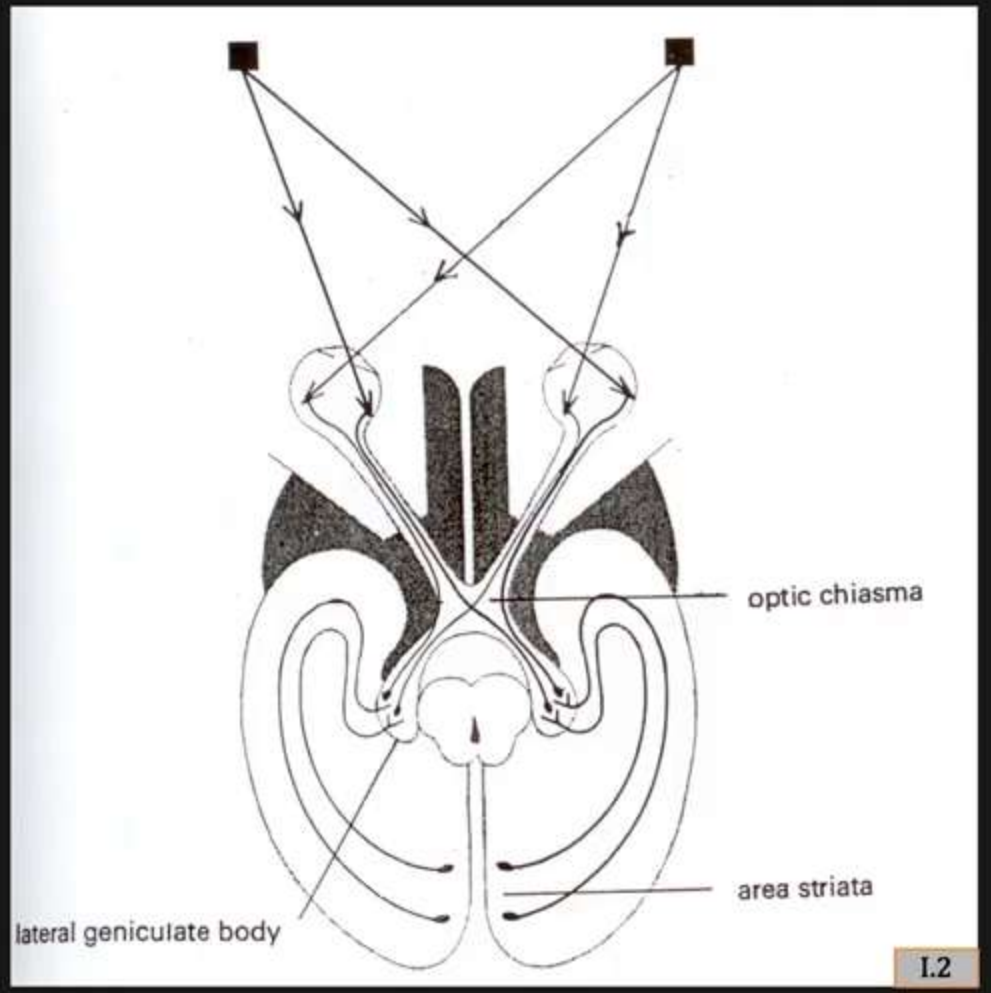


Prey eyes are configured for larger field of view

The Human Stereo System

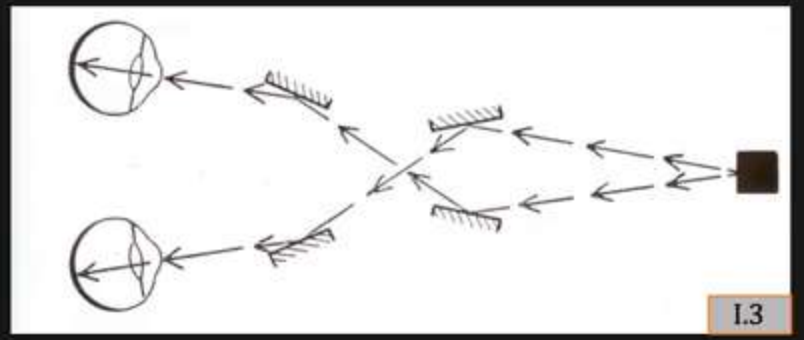


Stereopsis in Humans

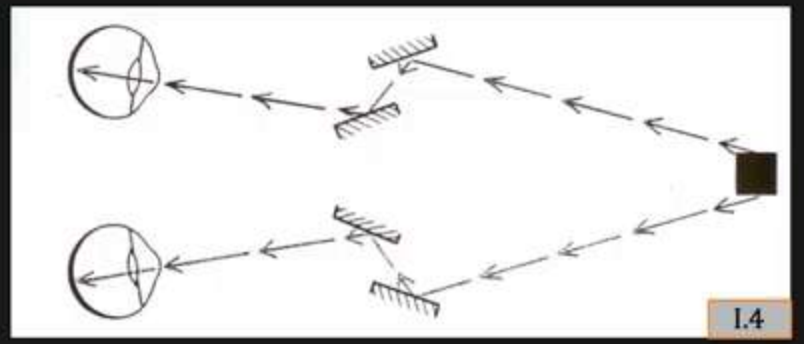


Human Stereo Experiments

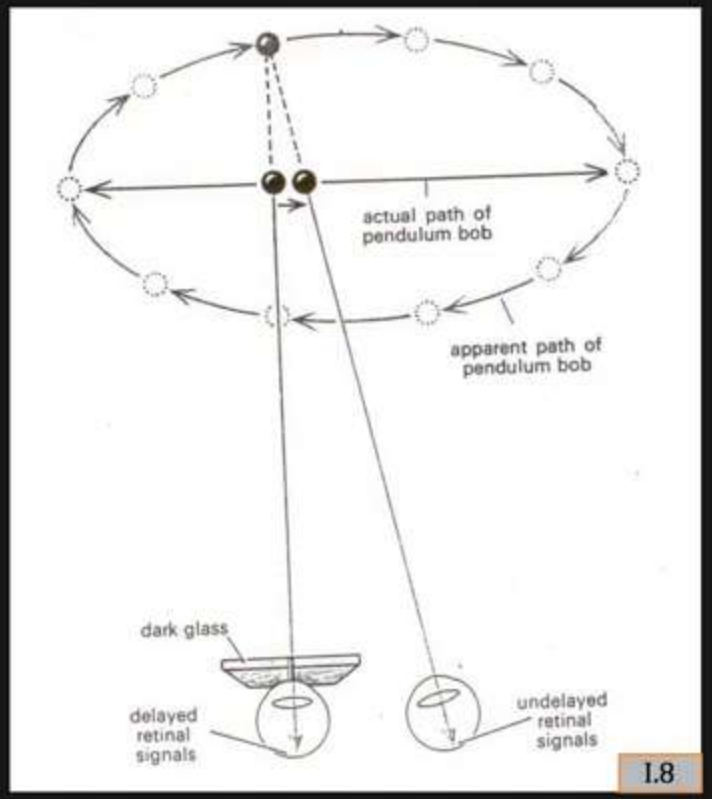
A *psuedoscope* gives reversed depth



A *telestereoscope* increases separation of the eyes

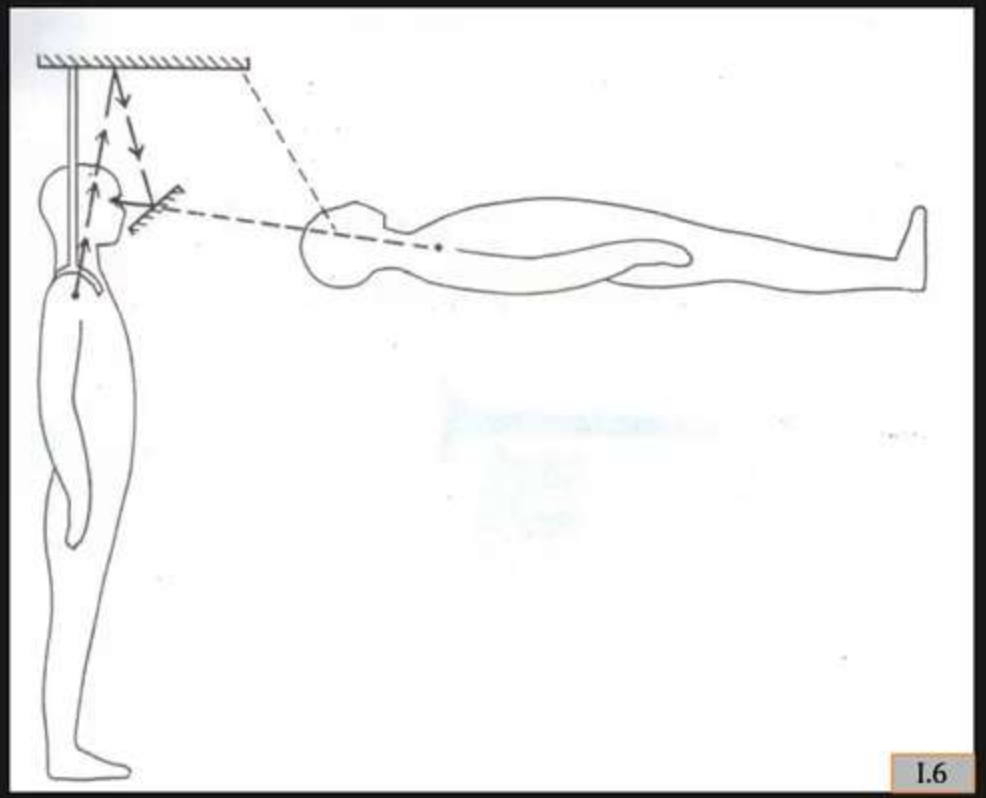


The Pulfrich Pendulum Effect



A pendulum swinging in a straight arc is viewed throug
dark glass in one eye. The motion appears to be ellipti

Stratton's Experiment

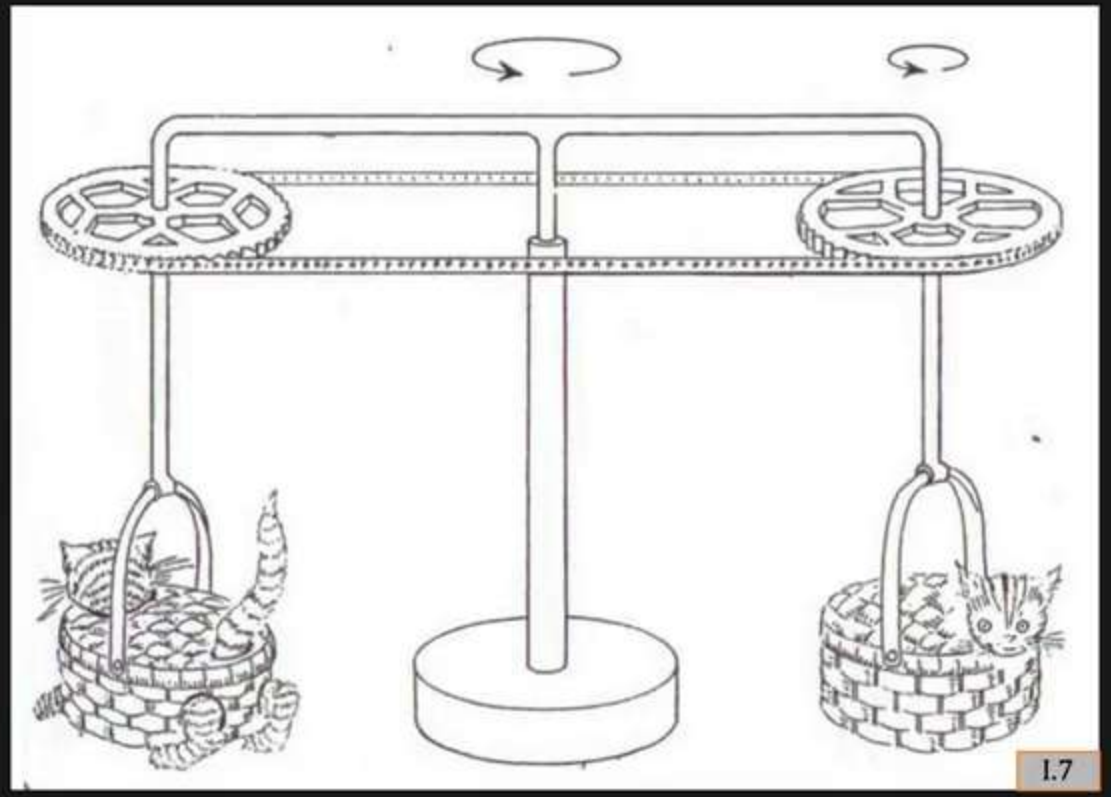


When wearing this device, Stratton saw himself suspended in space before his eyes.

Pfister's Hen

Prisms are placed in front of each eye to rotate the field of view, effecting the efficiency of depth perception

Percpetual Learning and Vision



Apparatus designed by Held and Hein to discover whether learning takes place in a passive animal.

Optical Flow

Method to estimate apparent motion of scene points from a sequence of images.

Topics:

(1) Motion Field and Optical Flow

(2) Optical Flow Constraint Equation

(3) Lucas-Kanade Method

(4) Coarse-to-Fine Flow Estimation

(5) Applications of Optical Flow

Motion Field

Image velocity of a point that is moving in the scene

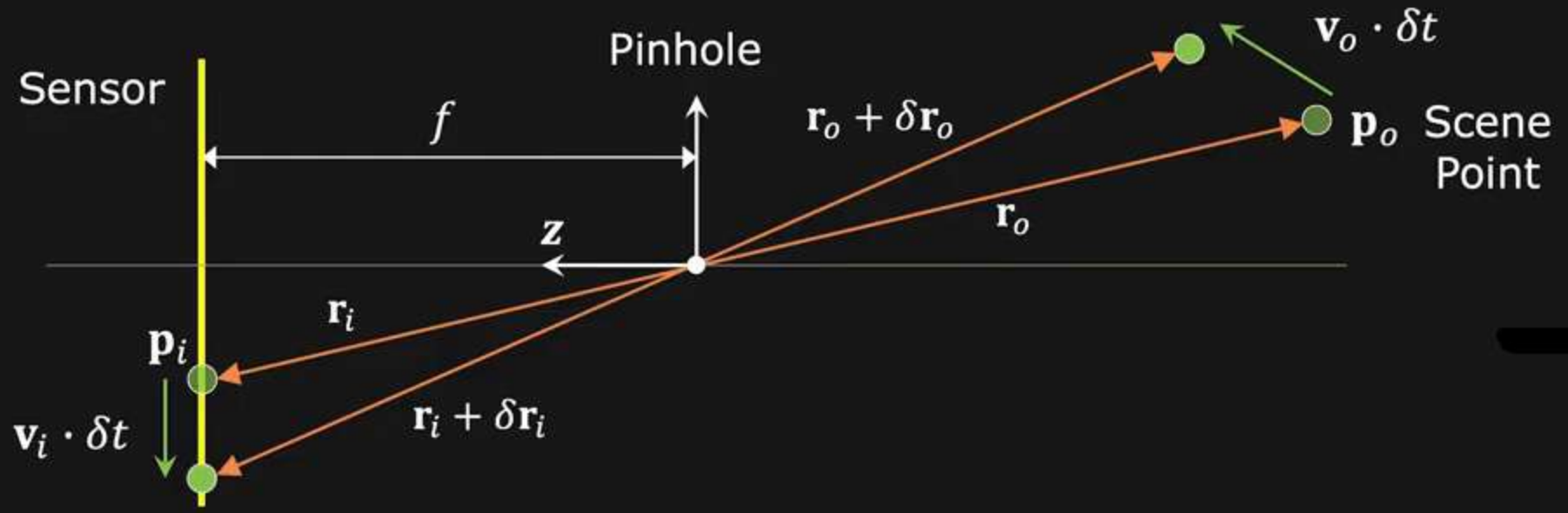


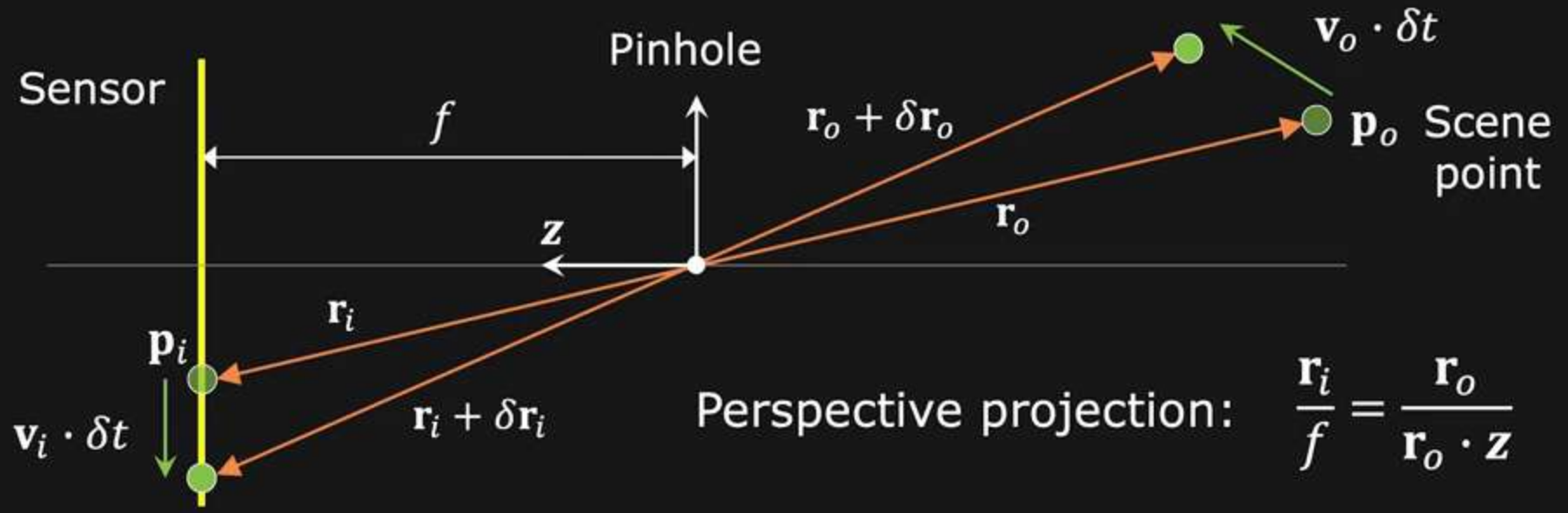
Image Point Velocity: $\mathbf{v}_i = \dfrac{d\mathbf{r}_i}{dt}$ (Motion Field)

Scene Point Velocity: $\mathbf{v}_o = \dfrac{d\mathbf{r}_o}{dt}$

[Horn 1981]

Motion Field

Image velocity of a point that is moving in the scene



Perspective projection: $\frac{\mathbf{r}_i}{f} = \frac{\mathbf{r}_o}{\mathbf{r}_o \cdot \mathbf{z}}$

Image Point Velocity: $\mathbf{v}_i = \frac{d\mathbf{r}_i}{dt} = f\frac{(\mathbf{r}_o \cdot \mathbf{z})\mathbf{v}_0 - (\mathbf{v}_o \cdot \mathbf{z})\mathbf{r}_0}{(\mathbf{r}_o \cdot \mathbf{z})^2}$

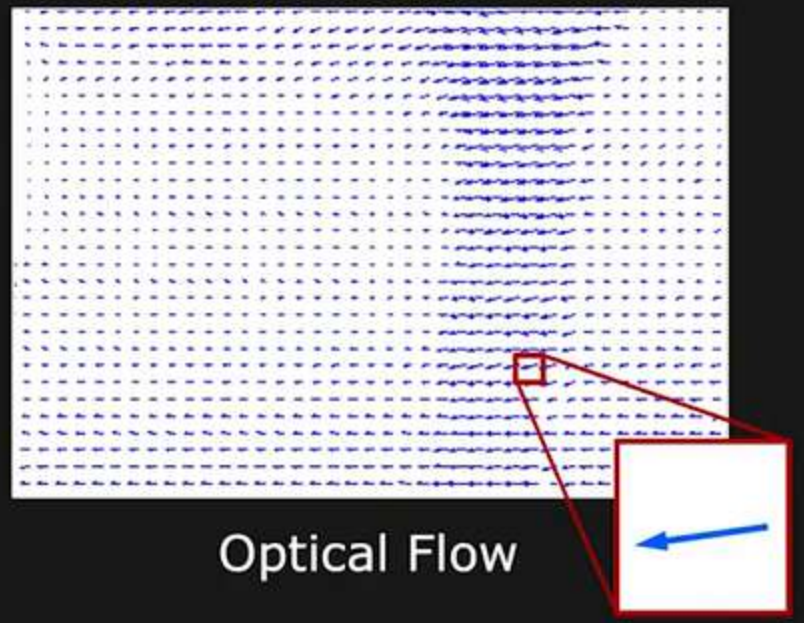
(Motion Field)

$$\mathbf{v}_i = f\frac{(\mathbf{r}_o \times \mathbf{v}_0) \times \mathbf{z}}{(\mathbf{r}_o \cdot \mathbf{z})^2}$$

Optical Flow

Motion of brightness patterns in the image

Image Sequence
(2 frames)

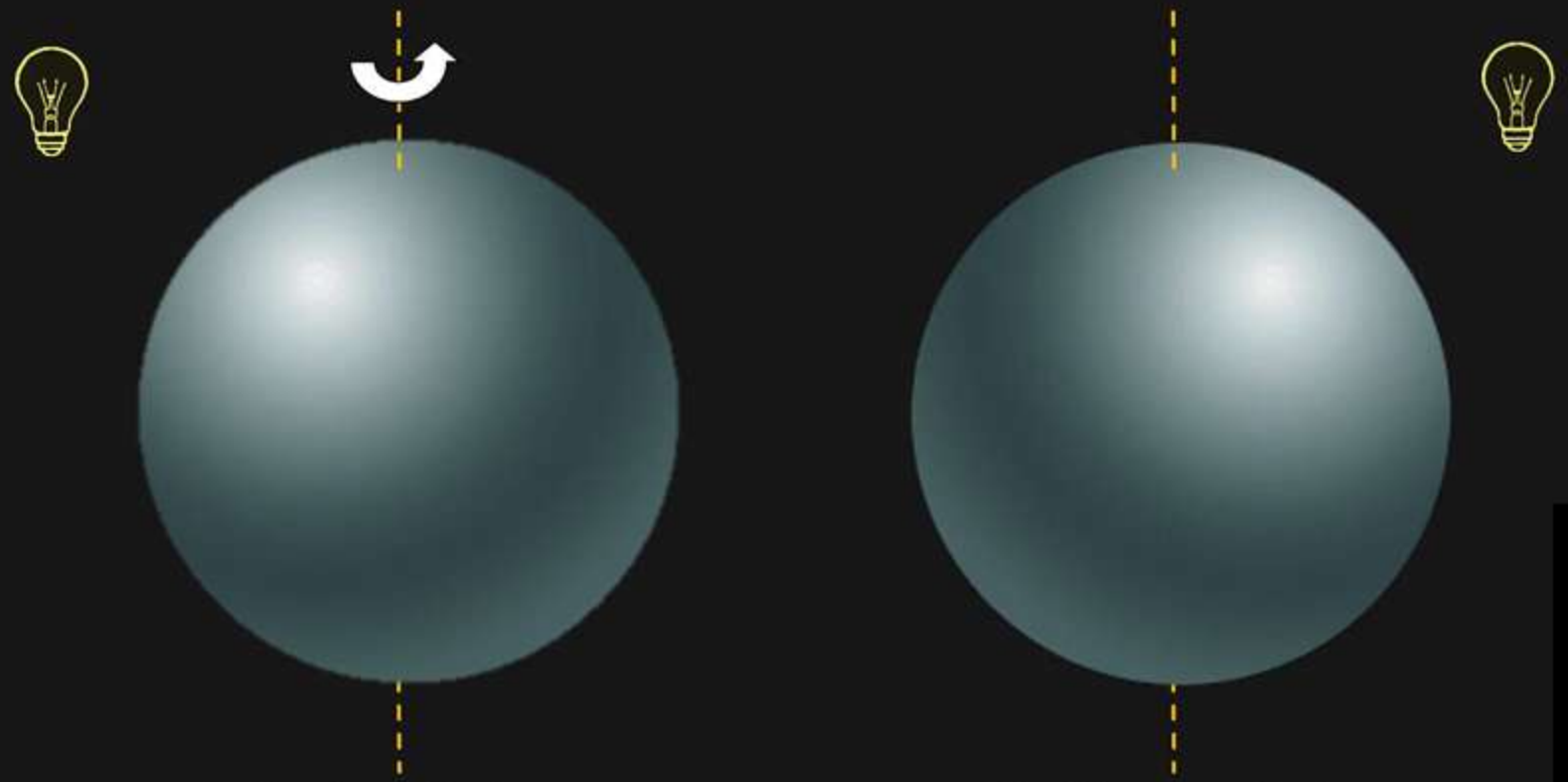


Optical Flow

Velocity of
brightness patt

Ideally, Optical Flow = Motion Field

When is Optical Flow ≠ Motion Field?



Spinning Sphere
Stationary Light Source

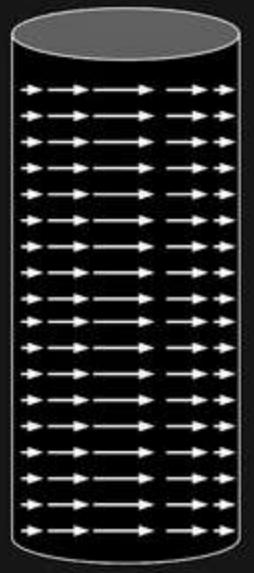
Motion Field exists
But no Optical Flow

Stationary Sphere
Moving Light Source

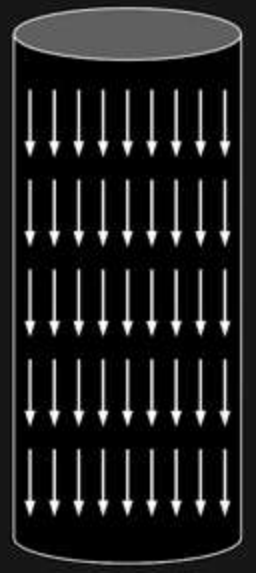
No Motion Field exists
But there is Optical Flow

When is Optical Flow ≠ Motion Field?

Barber Pole Illusion

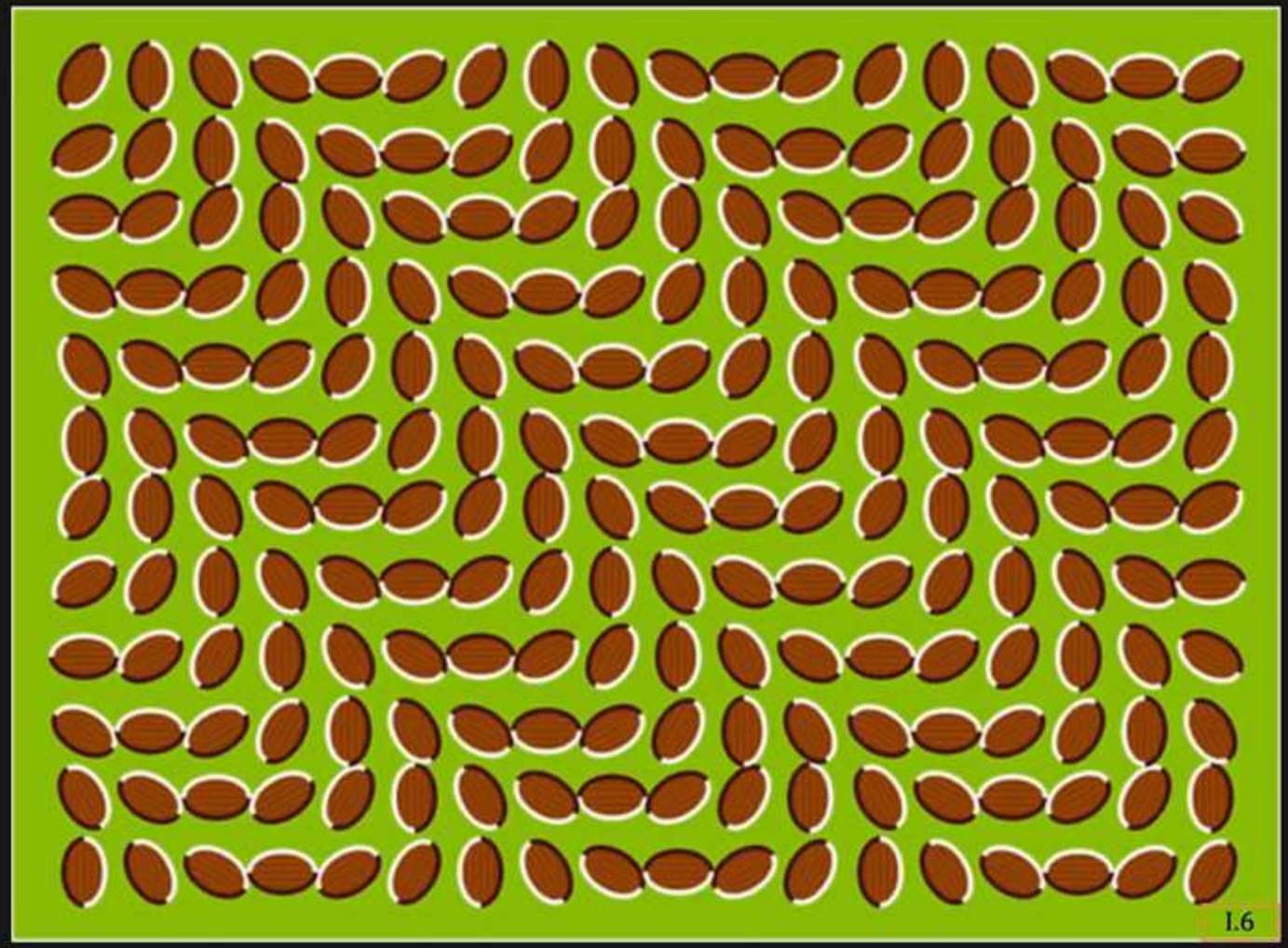


Motion Field



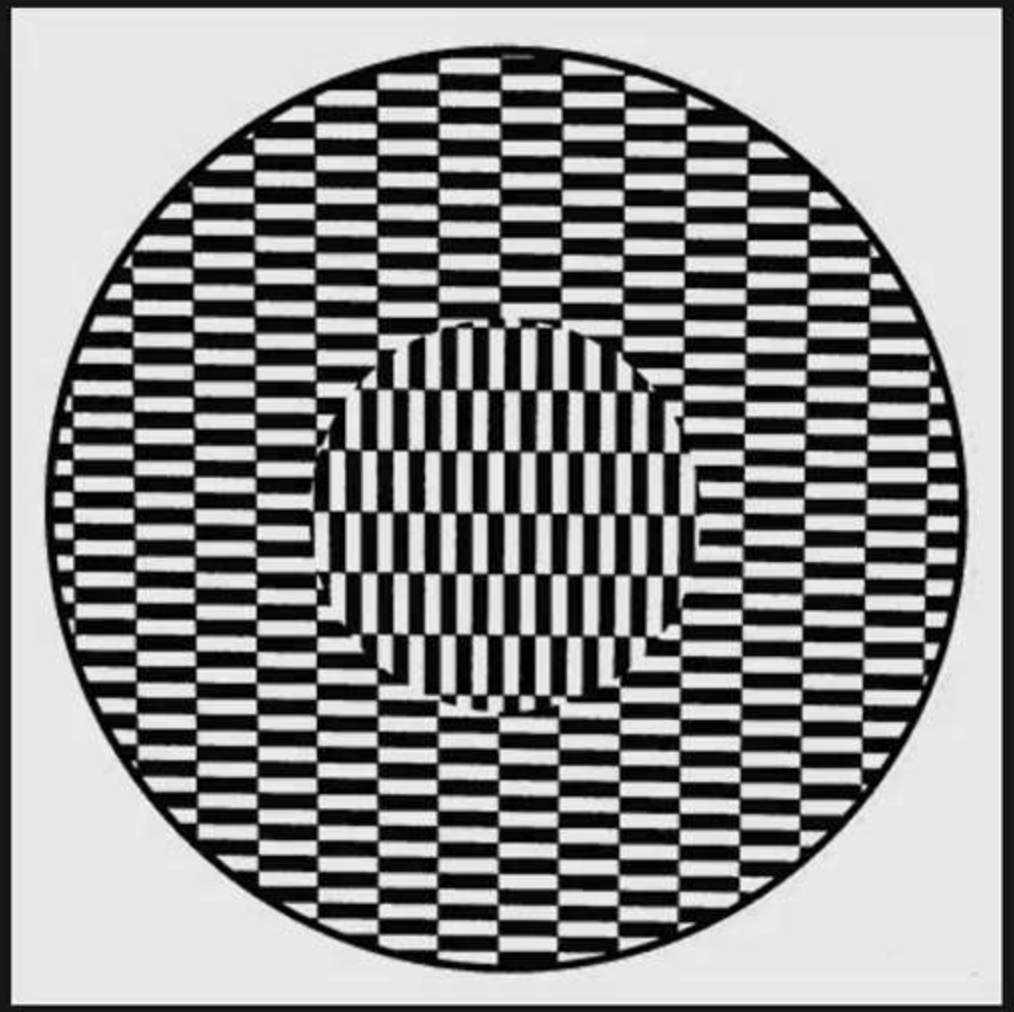
Optical Flow

Motion Illusions



Donguri Wave Illusion

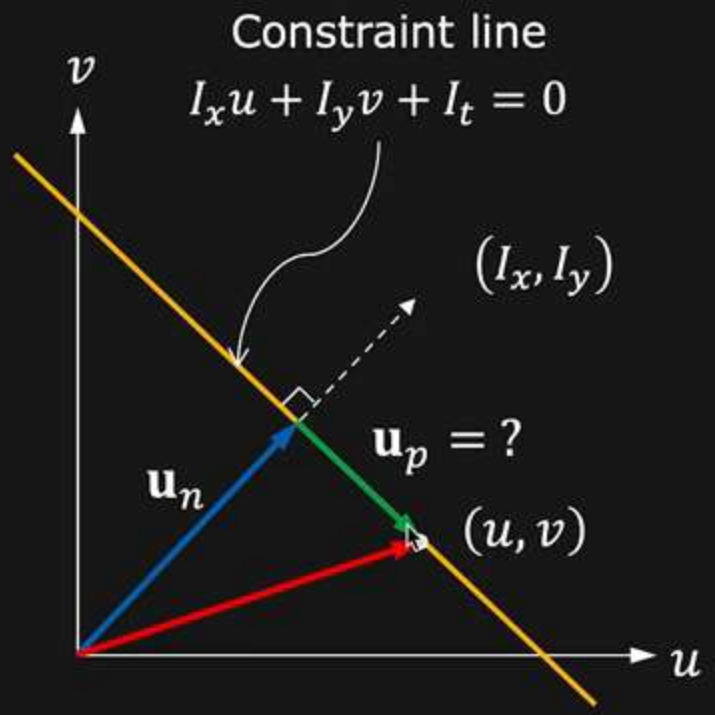
Motion Illusions



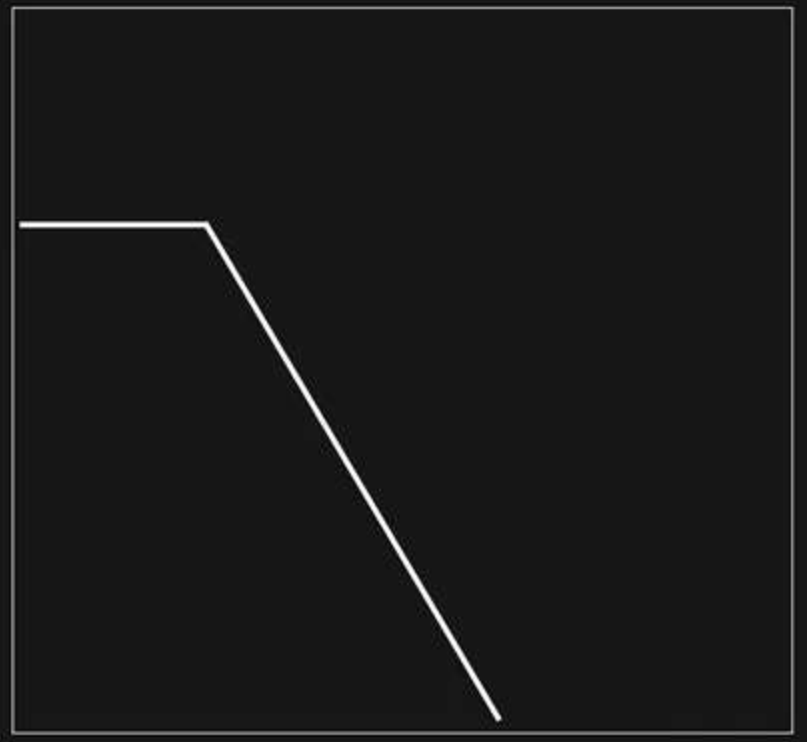
Ouchi Pattern

Parallel Flow

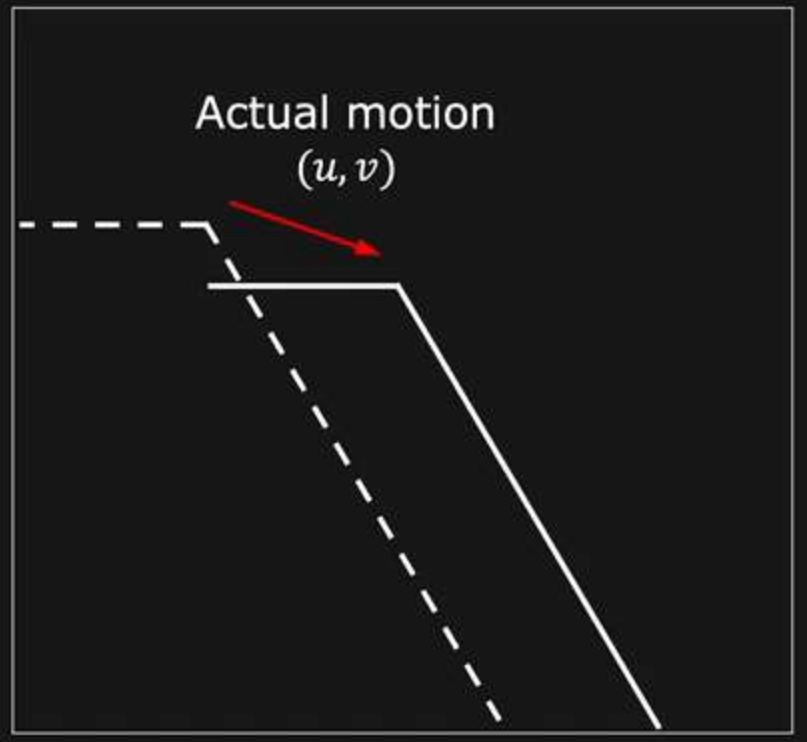
We cannot determine $\mathbf{u}_p$,
the optical flow component
parallel to the constraint line.



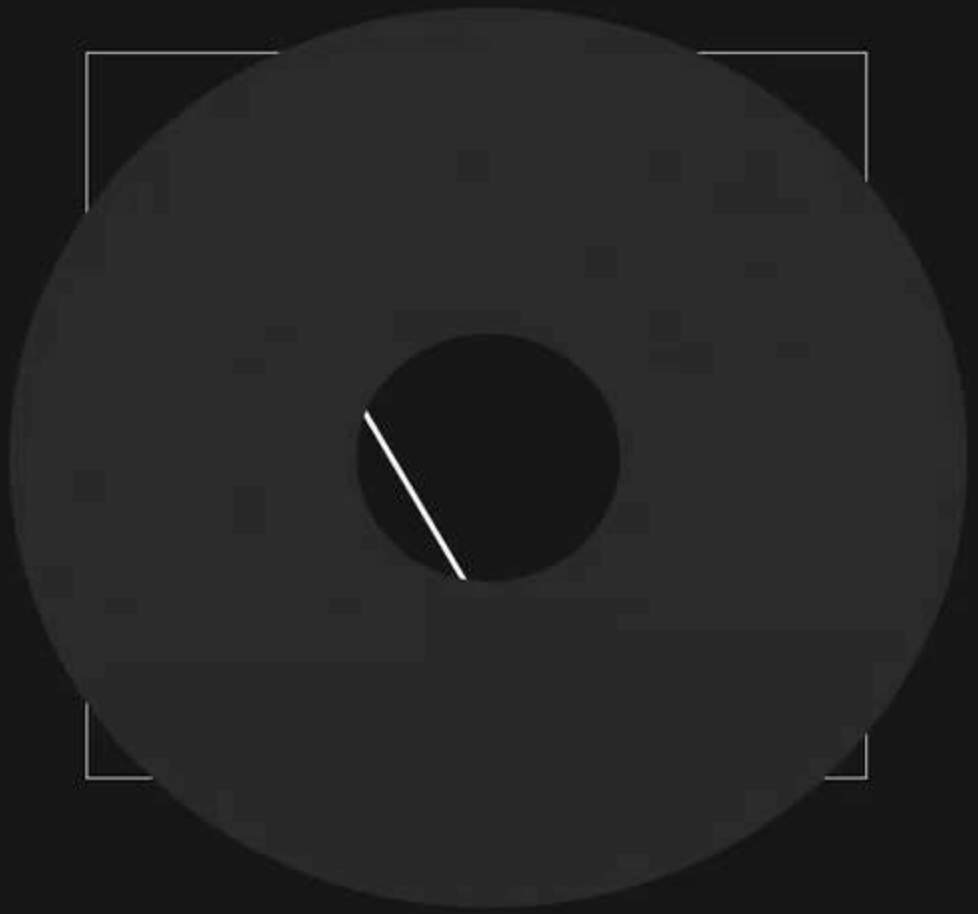
Aperture Problem



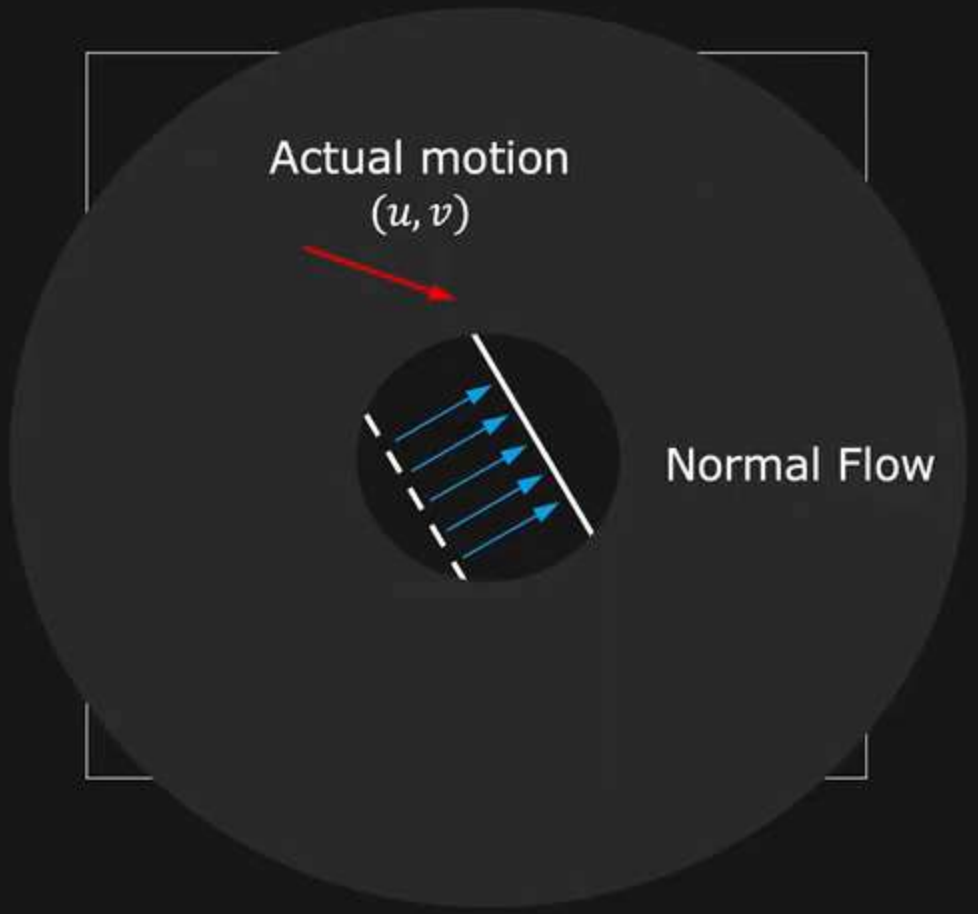
Aperture Problem



Aperture Problem



Aperture Problem



Locally, we can only determine normal flow!

Optical Flow is Under Constrained

Constraint Equation: $I_x u + I_y v + I_t = 0$

2 unknowns, 1 equation.

Lucas-Kanade Solution

Assumption: For each pixel, assume Motion Field, and hence Optical Flow (u, v), is constant within a small neighborhood W.

That is for all points $(k, l) \in W$:

$$I_x(k,l)u + I_y(k,l)v + I_t(k,l) = 0$$

Lucas-Kanade Solution

For all points $(k,l) \in W$: $I_x(k,l)u + I_y(k,l)v + I_t(k,l) = 0$

Let the size of window W be $n \times n$

In matrix form:

$$\begin{bmatrix} I_x(1,1) & I_y(1,1) \\ I_x(k,l) & I_y(k,l) \\ \vdots & \vdots \\ I_x(n,n) & I_y(n,n) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} I_t(1,1) \\ I_t(k,l) \\ \vdots \\ I_t(n,n) \end{bmatrix}$$

A (Known) $n^2 \times 2$ | $\mathbf{u}$ (Unknown) 2×1 | B (Known) $n^2 \times 1$

n^2 Equations, 2 Unknowns: Find Least Squares Solution

Least Squares Solution

Solve linear system: $A\mathbf{u} = B$

$$A^T A\mathbf{u} = A^T B$$ (Least-Squares using Pseudo-Inverse)

In matrix form:

$$\begin{bmatrix} \sum_w I_x I_x & \sum_w I_x I_y \\ \sum_w I_x I_y & \sum_w I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -\sum_w I_x I_t \\ -\sum_w I_y I_t \end{bmatrix}$$

Indices (k, l) not written for simplicity

$A^T A$	$\mathbf{u}$	$A^T B$
(Known)	(Unknown)	(Known)
2×2	2×1	2×1

$$\mathbf{u} = (A^T A)^{-1} A^T B$$

Fast and Easy to Solve

When Does Optical Flow Estimation Work?

$$A\mathbf{u} = B \qquad A^T A\mathbf{u} = A^T B$$

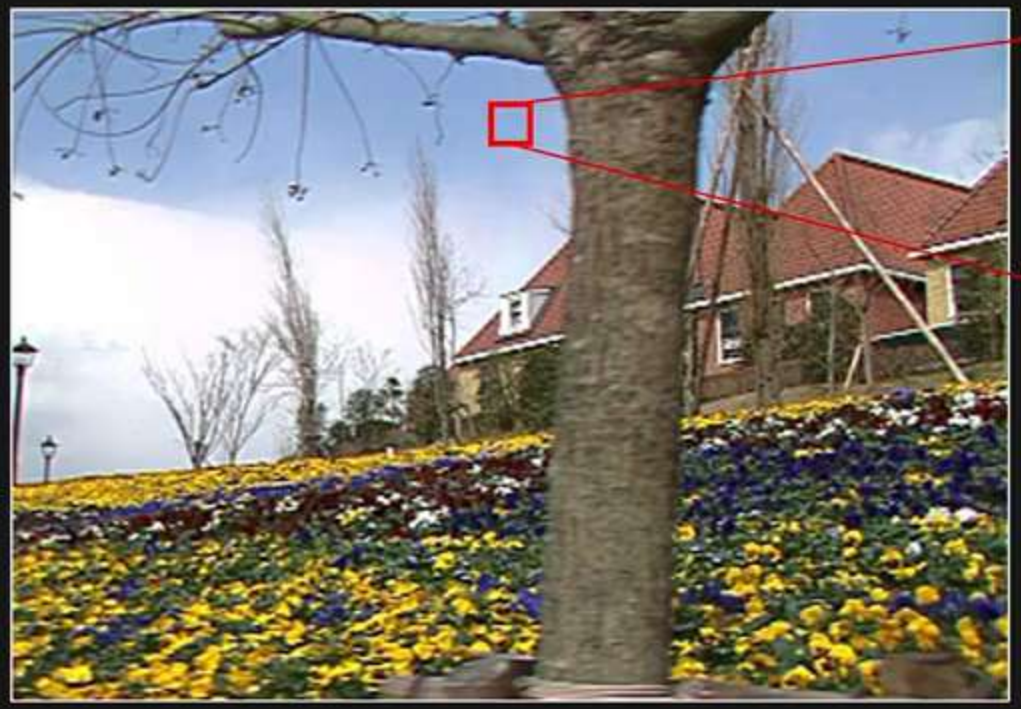
- $A^T A$ must be invertible. That is $\det(A^T A) \neq 0$
- $A^T A$ must be well-conditioned.

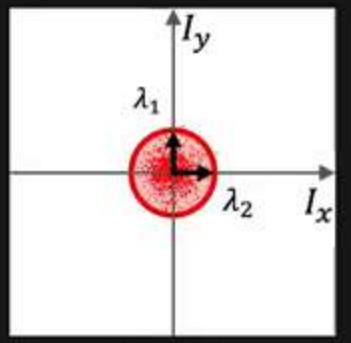
  If λ_1 and λ_2 are eigen values of $A^T A$, then

  $\lambda_1 > \epsilon$ and $\lambda_2 > \epsilon$

  $\lambda_1 \geq \lambda_2$ but not $\lambda_1 \ggg \lambda_2$

Smooth Regions (Bad)



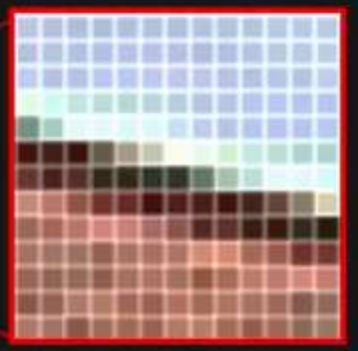


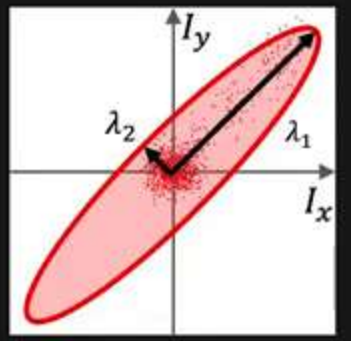
$\lambda_1 \sim \lambda_2$
Both are Small

Equations for all pixels in window are more or less the sam

Cannot reliably compute flow!

Edges (Bad)





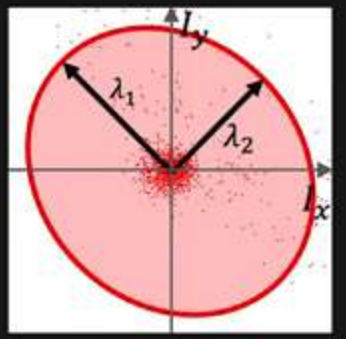
$\lambda_1 \ggg \lambda_2$

Badly conditioned. Prominent gradient in one direction.

Cannot reliably compute flow!

Same as Aperture Problem.

Textured Regions (Good)



$\lambda_1 \sim \lambda_2$
Both are Large

Well conditioned. Large and diverse gradient magnitudes.

Can reliably compute optical flow.

What if we have Large Motion?

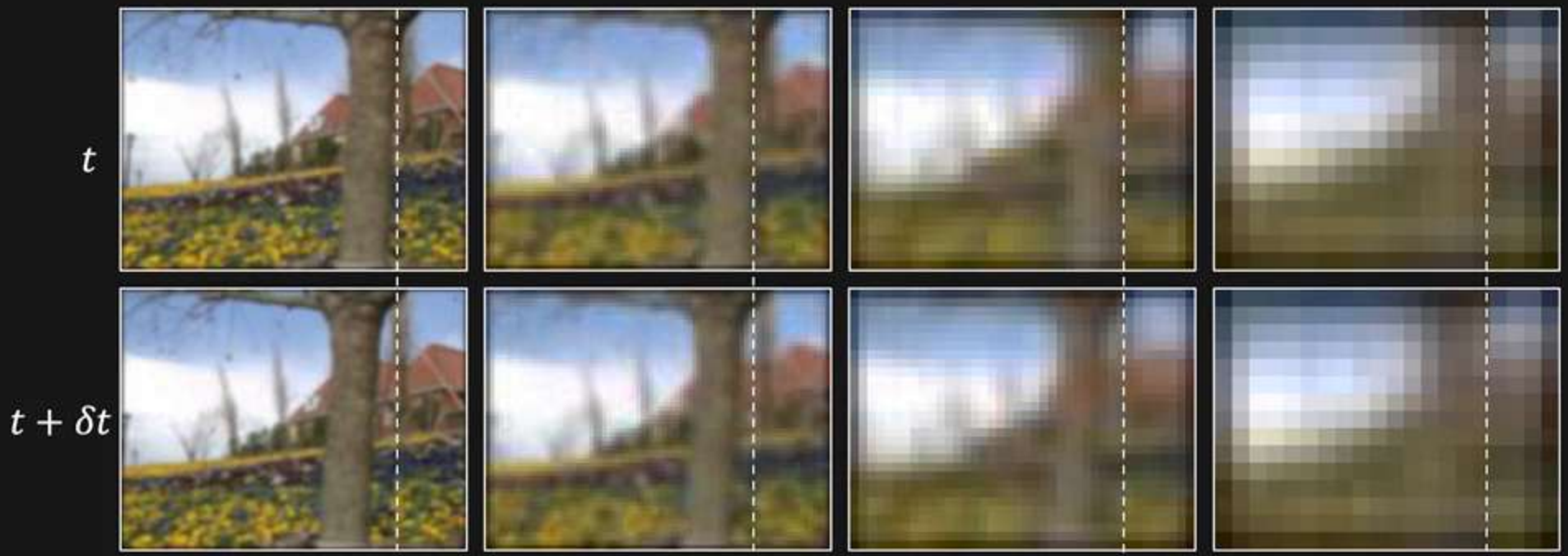
Taylor Series approximation of

$I(x + \delta x, y + \delta y, t + \delta t)$ is not valid

Our simple linear constraint equation not valid

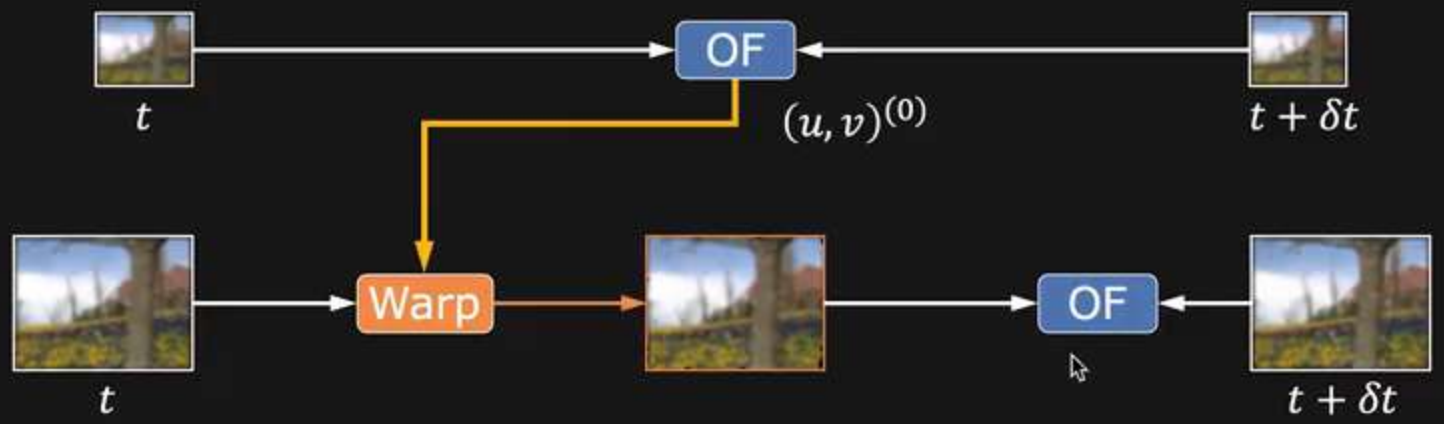
$$I_x u + I_y v + I_t \neq 0$$

Large Motion: Coarse-to-Fine Estimation

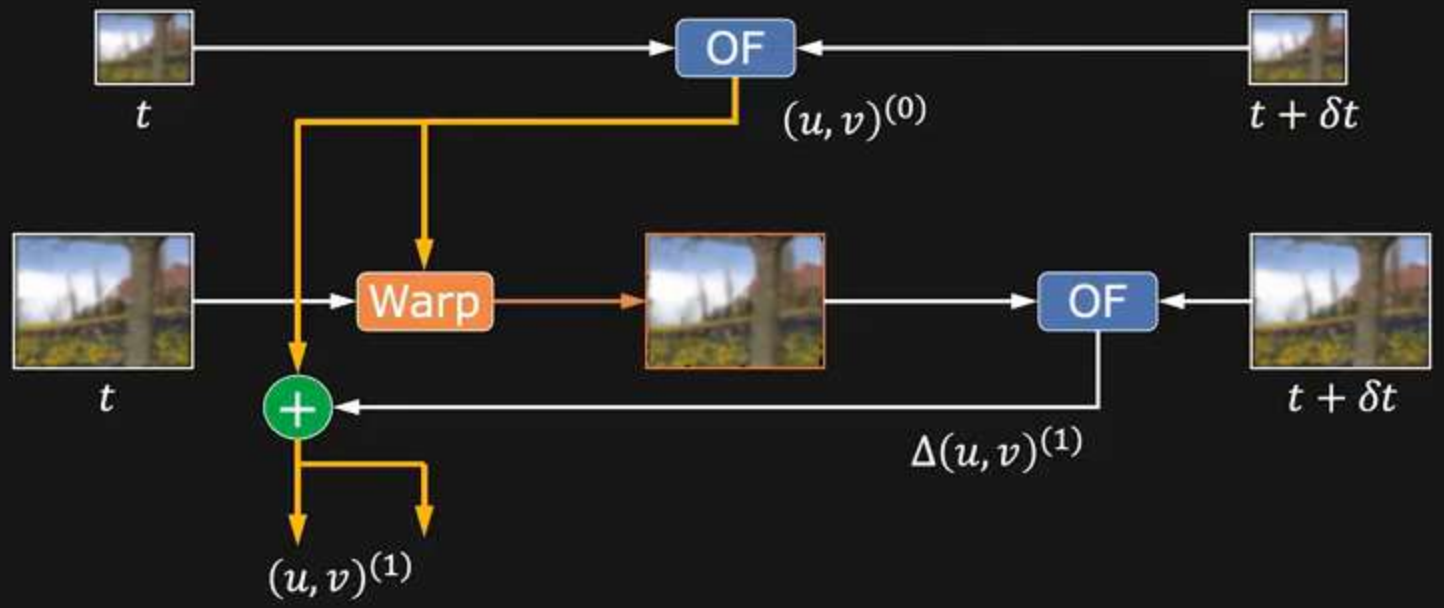


At lowest resolution, motion ≤ 1 pixel

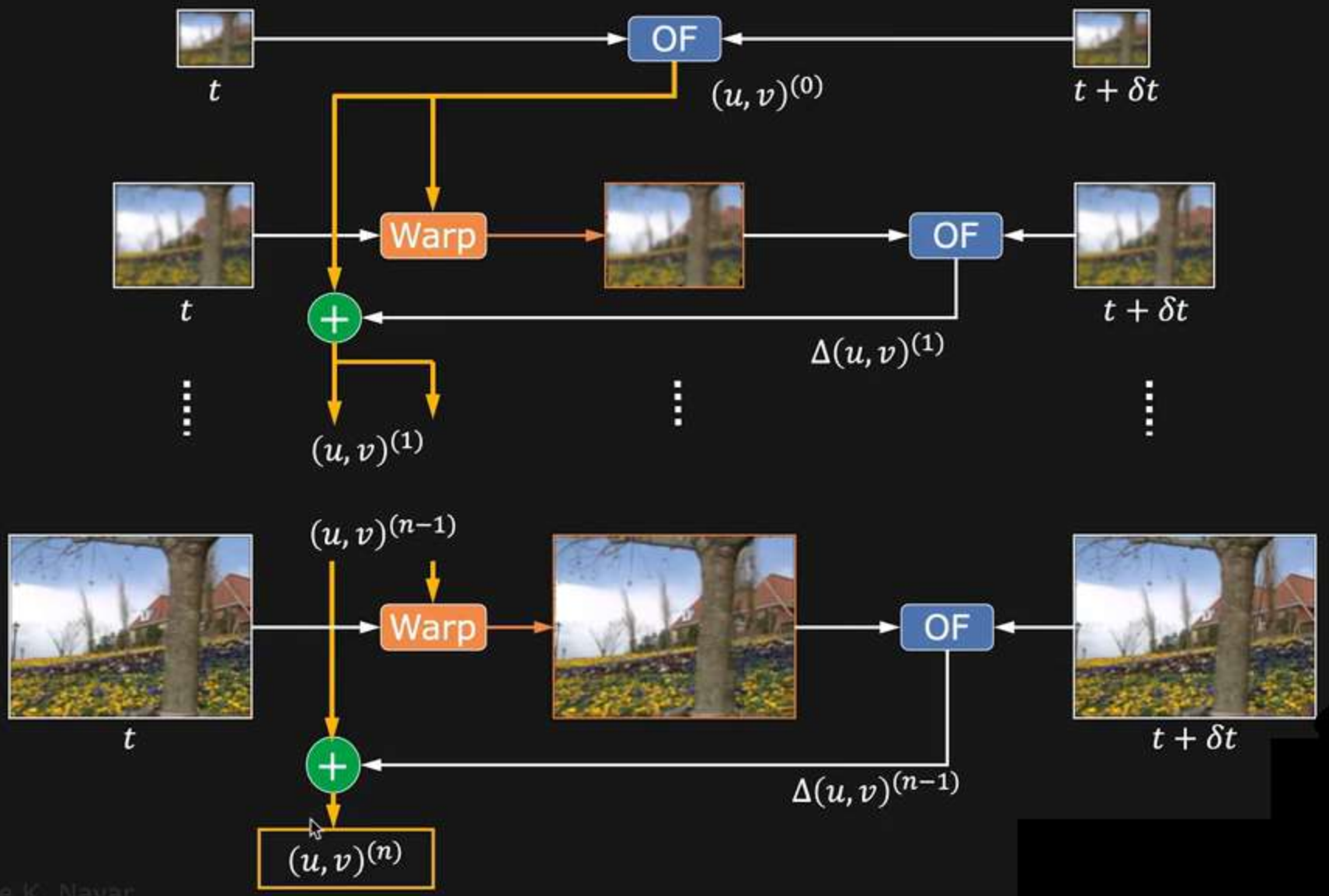
Coarse-to-Fine Estimation Algorithm



Coarse-to-Fine Estimation Algorithm

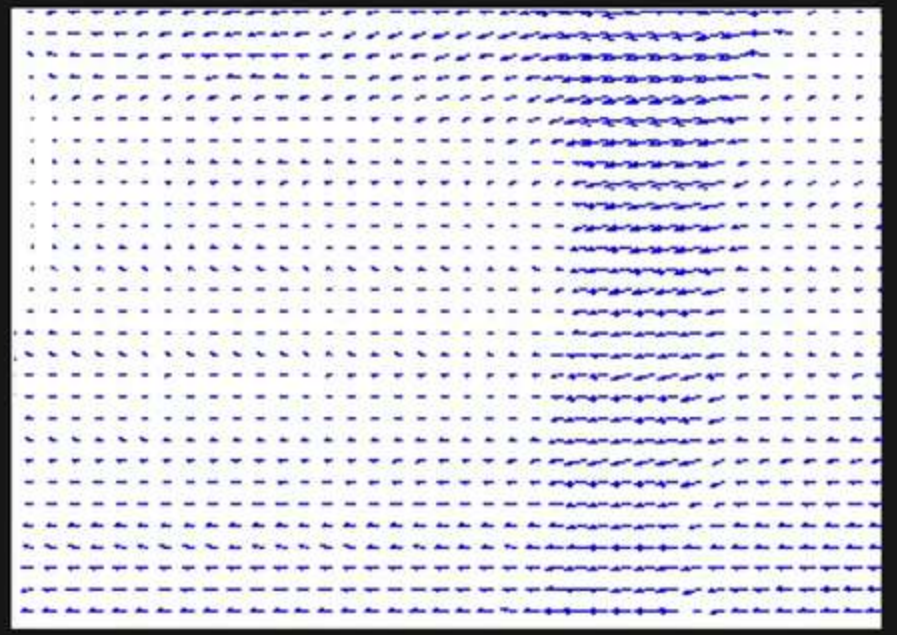


Coarse-to-Fine Estimation Algorithm



Results: Tree Sequence

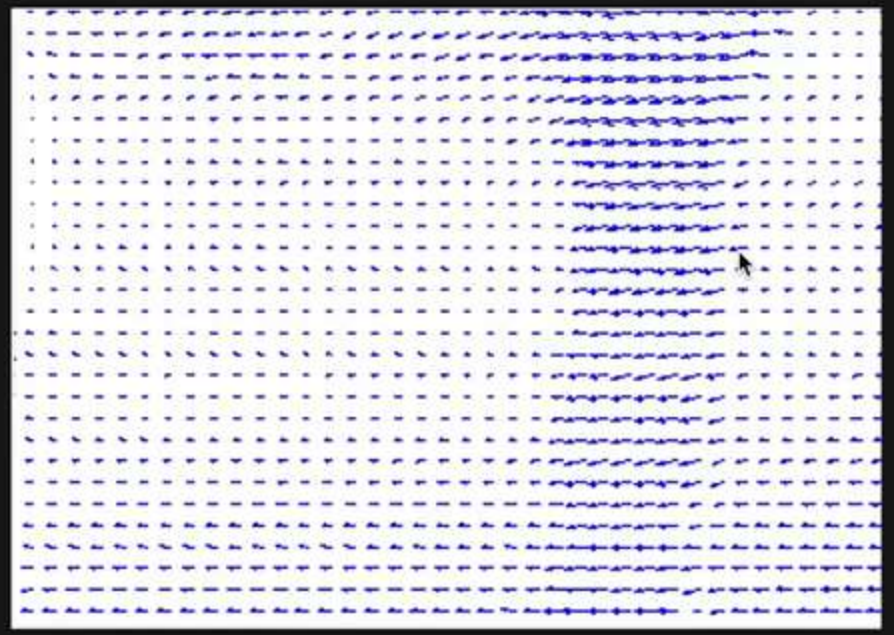
Image Sequence



Optical Flow

Results: Tree Sequence

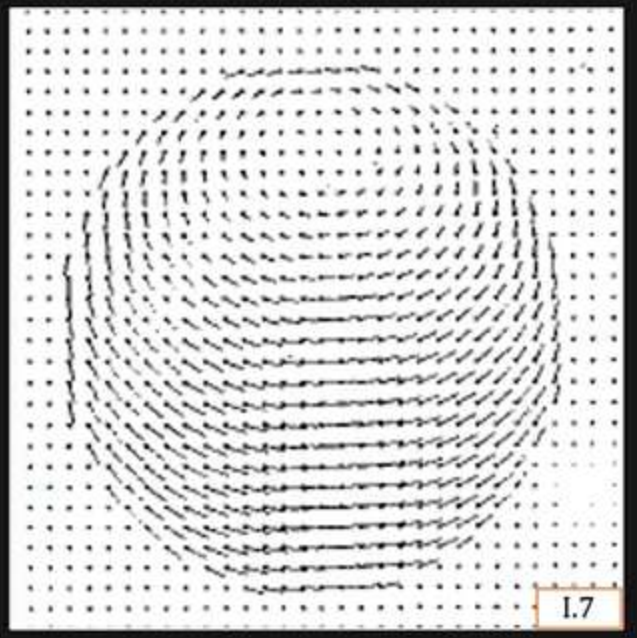
Image Sequence



Optical Flow

Results: Rotating Ball

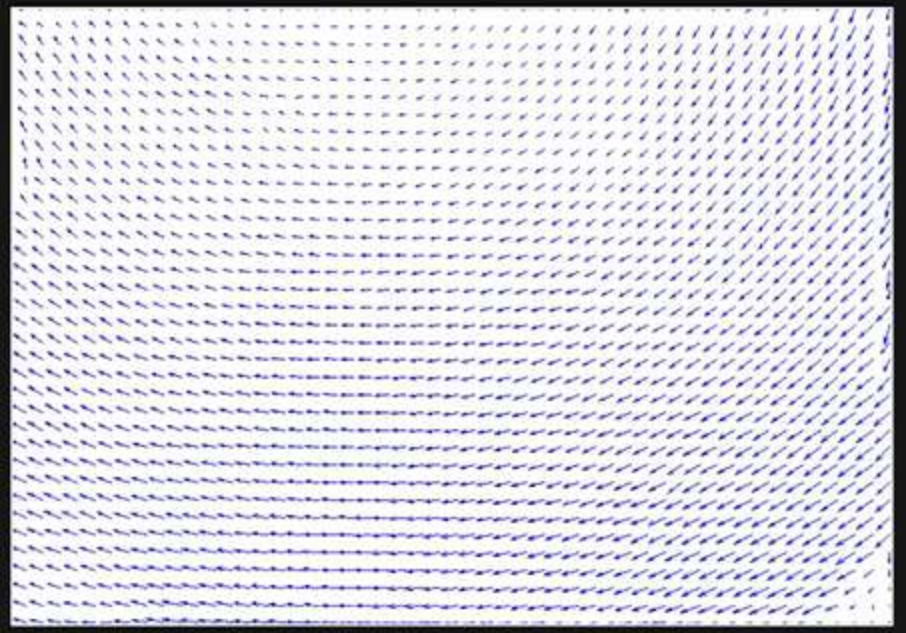
Image Sequence



Optical Flow

Results: Rotating Camera

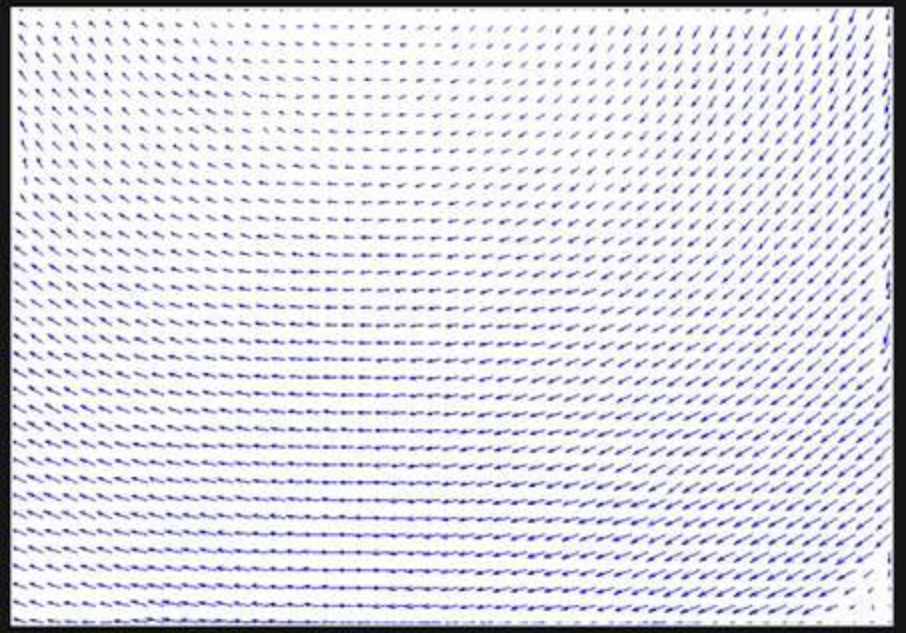
Image Sequence



Optical Flow

Results: Rotating Camera

Image Sequence



Optical Flow

Alternative Approach: Template Matching

Determine Flow using Template Matching

Template Window T

Image I_1 at time t

Search Window S

Image I_2 at time $t + \delta t$

For each template window T in image I_1,
find the corresponding match in image I_2.

Large Motion: Template matching

Determine Flow using Template Matching

Template Window T

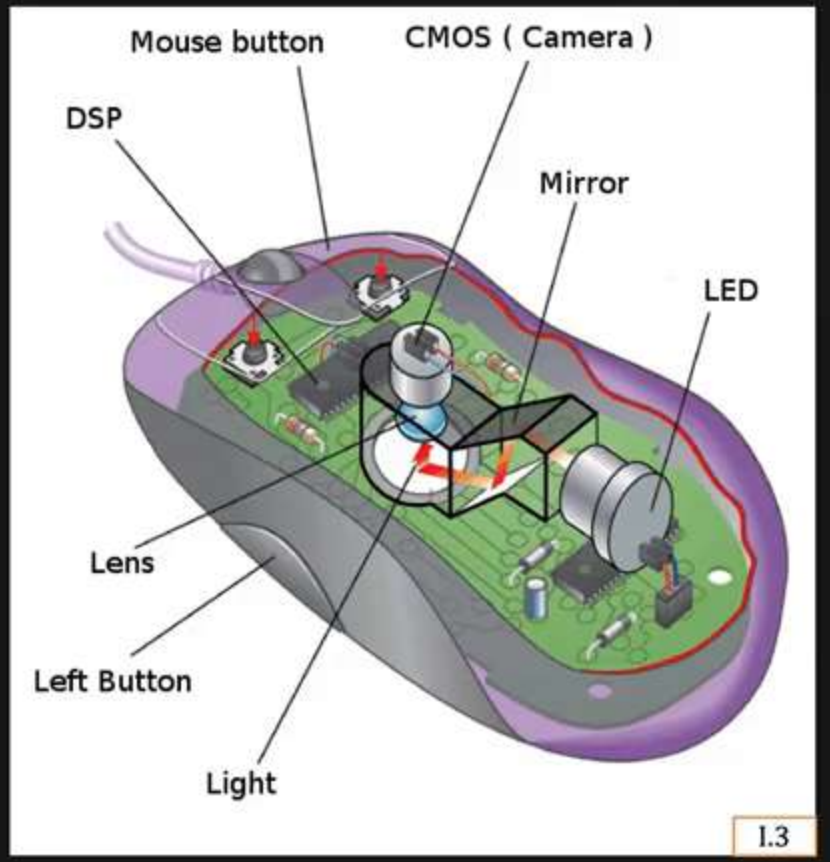
Image I_1 at time t

Search Window S

Image I_2 at time $t + \delta t$

Template matching is slow
when search window S is large.
Also, mismatches are possible.

Optical Mouse



Estimating Mouse Movements

Traffic Monitoring

Finding Velocities of Vehicles

Video Retiming

Optical Flow is used to determine the intermediate frame
to produce slow-motion effect.

Video Retiming

Optical Flow is used to determine the intermediate frame
to produce slow-motion effect.

Video Retiming

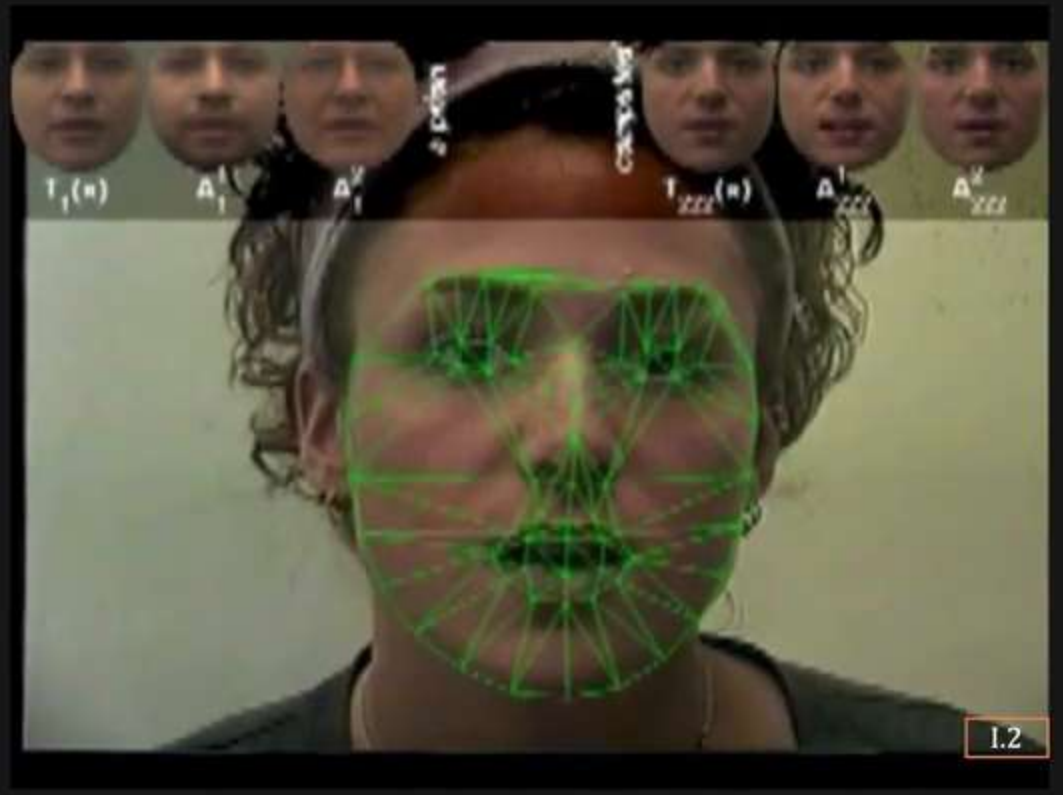
Optical Flow is used to determine the intermediate frames to produce slow-motion effect.

Image Stabilization

Captured Video

Optical Flow is used to remove camera shake.

Face Tracking



Tracking of Facial Features

Games

Flow Based Player Interaction

Structure From Motion

Compute 3D scene structure and camera motion from a sequence of frames.

Topics:

(1) Structure from Motion Problem

(2) SFM Observation Matrix

(3) Rank of Observation Matrix

(4) Tomasi-Kanade Factorization

Feature Detection and Tracking

- Detect feature points: Corners, SIFT points, …
- Track feature points: Template Matching, Optical Flow…

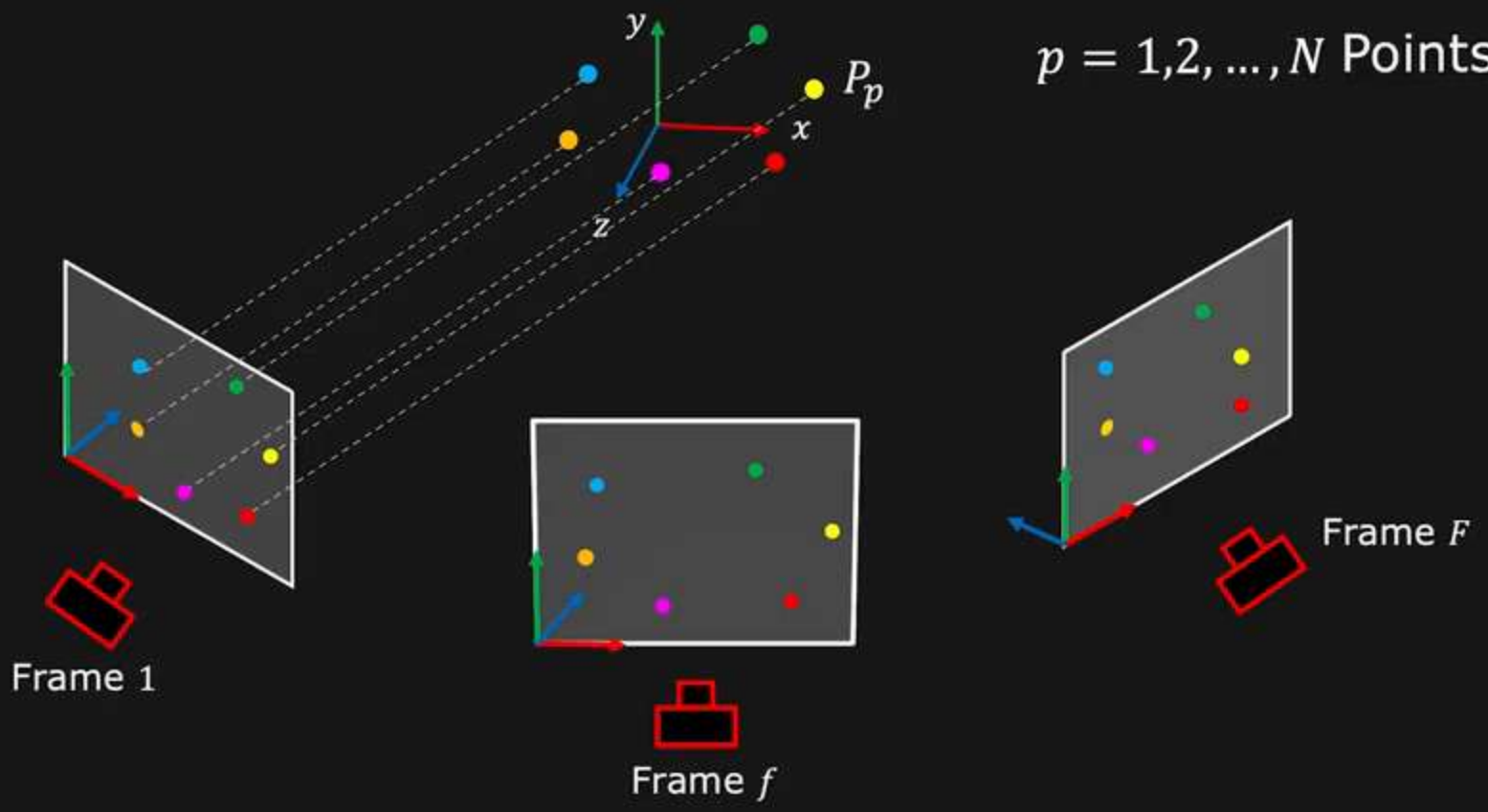
Feature Detection and Tracking

- Detect feature points: Corners, SIFT points, …
- Track feature points: Template Matching, Optical Flow…

Feature Detection and Tracking

- Detect feature points: Corners, SIFT points, …
- Track feature points: Template Matching, Optical Flow…

Orthographic Structure from Motion

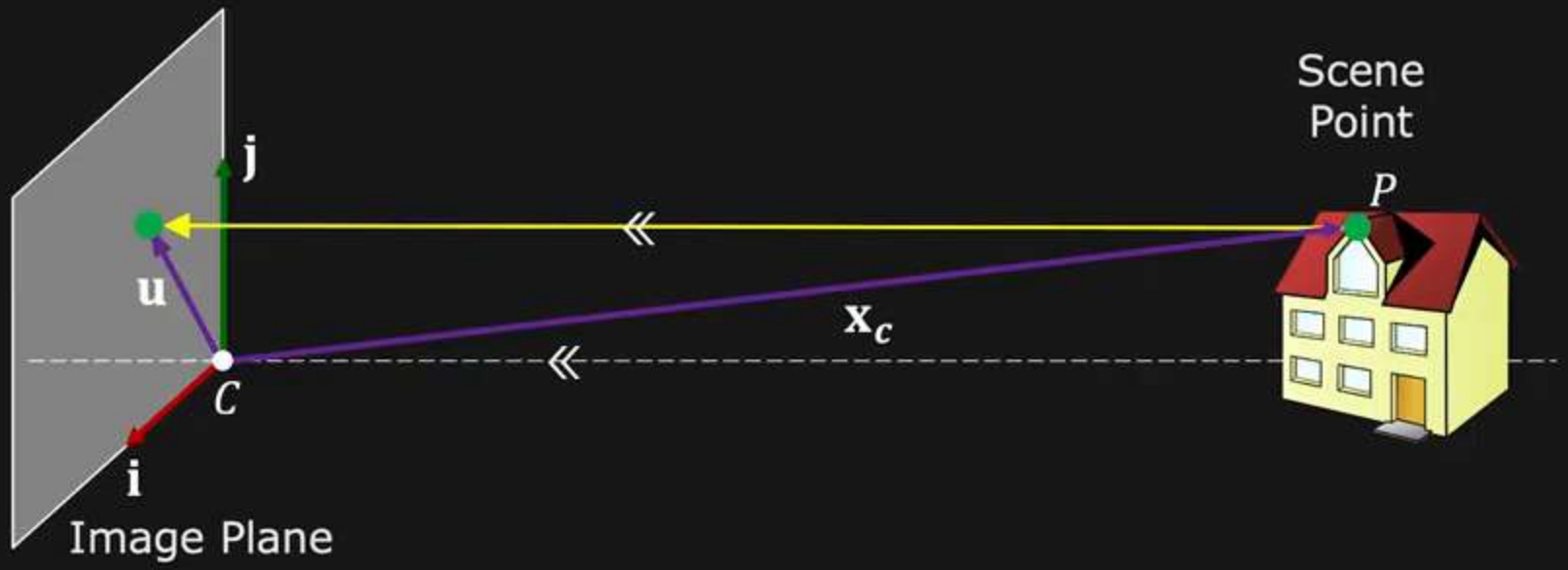


$p = 1,2,\ldots,N$ Points

Given sets of corresponding image points (2D): $(u_{f,p}, v_{f,p})$

Find scene points (3D) P_p, assuming orthographic camera.

From 3D to 2D: Orthographic Projection

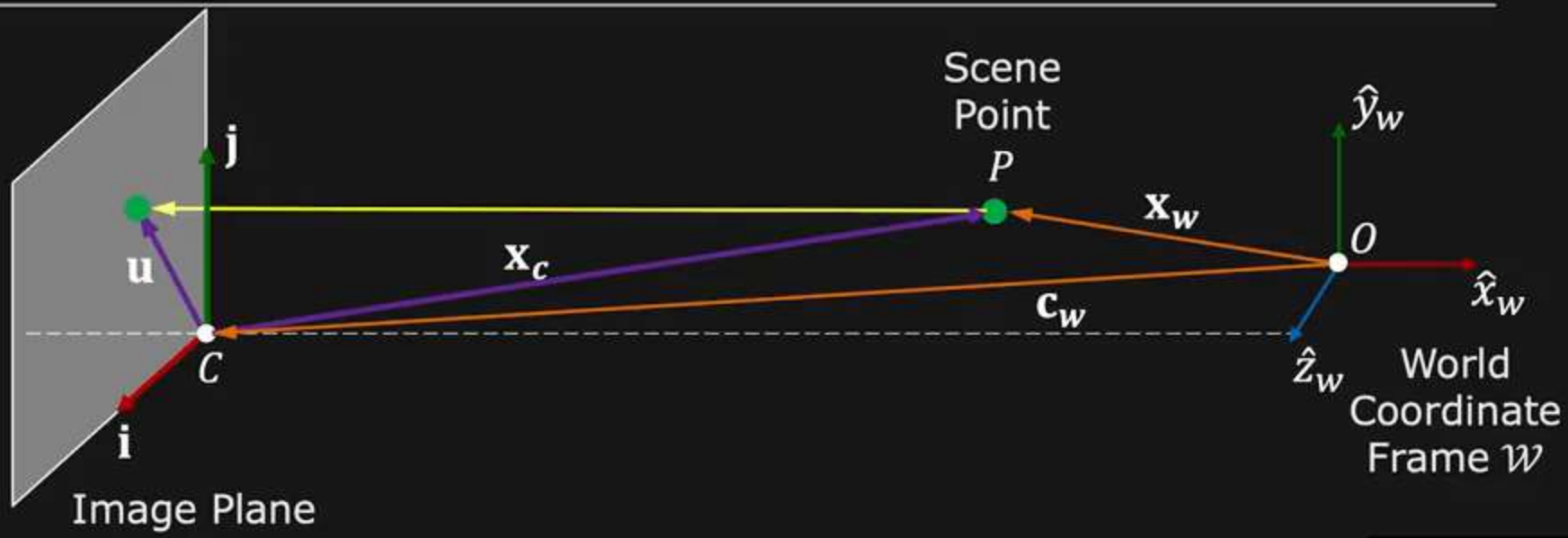


$$u = \mathbf{i} \cdot \mathbf{x}_c = \mathbf{i}^T \mathbf{x}_c$$

$$v = \mathbf{j} \cdot \mathbf{x}_c = \mathbf{j}^T \mathbf{x}_c$$

Perspective cameras exhibit orthographic projection when distance of scene from camera is large compared to depth variation within scene (magnification is nearly constant).

From 3D to 2D: Orthographic Projection

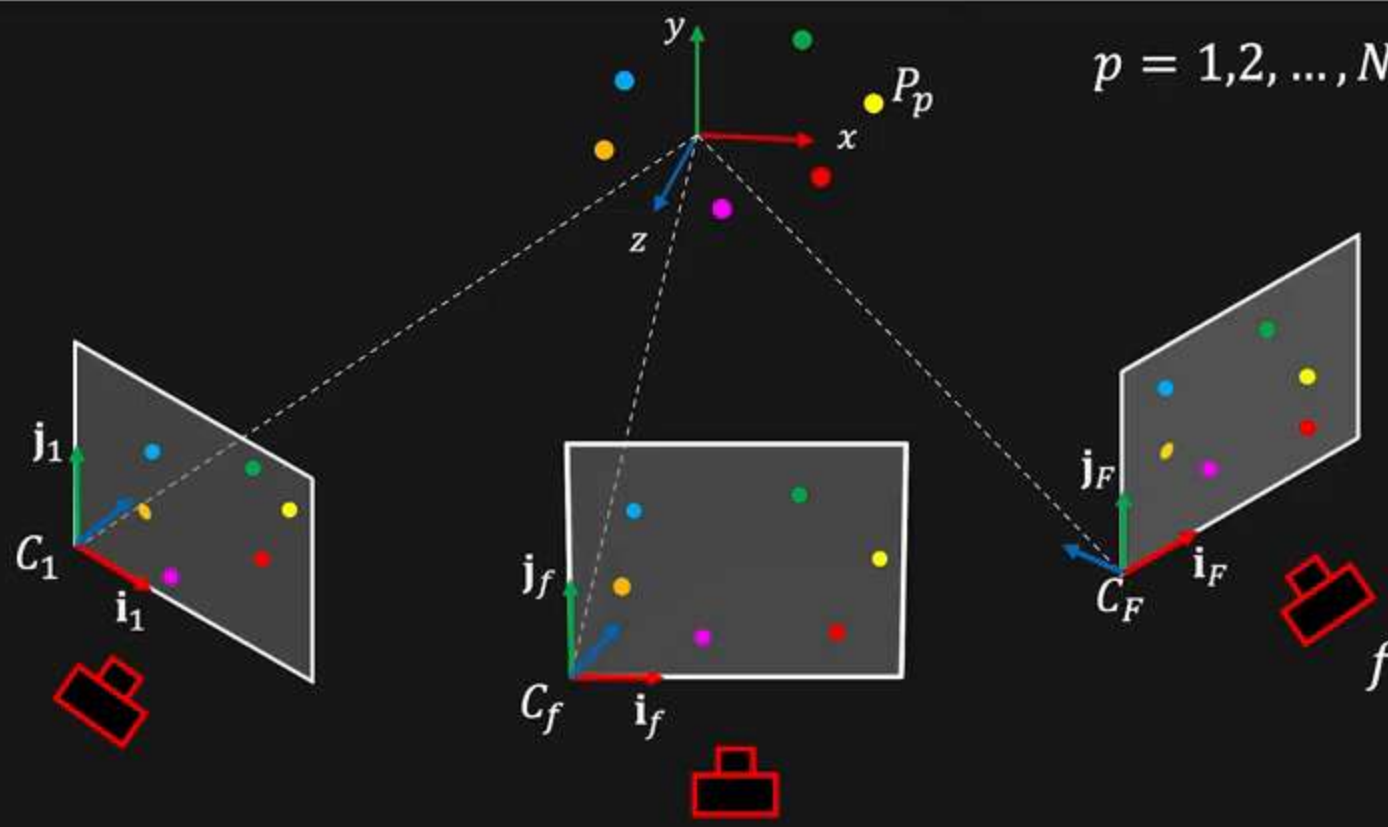


$$u = \mathbf{i}^T \mathbf{x}_c = \mathbf{i}^T (\mathbf{x}_w - \mathbf{c}_w) = \mathbf{i}^T (P - C)$$

$$v = \mathbf{j}^T \mathbf{x}_c = \mathbf{j}^T (\mathbf{x}_w - \mathbf{c}_w) = \mathbf{j}^T (P - C)$$

$$\boxed{\begin{aligned} u &= \mathbf{i}^T (P - C) \\ v &= \mathbf{j}^T (P - C) \end{aligned}}$$

Orthographic SFM



$p = 1,2, \ldots, N$ Points

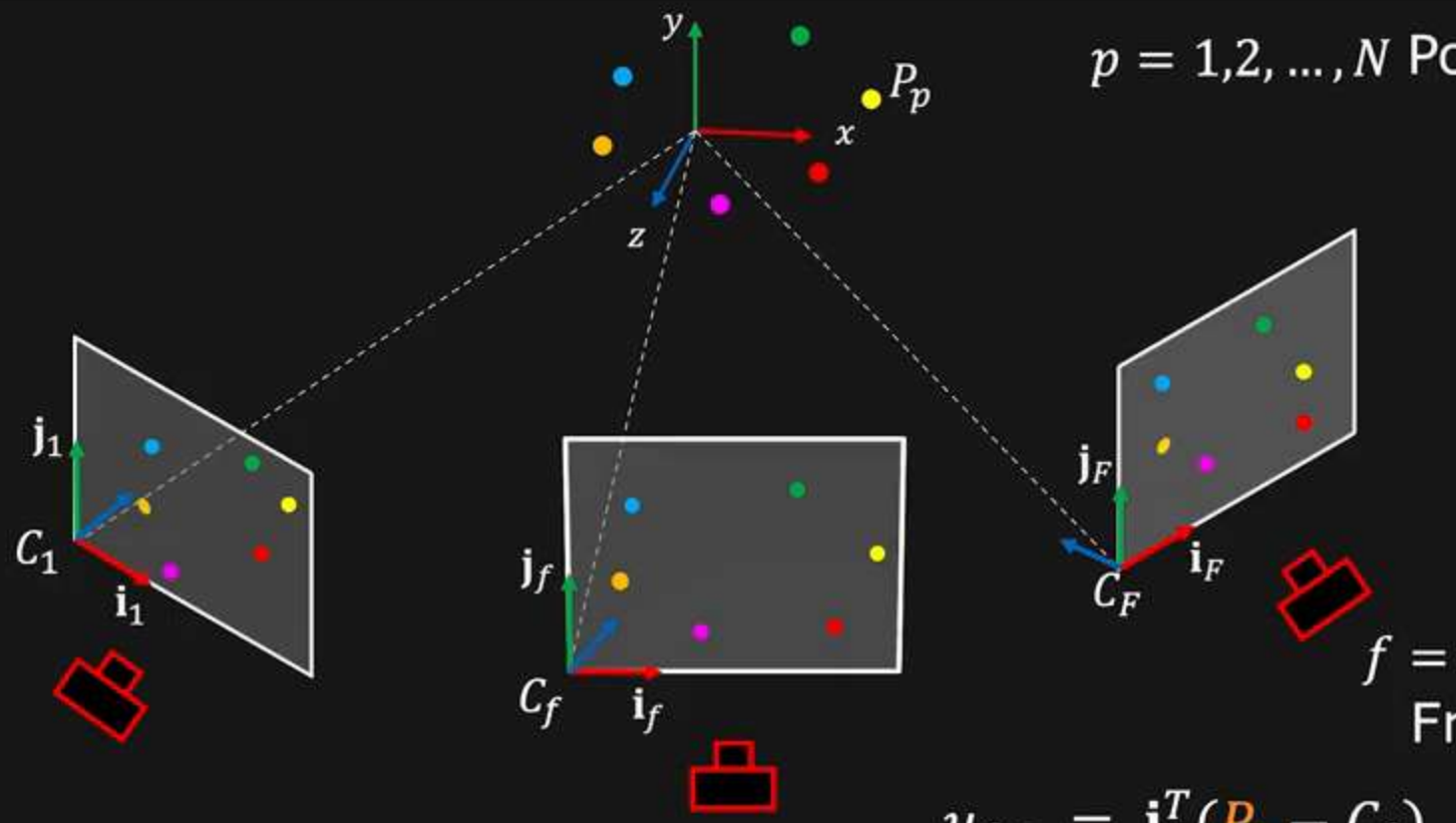
$f = 1,2, \ldots, F$ Frames

Given corresponding image points (2D) $(u_{f,p}, v_{f,p})$

Find scene points $\{P_p\}$.

Camera Positions $\{C_f\}$, camera orientations $\{(\mathbf{i}_f, \mathbf{j}_f)\}$ are unkn

Orthographic SFM



$p = 1,2,\ldots,N$ Points

$f = 1,2,\ldots,F$ Frames

Image of point P_p in camera frame f:

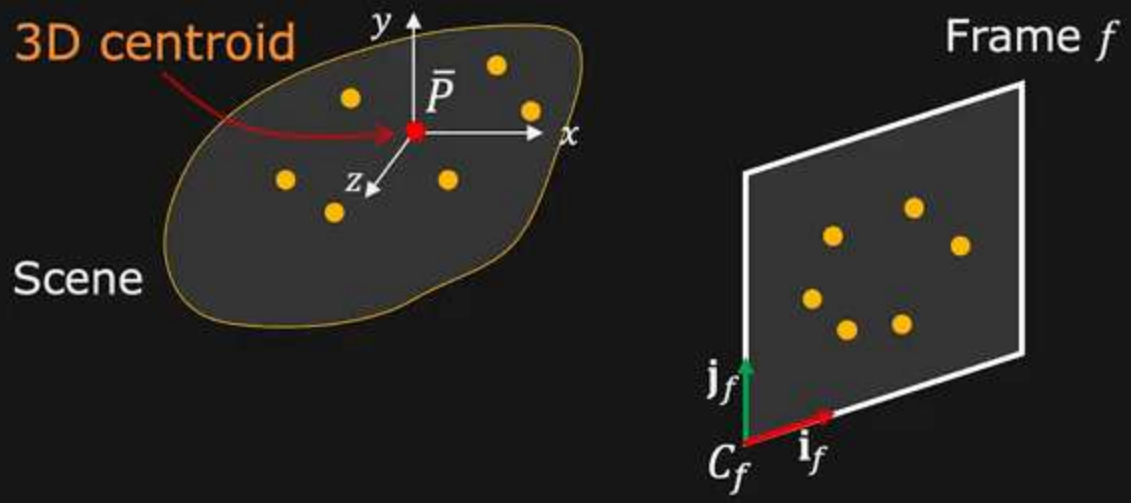
$$u_{f,p} = \mathbf{i}_f^T(P_p - C_f)$$

$$v_{f,p} = \mathbf{j}_f^T(P_p - C_f)$$

Known Unknown

We can remove C_f from equations to simply SFM problem

Centering Trick

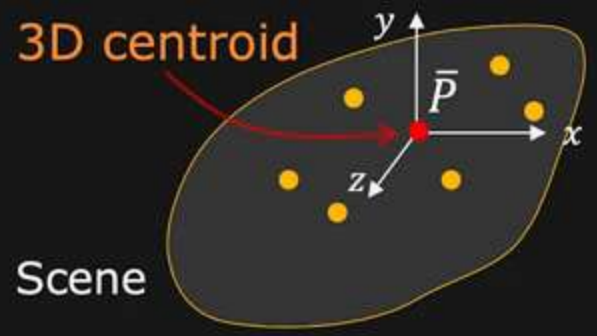


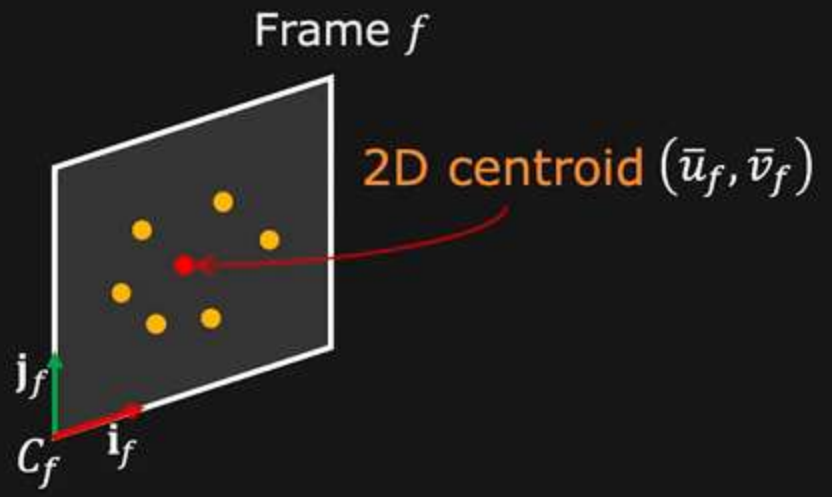
Assume origin of world at centroid of scene points:

$$\frac{1}{N}\sum_{p=1}^{N} P_p = \bar{P} = \mathbf{0}$$

We will compute scene points w.r.t their centroid!

Centering Trick





Centroid $(\bar{u}_f, \bar{v}_f)$ of the image points in frame f:

$$\bar{u}_f = \frac{1}{N}\sum_{p=1}^{N} u_{f,p} = \frac{1}{N}\sum_{p=1}^{N} \mathbf{i}_f^T (P_p - C_f)$$

$$\bar{u}_f = \frac{1}{N}\mathbf{i}_f^T \sum_{p=1}^{N} P_p - \frac{1}{N}\sum_{p=1}^{N} \mathbf{i}_f^T C_f$$

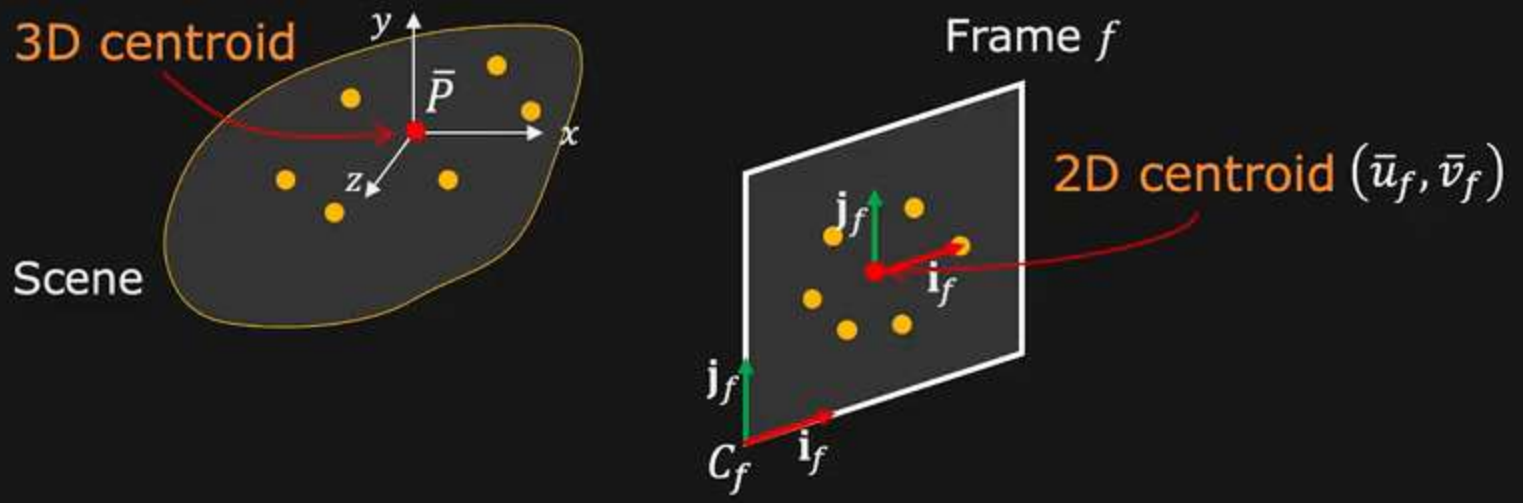
$$\boxed{\bar{u}_f = -\mathbf{i}_f^T C_f}$$

$$\bar{v}_f = \frac{1}{N}\sum_{p=1}^{N} v_{f,p} = \frac{1}{N}\sum_{p=1}^{N} \mathbf{j}_f^T (P_p - C_f)$$

$$\bar{v}_f = \frac{1}{N}\mathbf{j}_f^T \sum_{p=1}^{N} P_p - \frac{1}{N}\sum_{p=1}^{N} \mathbf{j}_f^T C_f$$

$$\boxed{\bar{v}_f = -\mathbf{j}_f^T C_f}$$

Centering Trick



Shift camera origin to the centroid $(\bar{u}_f, \bar{v}_f)$.

Image points w.r.t. $(\bar{u}_f, \bar{v}_f)$:

$$\tilde{u}_{f,p} = u_{f,p} - \bar{u}_f \qquad \tilde{v}_{f,p} = v_{f,p} - \bar{v}_f$$

$$= \mathbf{i}_f^T (P_p - C_f) - \mathbf{i}_f^T C_f \qquad = \mathbf{j}_f^T (P_p - C_f) - \mathbf{j}_f^T C_f$$

$$\boxed{\tilde{u}_{f,p} = \mathbf{i}_f^T P_p} \qquad \boxed{\tilde{v}_{f,p} = \mathbf{j}_f^T P_p}$$

Camera locations C_f now removed from equations.

Observation Matrix W

$$\tilde{u}_{f,p} = \mathbf{i}_f^T P_p$$
$$\tilde{v}_{f,p} = \mathbf{j}_f^T P_p$$

$$\begin{bmatrix} \tilde{u}_{f,p} \\ \tilde{v}_{f,p} \end{bmatrix} = \begin{bmatrix} \mathbf{i}_f^T \\ \mathbf{j}_f^T \end{bmatrix} P_p$$

$$\begin{array}{c} \\ \text{Image 1} \\ \text{Image 2} \\ \\ \text{Image F} \\ \hline \text{Image 1} \\ \text{Image 2} \\ \\ \text{Image F} \end{array} \begin{array}{c} \begin{array}{cccc} \text{Point 1} & \text{Point 2} & & \text{Point N} \end{array} \\ \begin{bmatrix} \tilde{u}_{1,1} & \tilde{u}_{1,2} & \cdots & \tilde{u}_{1,N} \\ \tilde{u}_{2,1} & \tilde{u}_{2,2} & \cdots & \tilde{u}_{2,N} \\ \vdots & \vdots & \vdots & \vdots \\ \tilde{u}_{F,1} & \tilde{u}_{F,2} & \cdots & \tilde{u}_{F,N} \\ \hline \tilde{v}_{1,1} & \tilde{v}_{1,2} & \cdots & \tilde{v}_{1,N} \\ \tilde{u}_{2,1} & \tilde{u}_{2,2} & \cdots & \tilde{v}_{2,N} \\ \vdots & \vdots & \vdots & \vdots \\ \tilde{v}_{F,1} & \tilde{v}_{F,2} & \cdots & \tilde{v}_{F,N} \end{bmatrix} \end{array} = \begin{bmatrix} \mathbf{i}_1^T \\ \mathbf{i}_2^T \\ \vdots \\ \mathbf{i}_F^T \\ \hline \mathbf{j}_1^T \\ \mathbf{j}_2^T \\ \vdots \\ \mathbf{j}_F^T \end{bmatrix} \begin{array}{c} \begin{array}{cccc} \text{Point 1} & \text{Point 2} & & \text{Point N} \end{array} \\ \begin{bmatrix} P_1 & P_2 & \cdots & P_N \end{bmatrix} \end{array}$$

$W_{2F \times N}$
Centroid-Subtracted Feature Points (Known)

$M_{2F \times 3}$
Camera Motion (Unknown)

$S_{3 \times N}$
Scene Structure (Unknown)

Observation Matrix W

$$
\begin{array}{c}
\\
\text{Image 1} \\
\text{Image 2} \\
\\
\text{Image F} \\
\text{Image 1} \\
\text{Image 2} \\
\\
\text{Image F}
\end{array}
\begin{array}{c}
\begin{array}{cccc} \text{Point 1} & \text{Point 2} & & \text{Point N} \end{array} \\
\begin{bmatrix}
\tilde{u}_{1,1} & \tilde{u}_{1,2} & \cdots & \tilde{u}_{1,N} \\
\tilde{u}_{2,1} & \tilde{u}_{2,2} & \cdots & \tilde{u}_{2,N} \\
\vdots & \vdots & \vdots & \vdots \\
\tilde{u}_{F,1} & \tilde{u}_{F,2} & \cdots & \tilde{u}_{F,N} \\
\tilde{v}_{1,1} & \tilde{v}_{1,2} & \cdots & \tilde{v}_{1,N} \\
\tilde{u}_{2,1} & \tilde{u}_{2,2} & \cdots & \tilde{v}_{2,N} \\
\vdots & \vdots & \vdots & \vdots \\
\tilde{v}_{F,1} & \tilde{v}_{F,2} & \cdots & \tilde{v}_{F,N}
\end{bmatrix}
\end{array}
=
\begin{bmatrix}
\mathbf{i}_1^T \\
\mathbf{i}_2^T \\
\vdots \\
\mathbf{i}_F^T \\
\mathbf{j}_1^T \\
\mathbf{j}_2^T \\
\vdots \\
\mathbf{j}_F^T
\end{bmatrix}
\begin{array}{c}
\begin{array}{cccc} \text{Point 1} & \text{Point 2} & & \text{Point N} \end{array} \\
\begin{bmatrix} P_1 & P_2 & \cdots & P_N \end{bmatrix}
\end{array}
$$

$W_{2F\times N}$
Centroid-Subtracted Feature Points (Known)

$M_{2F\times 3}$
Camera Motion (Unknown)

$S_{3\times N}$
Scene Structure (Unknown)

Can we find M and S from W?

Linear Independence of Vectors

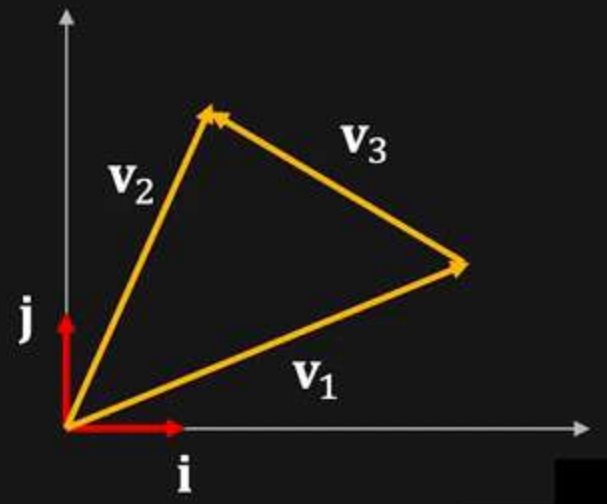
A set of vectors $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$ is said to be linearly independent if no vector can be represented as a weighted linear sum of the others.

$\{\mathbf{i}, \mathbf{j}\}$ is linearly independent.

$\{\mathbf{i}, \mathbf{j}, \mathbf{v}_1\}$ is linearly dependent.

$\{\mathbf{i}, \mathbf{j}, \mathbf{v}_3\}$ is linearly dependent.

$\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$ is linearly dependent.



Rank of a Matrix

Column Rank: The number of linearly independent columns of the matrix.

Row Rank: The number of linearly independent rows of the matrix.

$$m \begin{bmatrix} & A & \\ & & \\ & n & \end{bmatrix} = [\mathbf{c}_1 \quad \mathbf{c}_2 \quad \dots \quad \mathbf{c}_n] = \begin{bmatrix} \mathbf{r}_1^T \\ \mathbf{r}_2^T \\ \vdots \\ \mathbf{r}_m^T \end{bmatrix}$$

$$ColumnRank(A) \leq n \qquad RowRank(A) \leq m$$

$$ColumnRank(A) = RowRank(A) = Rank(A)$$

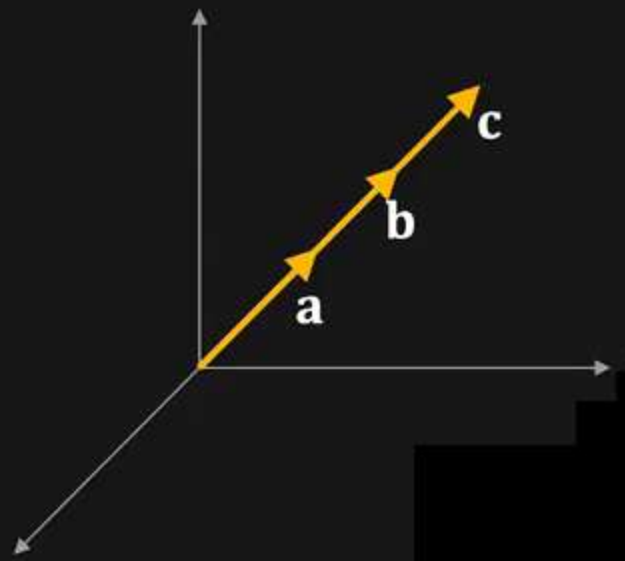
$$Rank(A) \leq \min(m, n)$$

Geometric Meaning of Matrix Rank

Rank is the dimensionality of the space spanned by column or row vectors of the matrix.

$$A = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix} = [\mathbf{a} \quad \mathbf{b} \quad \mathbf{c}]$$

$$Rank(A) = 1 \quad \Longleftrightarrow$$

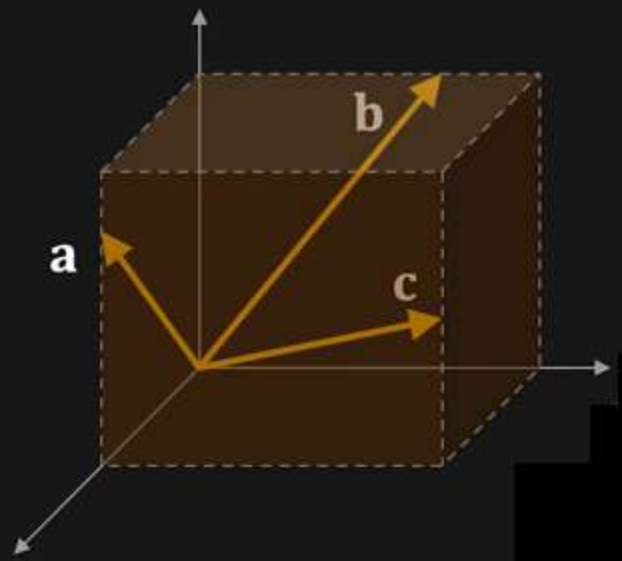


Geometric Meaning of Matrix Rank

Rank is the dimensionality of the space spanned by column or row vectors of the matrix.

$$A = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix} = [\mathbf{a} \quad \mathbf{b} \quad \mathbf{c}]$$

$$Rank(A) = 3 \quad \Longleftrightarrow$$



Important Properties of Matrix Rank

- $Rank(A^T) = Rank(A)$
- $Rank\left(A_{m\times n}B_{n\times p}\right) = \min\left(Rank(A_{m\times n}), Rank\left(B_{n\times p}\right)\right)$
$\leq \min(m, n, p)$
- $Rank(AA^T) = Rank(A^TA) = Rank(A^T) = Rank(A)$
- $A_{m\times m}$ is invertible iff $Rank(A_{m\times m}) = m$

...Back to Observation Matrix W

$$
\begin{array}{c}
\\
\text{Image 1} \\
\text{Image 2} \\
\\
\text{Image F} \\
\text{Image 1} \\
\text{Image 2} \\
\\
\text{Image F}
\end{array}
\begin{array}{c}
\begin{array}{cccc} \text{Point 1} & \text{Point 2} & & \text{Point N} \end{array} \\
\begin{bmatrix}
\tilde{u}_{1,1} & \tilde{u}_{1,2} & \cdots & \tilde{u}_{1,N} \\
\tilde{u}_{2,1} & \tilde{u}_{2,2} & \cdots & \tilde{u}_{2,N} \\
\vdots & \vdots & \vdots & \vdots \\
\tilde{u}_{F,1} & \tilde{u}_{F,2} & \cdots & \tilde{u}_{F,N} \\
\tilde{v}_{1,1} & \tilde{v}_{1,2} & \cdots & \tilde{v}_{1,N} \\
\tilde{u}_{2,1} & \tilde{u}_{2,2} & \cdots & \tilde{v}_{2,N} \\
\vdots & \vdots & \vdots & \vdots \\
\tilde{v}_{F,1} & \tilde{v}_{F,2} & \cdots & \tilde{v}_{F,N}
\end{bmatrix}
\end{array}
=
\begin{bmatrix}
\mathbf{i}_1^T \\
\mathbf{i}_2^T \\
\vdots \\
\mathbf{i}_F^T \\
\mathbf{j}_1^T \\
\mathbf{j}_2^T \\
\vdots \\
\mathbf{j}_F^T
\end{bmatrix}
\begin{array}{c}
\begin{array}{cccc} \text{Point 1} & \text{Point 2} & & \text{Point N} \end{array} \\
\begin{bmatrix} P_1 & P_2 & \cdots & P_N \end{bmatrix}
\end{array}
$$

$W_{2F\times N}$
Centroid-Subtracted Feature Points (Known)

$M_{2F\times 3}$
Camera Motion (Unknown)

$S_{3\times N}$
Scene Structure (Unknown)

Rank of Observation Matrix

$$\underset{2F\times N}{W} = \underset{2F\times 3}{M} \times \underset{3\times N}{S}$$

We know:

$$Rank(MS) \leq Rank(M) \qquad Rank(MS) \leq Rank(S)$$

$$\Rightarrow \quad Rank(MS) \leq \min(3,2F) \qquad Rank(MS) \leq \min(3,N)$$

$$\Rightarrow \quad Rank(W) = Rank(MS) \leq \min(3,N,2F)$$

Rank Theorem: $Rank(W) \leq 3$

Rank of Observation Matrix

$$\underset{2F\times N}{W} = \underset{2F\times 3}{M} \times \underset{3\times N}{S}$$

We know:

$$Rank(MS) \leq Rank(M) \qquad Rank(MS) \leq Rank(S)$$

$$\Rightarrow \quad Rank(MS) \leq \min(3,2F) \qquad Rank(MS) \leq \min(3,N)$$

$$\Rightarrow \quad Rank(W) = Rank(MS) \leq \min(3,N,2F)$$

Rank Theorem: $Rank(W) \leq 3$

We can "factorize" W into M and S!

Singular Value Decomposition (SVD)

For any matrix A there exists a factorization:

$$A_{M\times N} = U_{M\times M} \cdot \Sigma_{M\times N} \cdot V^T{}_{N\times N}$$

where U and V^T are orthonormal and Σ is diagonal.

MATLAB: `[U,S,V] = svd(A)`

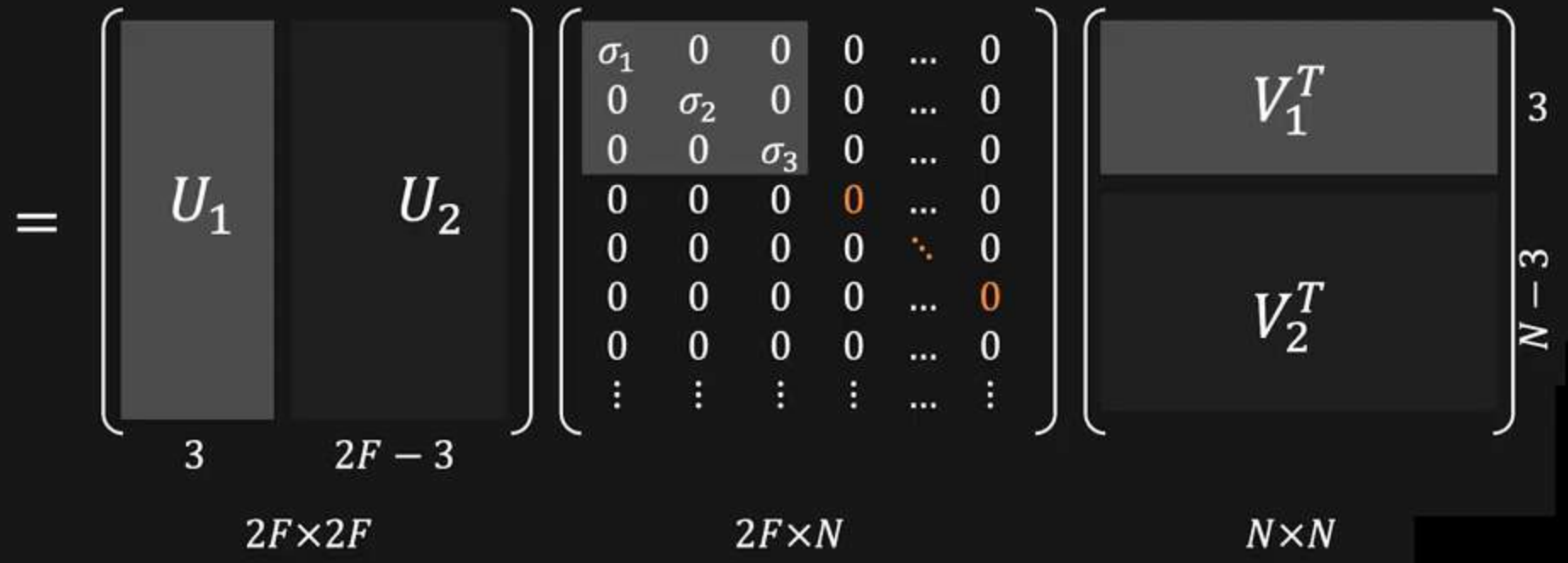
$$\Sigma_{M\times N} = \begin{pmatrix} \sigma_1 & 0 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & 0 & \dots & 0 \\ 0 & 0 & 0 & \sigma_4 & \dots & 0 \\ 0 & 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & \dots & \sigma_N \\ 0 & 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \dots & \vdots \end{pmatrix}$$

$\sigma_1, \dots, \sigma_N$: Singular Values

If $Rank(A) = r$ then A has r non-zero singular values.

Enforcing Rank Constraint

Using SVD: $W = U \Sigma V^T$

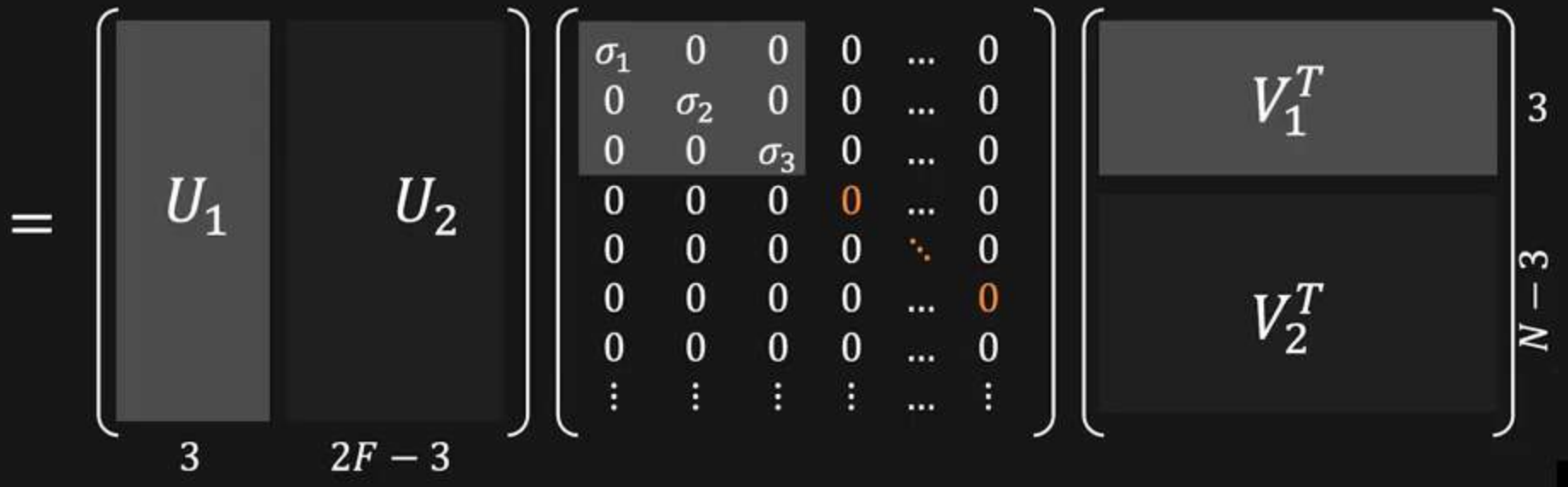


Since $Rank(W) \leq 3$, $Rank(\Sigma) \leq 3$.

Submatrices U_2 and V_2^T do not contribute to W.

Enforcing Rank Constraint

Using SVD: $W = U \Sigma V^T$



$$W = U_1 \ \Sigma_1 \ V_1^T$$

$(2F \times 3)(3 \times 3)(3 \times P)$

Factorization (Finding M, S)

$$W = \underbrace{U_1 (\Sigma_1)^{1/2}}_{(2F\times 3)} \underbrace{(\Sigma_1)^{1/2} V_1^T}_{(3\times N)}$$

$= M?$ $= S?$

Not so fast. Decomposition not unique!

For any 3x3 non-singular matrix Q:

$$W = \underbrace{U_1 (\Sigma_1)^{1/2} Q}_{(2F\times 3)} \underbrace{Q^{-1} (\Sigma_1)^{1/2} V_1^T}_{(3\times N)}$$ is also vali

$= M$ $= S$ ... for some Q.

How to find the matrix Q ?

Orthonormality of M

The Motion Matrix M:

$$M = \begin{bmatrix} \mathbf{i}_1^T \\ \vdots \\ \mathbf{i}_F^T \\ \mathbf{j}_1^T \\ \vdots \\ \mathbf{j}_F^T \end{bmatrix} = \underbrace{U_1(\Sigma_1)^{1/2}}_{\text{Computed}} Q = \underbrace{\begin{bmatrix} \hat{\mathbf{i}}_1^T \\ \vdots \\ \hat{\mathbf{i}}_F^T \\ \hat{\mathbf{j}}_1^T \\ \vdots \\ \hat{\mathbf{j}}_F^T \end{bmatrix}}_{\text{Computed}} Q = \begin{bmatrix} \hat{\mathbf{i}}_1^T Q \\ \vdots \\ \hat{\mathbf{i}}_F^T Q \\ \hat{\mathbf{j}}_1^T Q \\ \vdots \\ \hat{\mathbf{j}}_F^T Q \end{bmatrix}$$

Orthonormality Constraints:

$$\mathbf{i}_f \cdot \mathbf{i}_f = \mathbf{i}_f^T \mathbf{i}_f = 1$$

$$\mathbf{j}_f \cdot \mathbf{j}_f = \mathbf{j}_f^T \mathbf{j}_f = 1$$

$$\mathbf{i}_f \cdot \mathbf{j}_f = \mathbf{i}_f^T \mathbf{j}_f = 0$$

$$\Rightarrow$$

$$\hat{\mathbf{i}}_f^T Q Q^T \hat{\mathbf{i}}_f = 1$$

$$\hat{\mathbf{j}}_f^T Q Q^T \hat{\mathbf{j}}_f = 1$$

$$\hat{\mathbf{i}}_f^T Q Q^T \hat{\mathbf{j}}_f = 0$$

Orthonormality of M

- We have computed $(\hat{\mathbf{i}}_f^T, \hat{\mathbf{j}}_f^T)$ for $f = 1, \ldots, F$.

$$\left.\begin{aligned} \hat{\mathbf{i}}_f^T Q Q^T \hat{\mathbf{i}}_f &= 1 \\ \hat{\mathbf{j}}_f^T Q Q^T \hat{\mathbf{j}}_f &= 1 \\ \hat{\mathbf{i}}_f^T Q Q^T \hat{\mathbf{j}}_f &= 0 \end{aligned}\right\} \; Q \text{ is unknown.}$$

- Q is 3×3 matrix, 9 variables, $3F$ quadratic equations.
- Q can be solved with 3 or more images ($F \geq 3$) using Newton's method.

Final Solution:

$$M = U_1 (\Sigma_1)^{1/2} Q$$

Camera Motion

$$S = Q^{-1} (\Sigma_1)^{1/2} V_1^T$$

Scene Structure

Summary: Orthographic SFM

1. Detect and track feature points.
2. Create the centroid subtracted matrix W of corresponding feature points.
3. Compute SVD of W and enforce rank constraint.

$$W = \underset{(2F\times 3)}{U} \underset{(3\times 3)}{\Sigma} \underset{(3\times P)}{V^T} = U_1 \Sigma_1 V_1^T$$

4. Set $M = U_1 (\Sigma_1)^{1/2} Q$ and $S = Q^{-1} (\Sigma_1)^{1/2} V_1^T$.
5. Find Q by enforcing the orthonormality constraint.

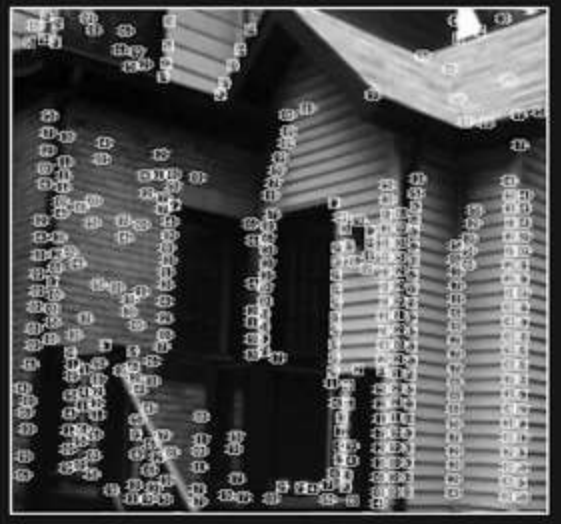
Results

Input Image Sequence

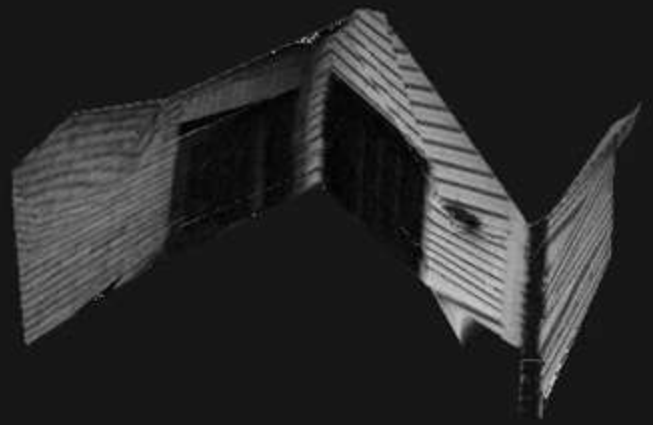
Estimated 3D Points

Results

Input Image Sequence



Tracked Features



3D Reconstruction

3D Reconstruction

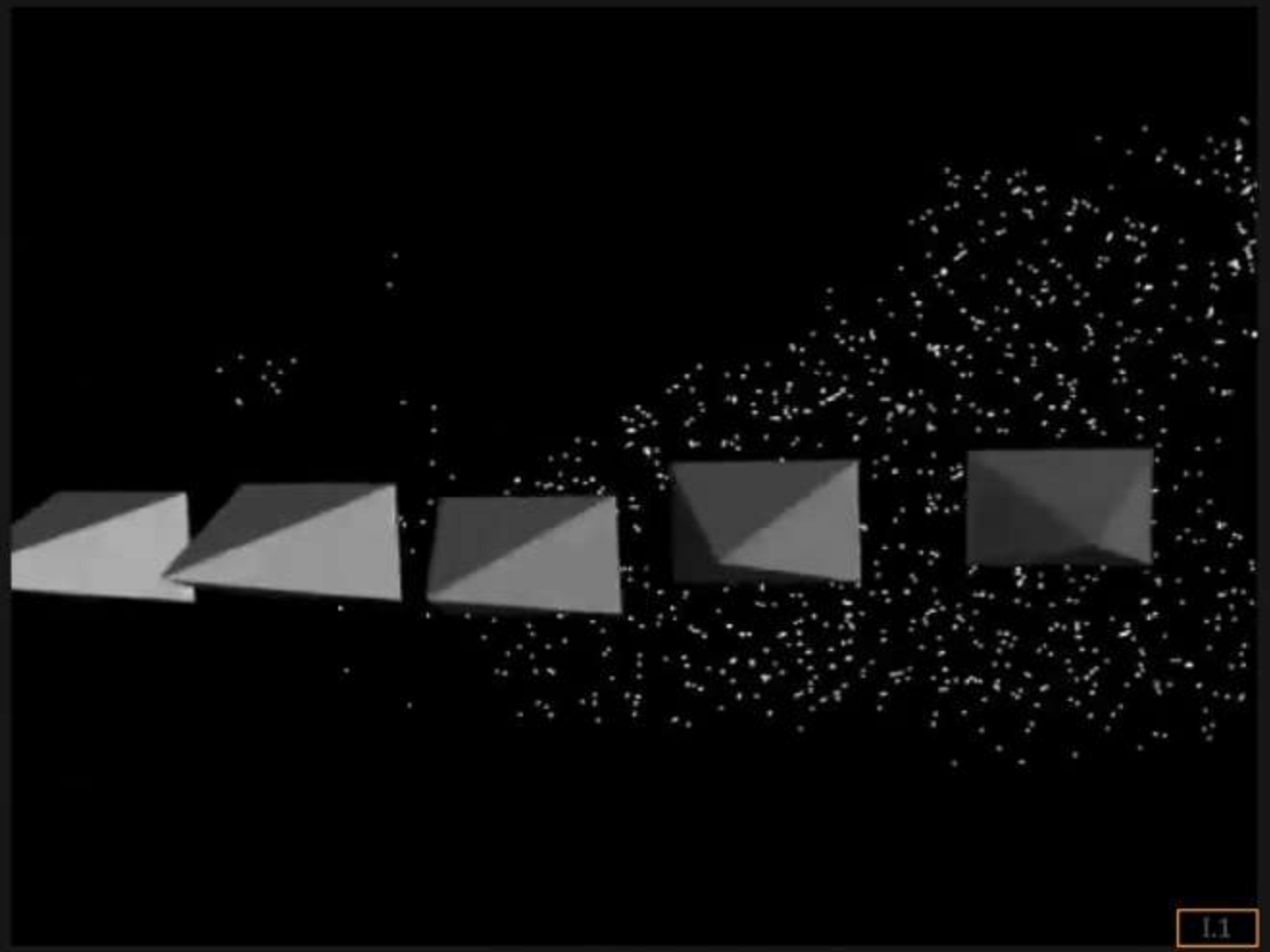
Structure From Motion: Result

Structure From Motion: Result

Structure From Motion: Result

Structure From Motion: Result

Structure From Motion: Result



Structure From Motion: Result

Structure From Motion: Result

Structure From Motion: Result

Structure From Motion: Result

Thank You

Kumar Bipin | linux.kbp@gmail.com