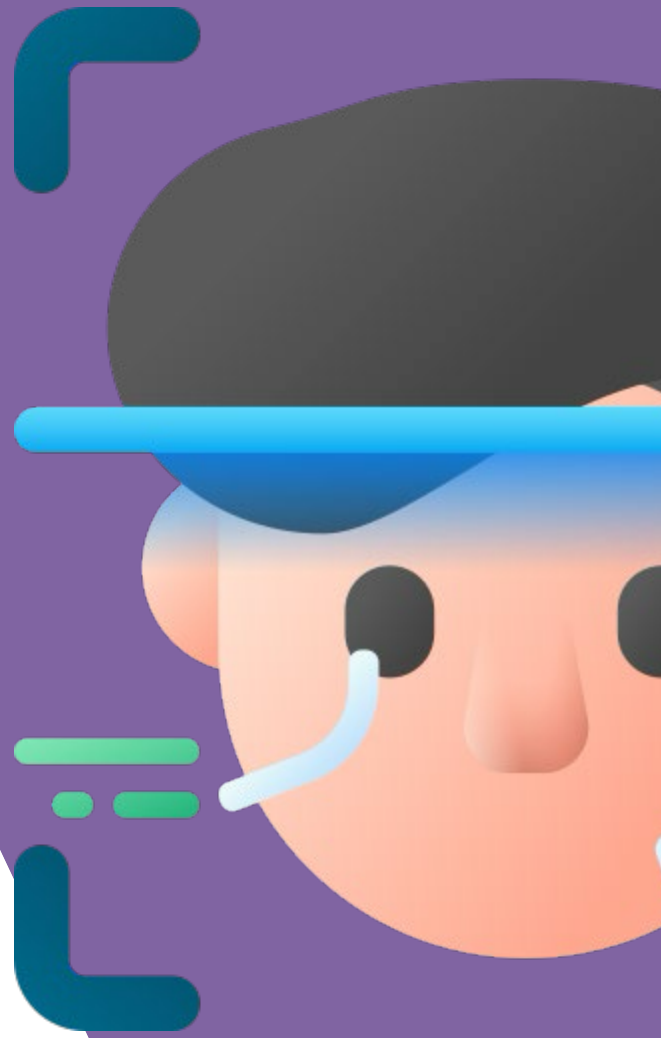
Computer Vision

A.A. 2024-2025

Lecture 12: Camera Calibration and Stereo Vision



References

- Chapter 4 Zissermann - Estimation of 2D Projective Transformations
- Chapter 7 Zissermann - Computation of the Camera Matrix P
- Chapter 8 More on single view geometry

Common Transformations

- Slide credit (next few slides): A. Efros and/or S. Seitz

Original

Cameras are dimensionality reduction machines, as they take the 3D world and turn it into 2D.

This is a projective transformation, as we are projecting a 3D space into a 2D space.

Transformed

Translation

Rotation

Scaling

Affine

Perspective

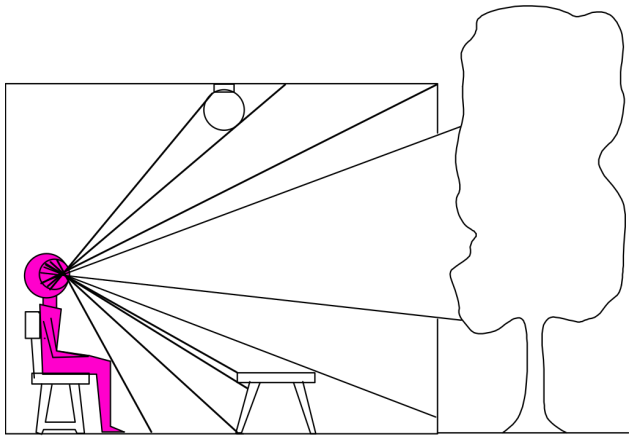
Cameras are Projective Transformation Machines

- Figures © Stephen E. Palmer, 2002

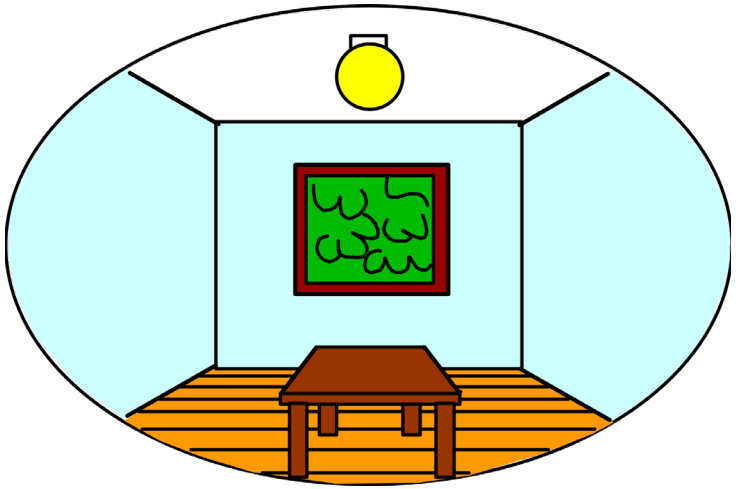
Projective Transformation

3D world

2D image



Point of observation



Camera calibration

- Computes 3D (real world) – 2D camera image relationship
- Necessary to recover 3D metric from image(s):
  - 3D reconstruction
  - Object/camera localization
  - etc..

Camera Obscura: dark room

Known during classical period in China and Greece (e.g., Mo-Ti, China, 470BC to 390BC)

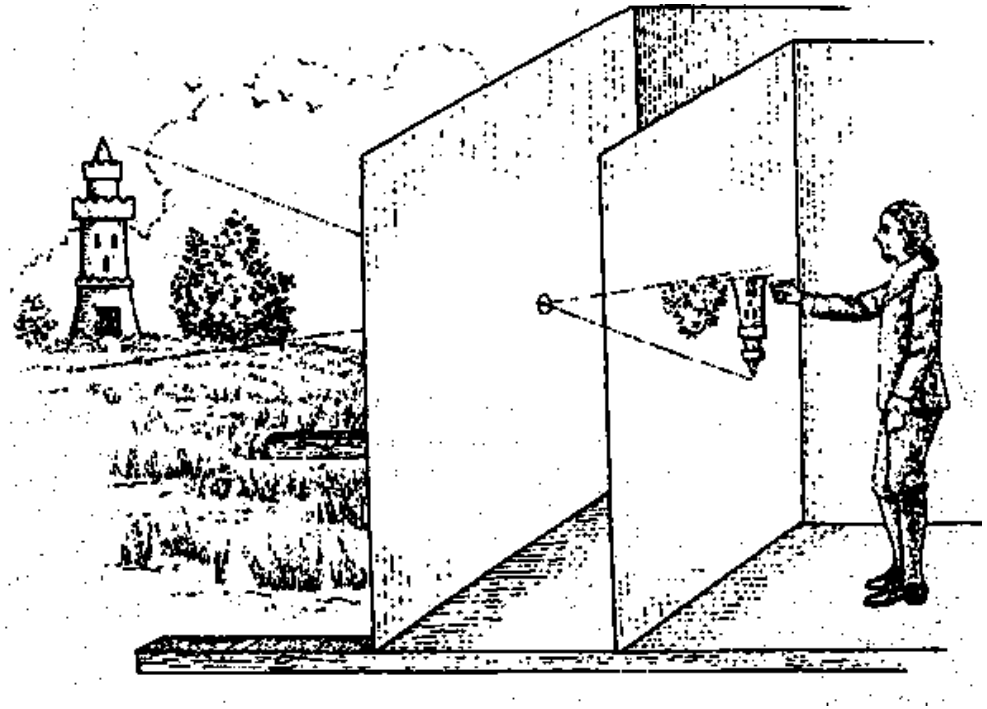


Illustration of Camera Obscura

In these very early ‘cameras’, light from a scene passed through a small opening (pinhole) and was projected onto a wall on the inside

First Photograph

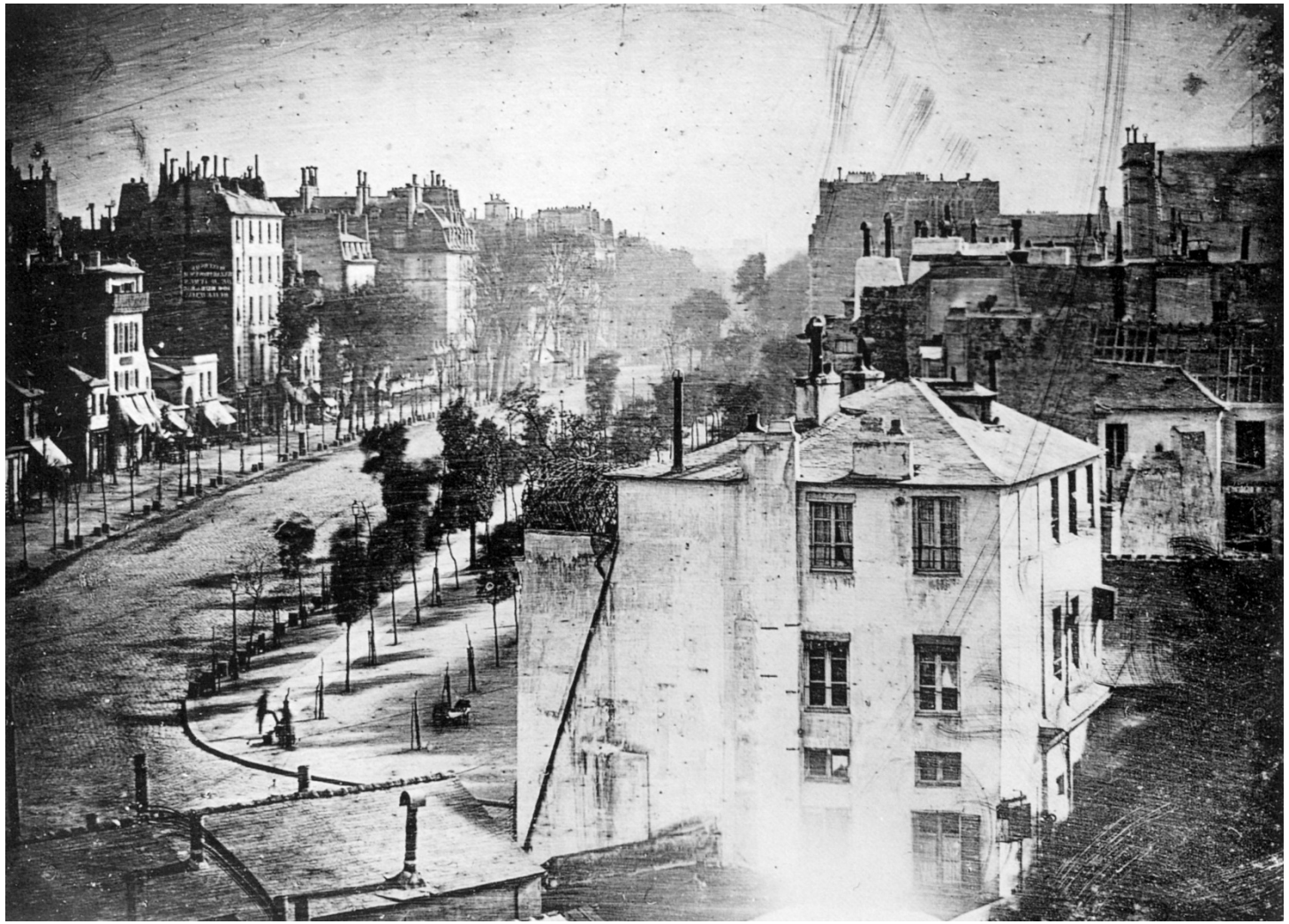
Oldest surviving photograph

- Took 8 hours on pewter plate (with some reports claiming it was several days)

Joseph Niepce, 1826

Stored at UT Austin

First Photograph with People

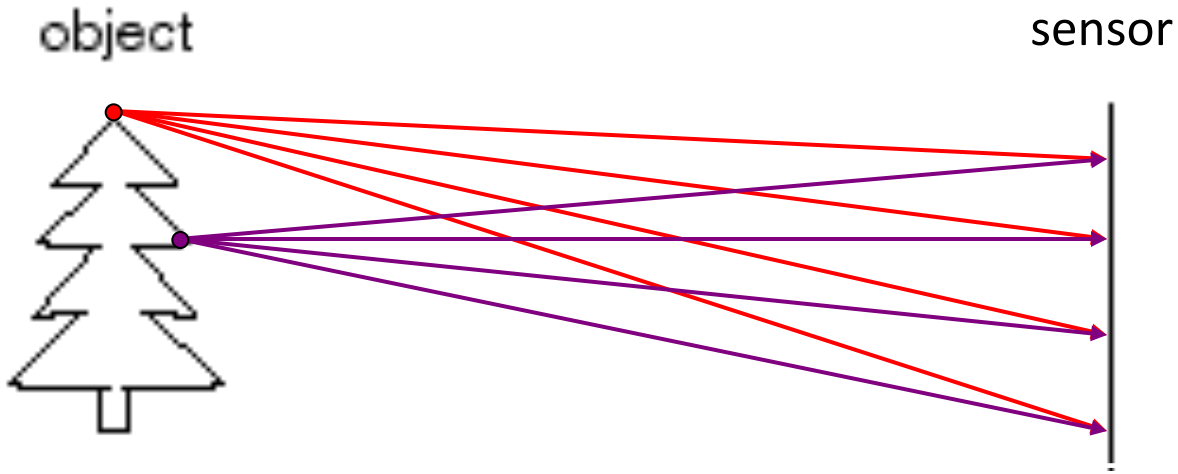


Boulevard du Temple, Louis Daguerre, 1838. Wikipedia.

Design a Camera

Idea 1: Put a sensor in front of an object

Do we get a reasonable image?

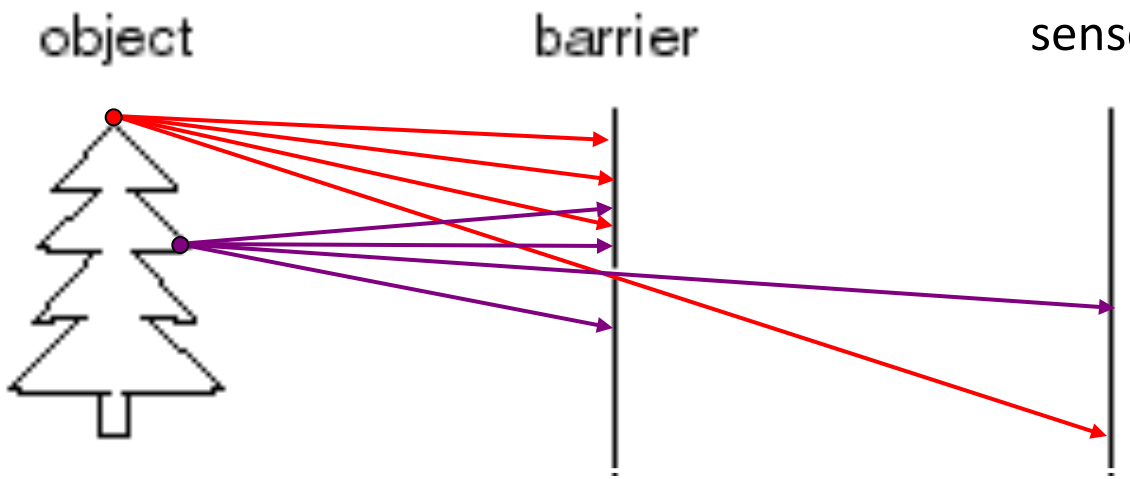


The problem with this naïve approach is that light rays from every point in the 3D world hit every point on the sensor, which results in a blurry image

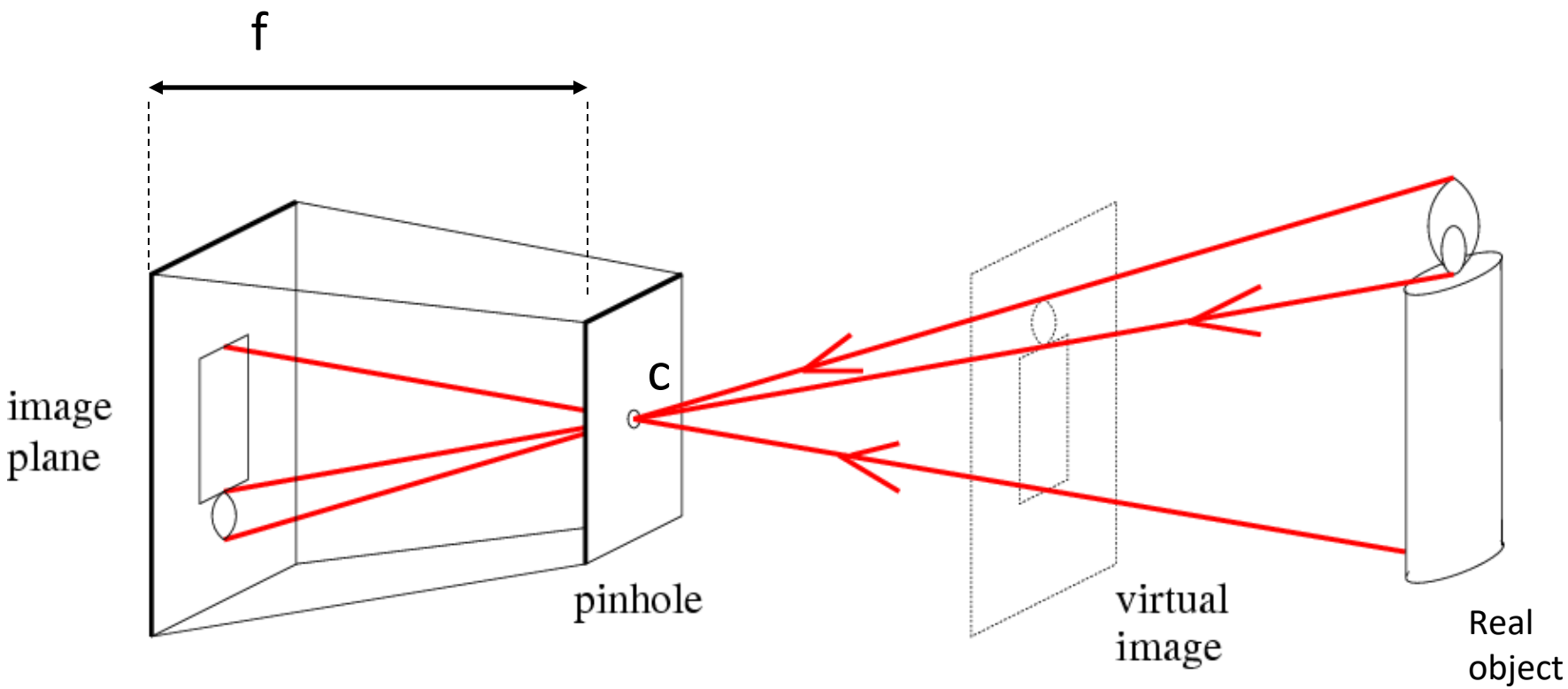
Design a Camera

Idea 2: Add a barrier to block most rays

- Pinhole in barrier
- Only sense light from one direction.
  - Reduces blurring.
- In most cameras, this **aperture** can vary in size.
- How does this transform the image? It gets inverted (upside down)



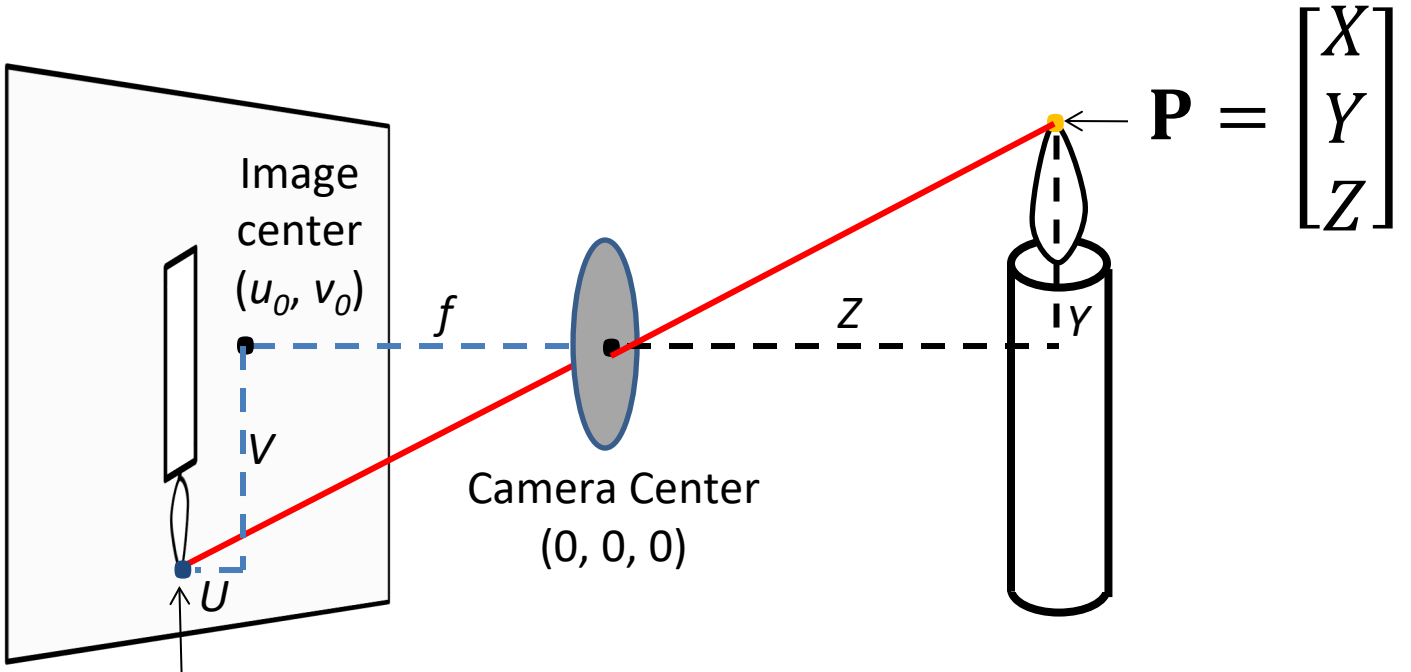
Pinhole Camera Model



f = Focal length

c = Optical center of the camera

Camera Projection

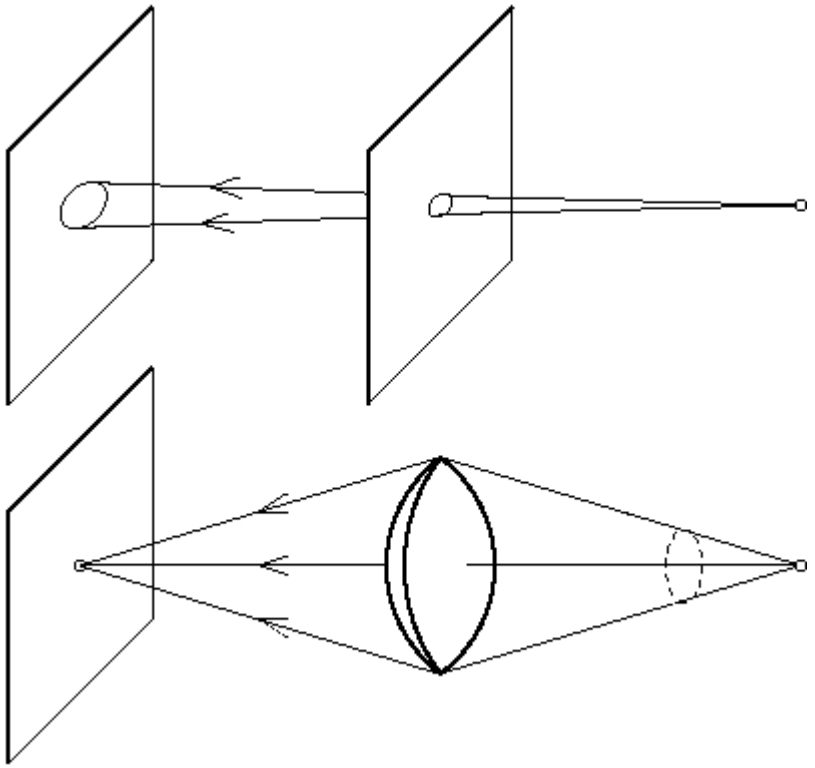


f = distance from image center

$$U = -X * \frac{f}{Z} \qquad V = -Y * \frac{f}{Z}$$

In **perspective projection,** 3D points in camera coordinates are mapped to the image plane by **dividing** them **by their Z component** and multiplying with the focal length.

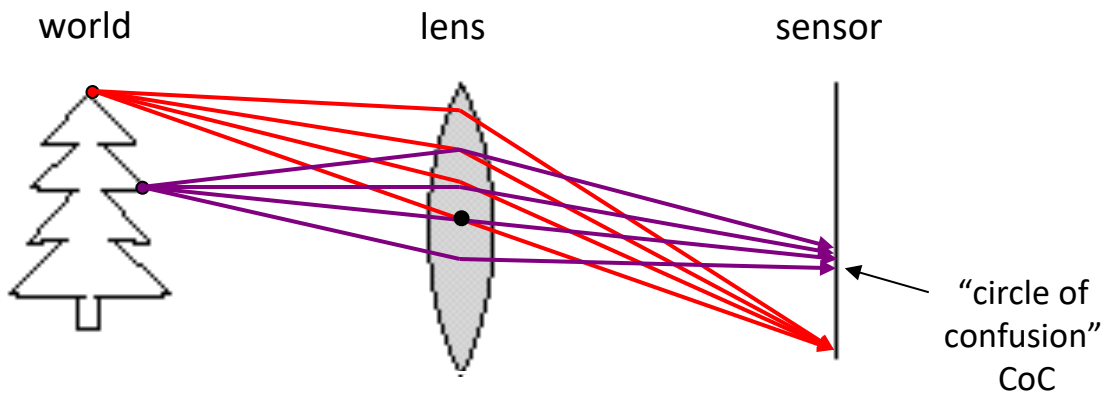
Lenses: The Need



Pinhole cameras are limited by the geometry of the aperture.
No control is possible without changing aperture.

Focus and Defocus

- A lens focuses light onto the film
  - There is a specific distance at which objects are “in focus”
    - other points project to a “circle of confusion” in the image
  - Changing the shape of the lens changes this distance



Camera calibration

Method to find a camera's internal and external parameters

We are going to see:

- Linear Camera Model
- Camera Calibration
- Extracting Intrinsic and Extrinsic Matrices
- Stereo matching

The recipe

In order to do the recontruction of a scene in 3D we need two things:

  - position and orientation of the camera wrt the world coordinate frames (external parameters of the camera)
  - how the camera maps (perspective projection) points in the world onto its image plane (internal parameters of the camera such as focal lenght)

- So we need to estimate intrinsic and extrinsic parameters but first of all we need a model for the camera (linear model): projection matrix

- From the projection matrix we can derive the internal and external parameters of the camera

Forwarding Imaging Model: 3D to 2D

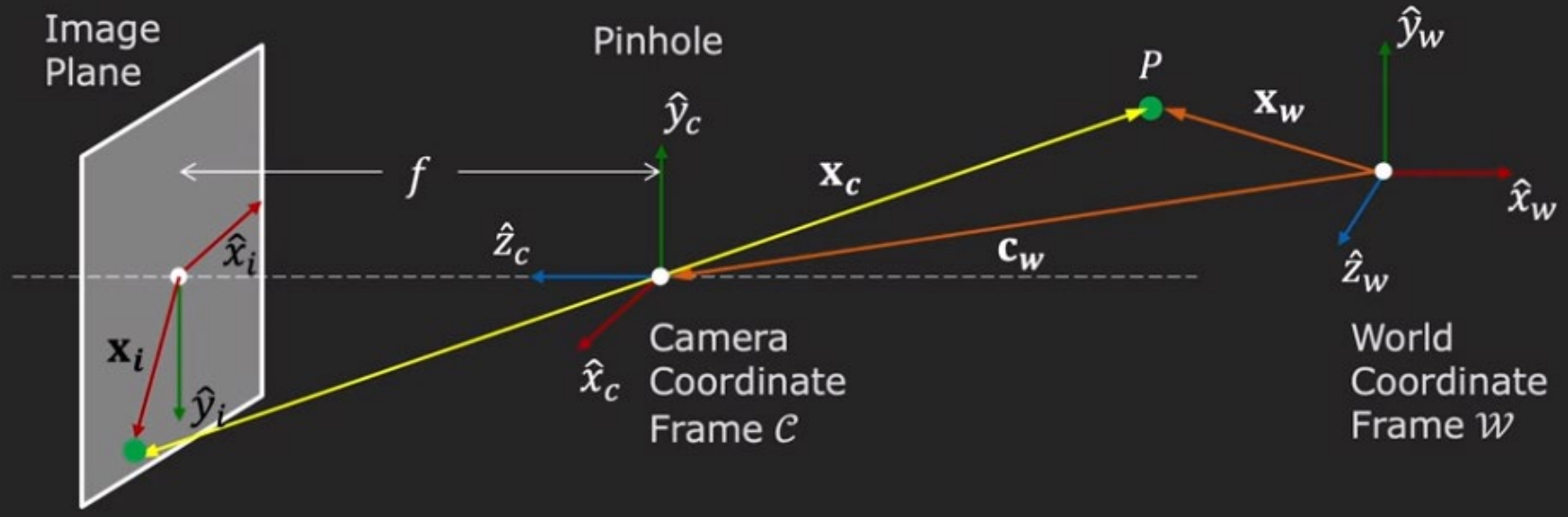


Image Coordinates

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

Perspective Projection

Camera Coordinates

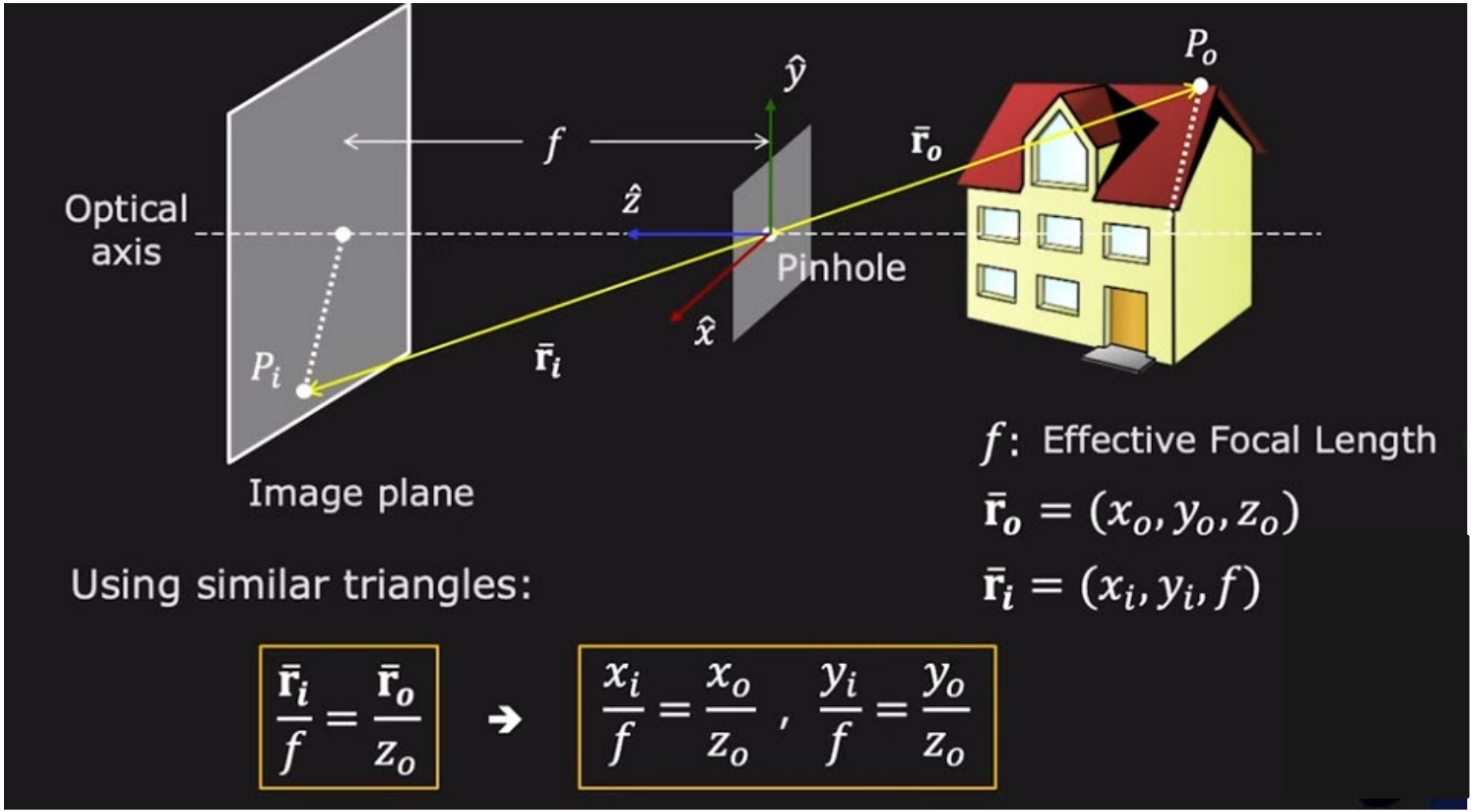
$$\mathbf{x}_c = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$$

Coordinate Transformation

World Coordinates

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

Perspective Imaging with Pinhole



Zo is the depth of the point Po in 3D

Perspective Projection

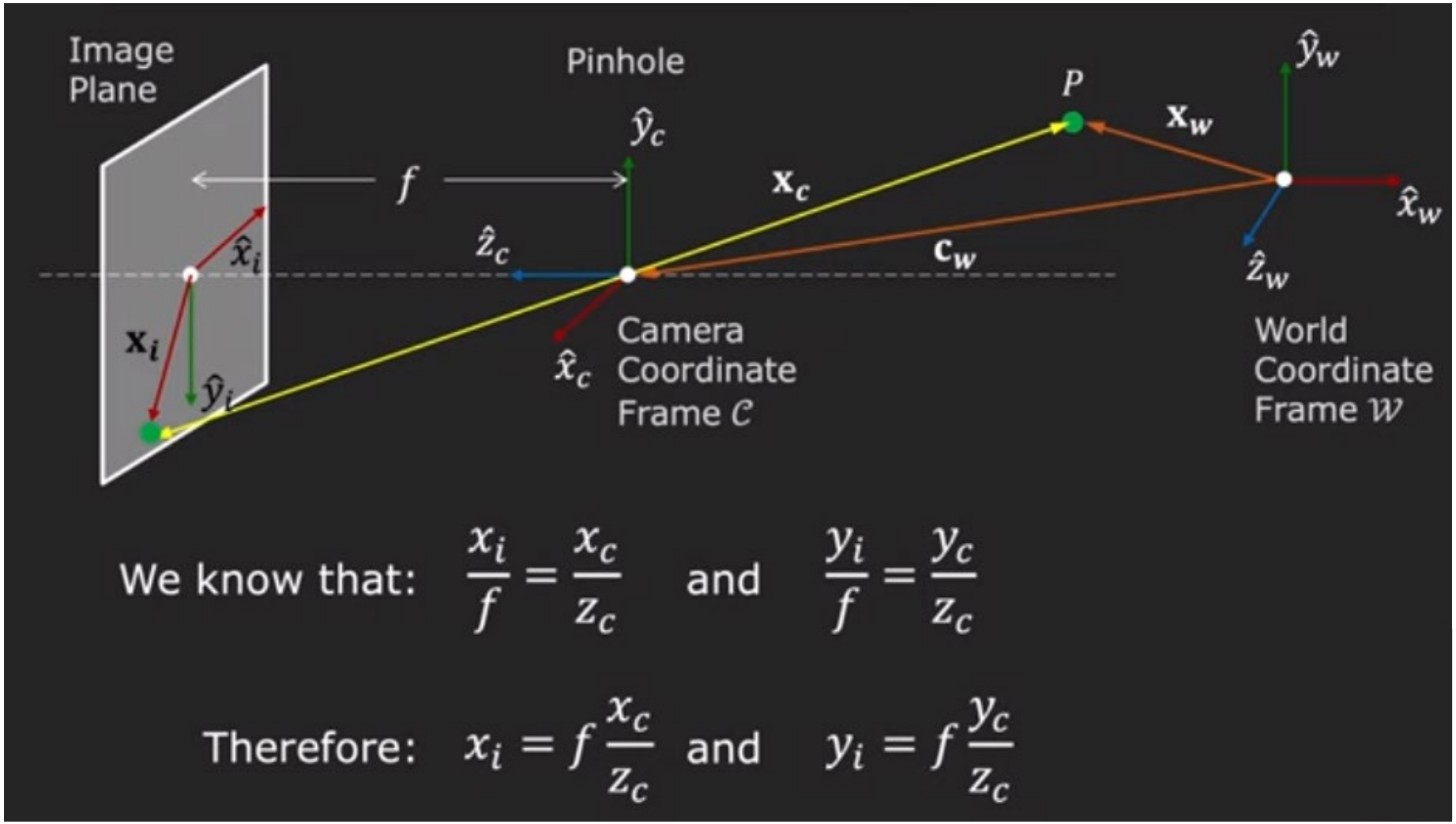
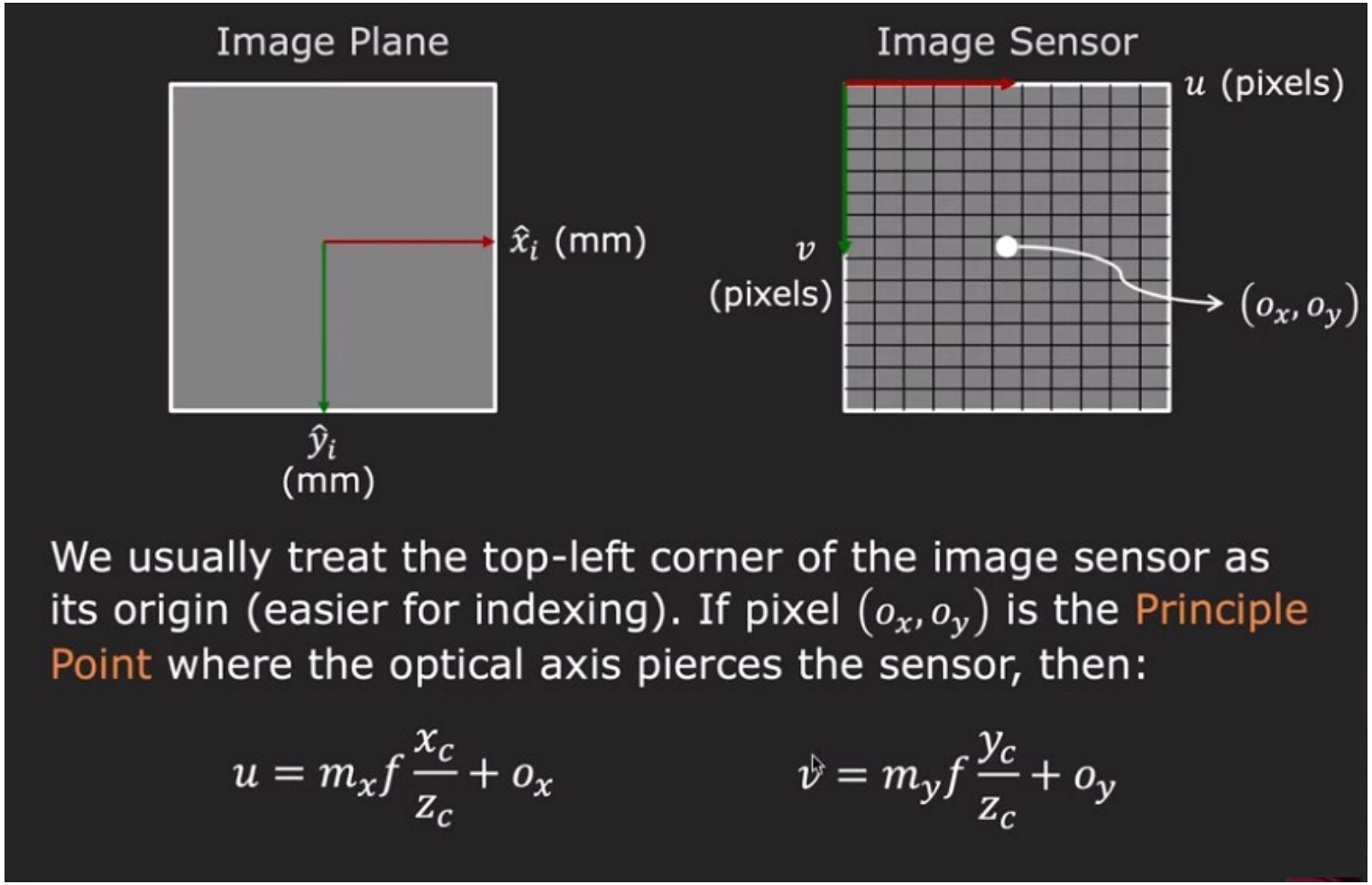


Image Plane to Image Sensor mapping

- mx and my: pixel density (the number of pixels per mm), unknown part of the calibration process



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.

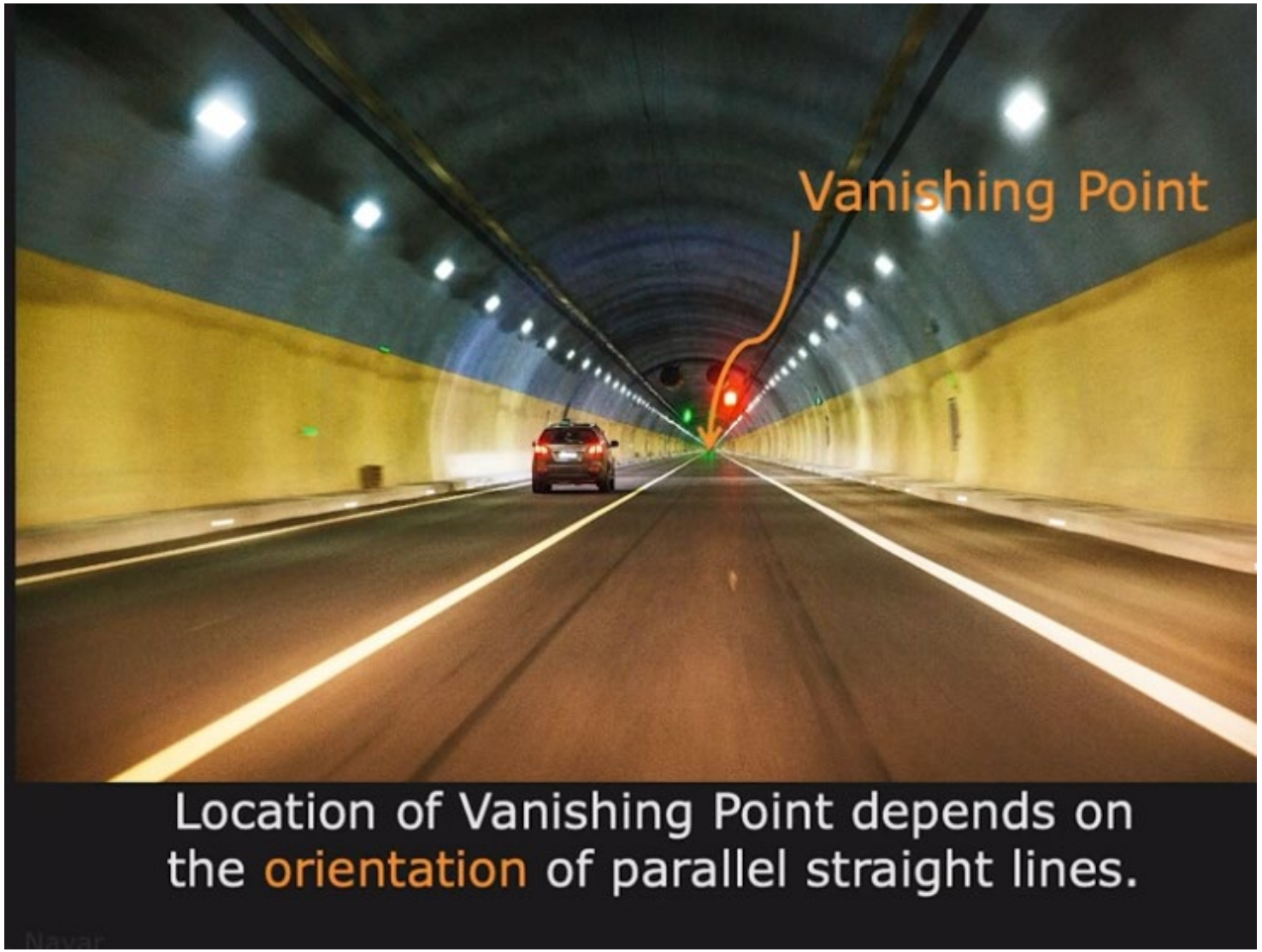
(f_x, f_y, o_x, o_y): Intrinsic parameters of the camera.
They represent the camera's internal geometry.

Image Magnification

- Object size inversely proportional to depth

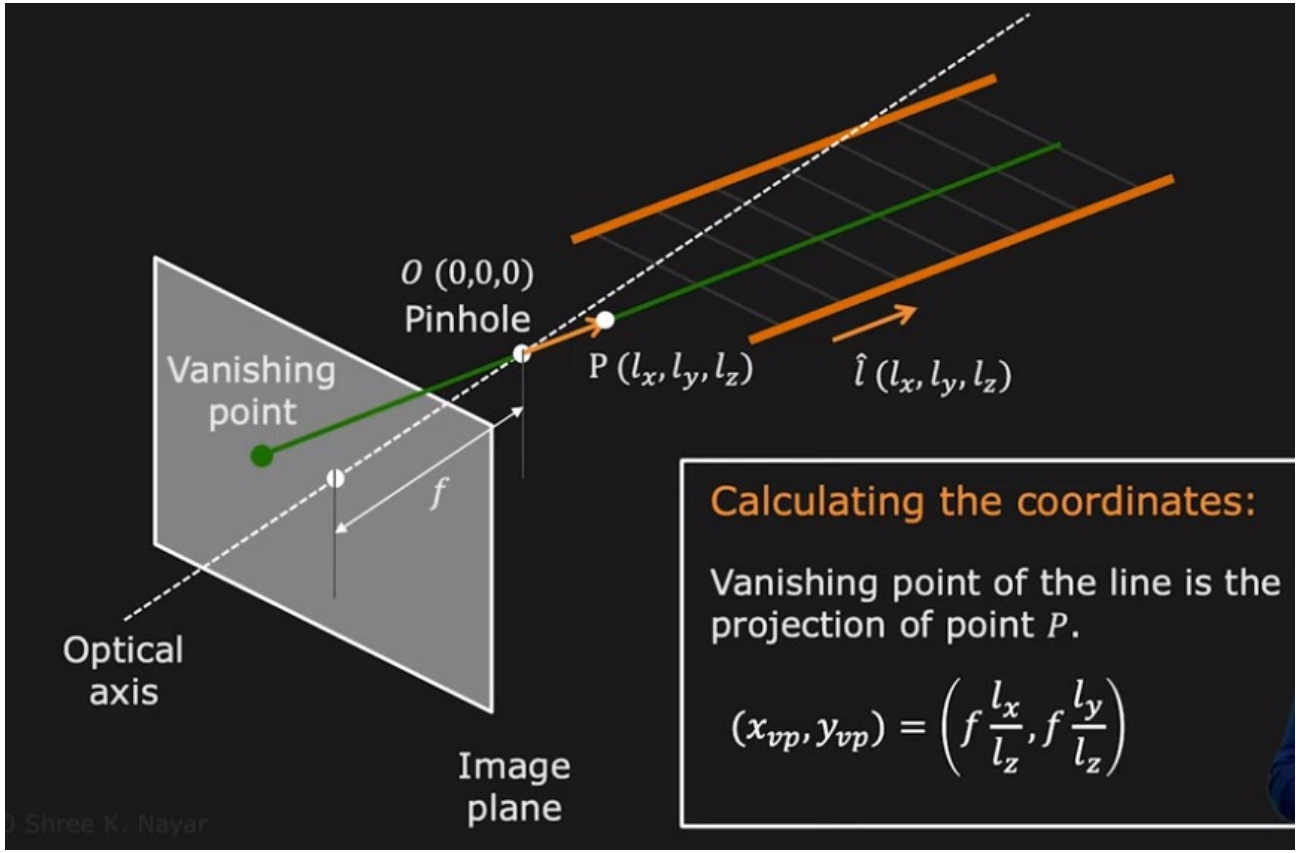
$$M = f/z_o$$

Vanishing Point

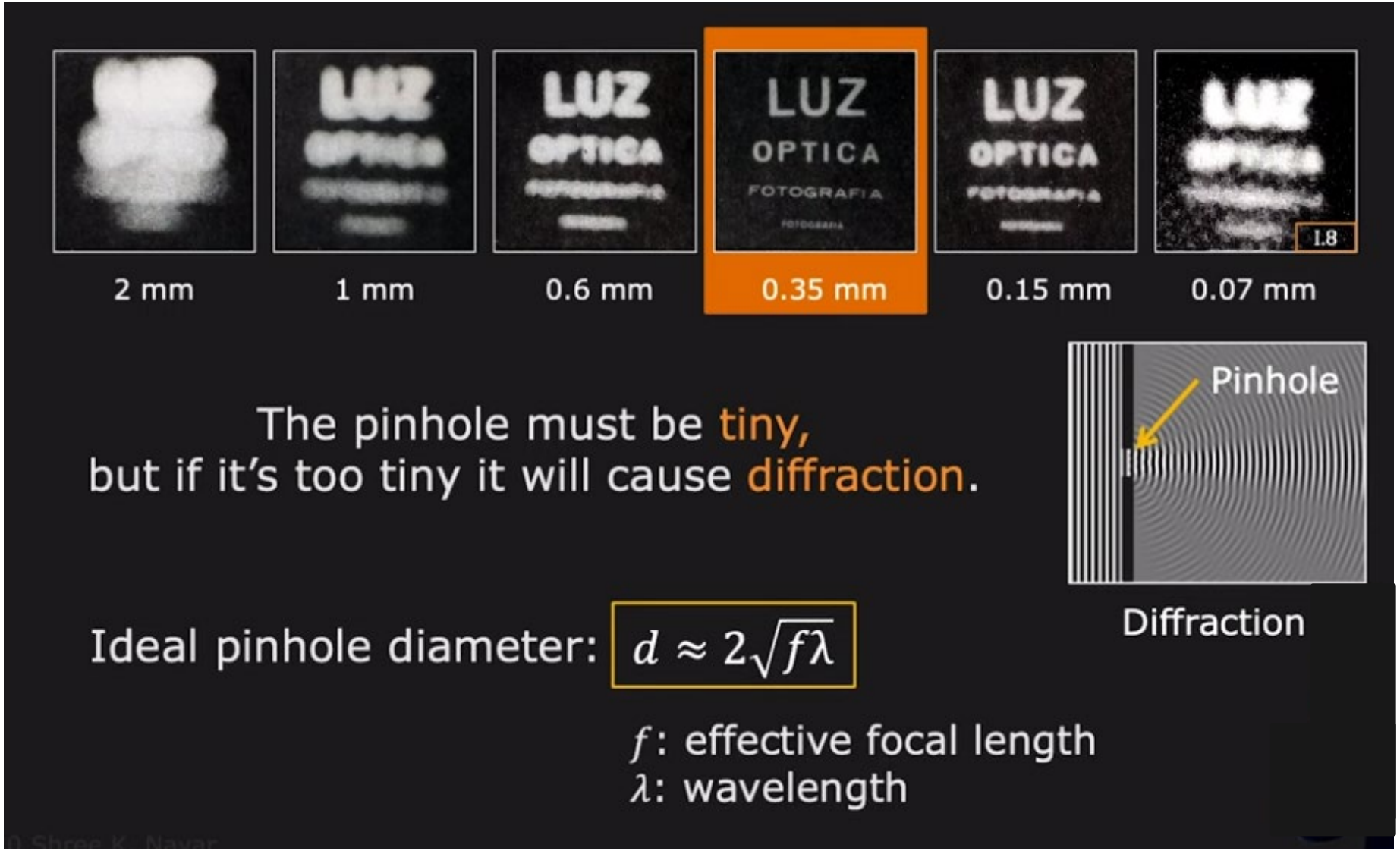


Finding vanishing point

1. Detect Lines in the Image (parallel in the 3D space)
2. Represent Lines in Homogeneous Coordinates
3. Find the Intersection of Lines between the cross product



What is the ideal pinhole size?

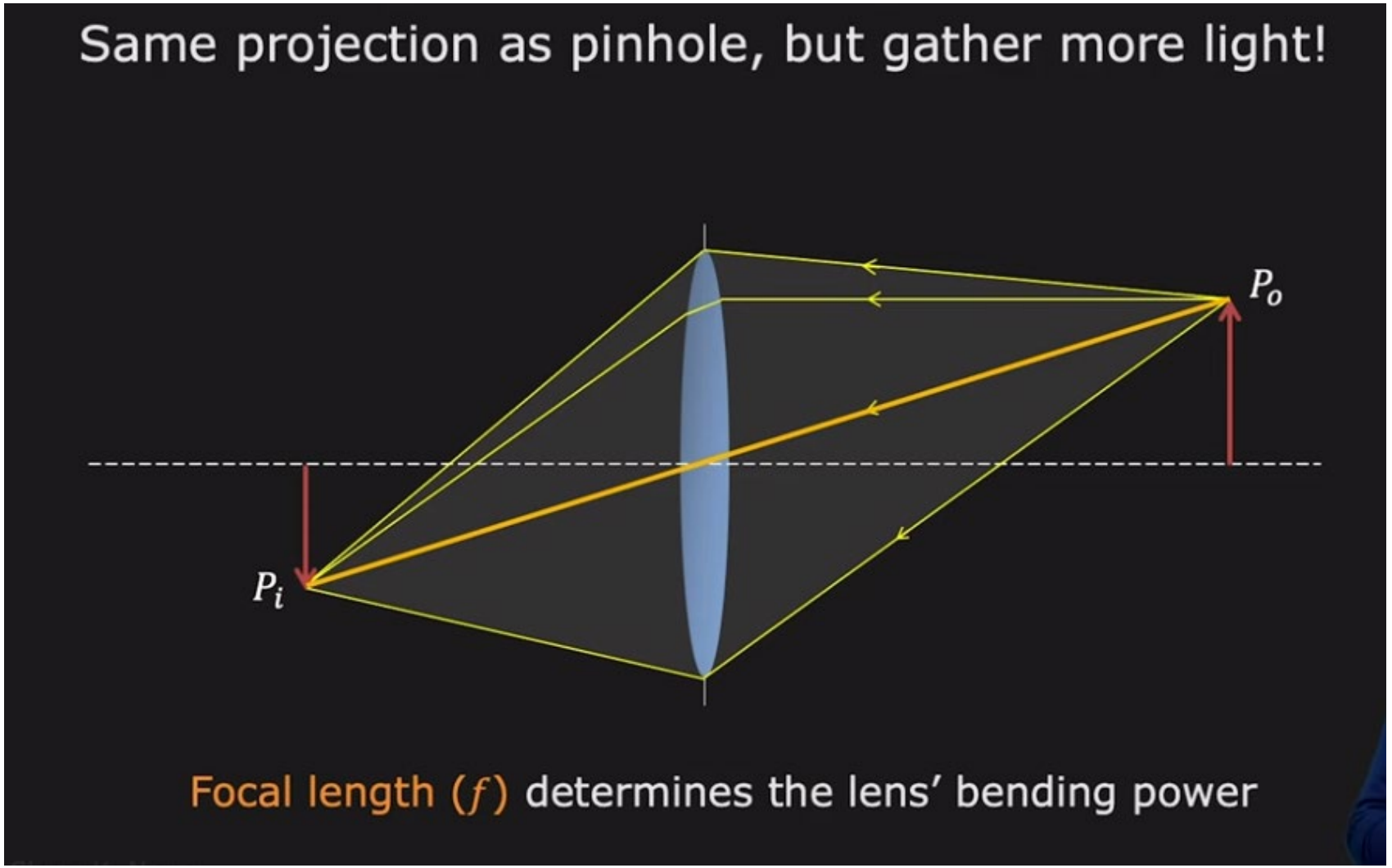


What about exposure time?

Pinholes pass less light and hence require long exposures to capture bright images.

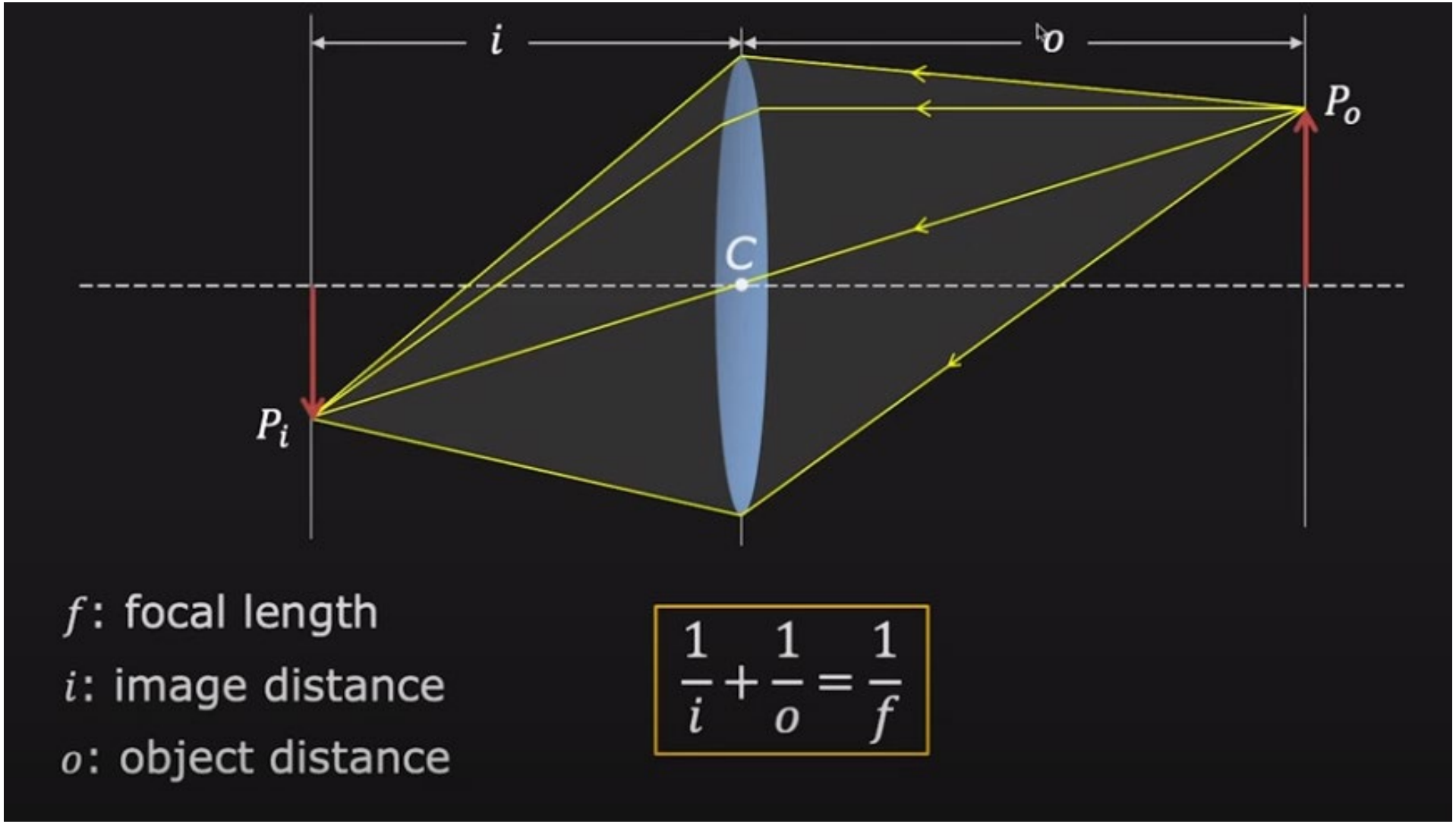
$f = 73\,mm,\ d = 0.2\,mm,$
Exposure, $T = 12\,s$

Lenses



The **focal length** f is the distance from the **lens** to the **focal point**, where **parallel rays of light converge** (for a converging lens). Depends on lens curvature and material (reflective index)

Gaussian Lens (Thin Lens) Law



Perspective Projection

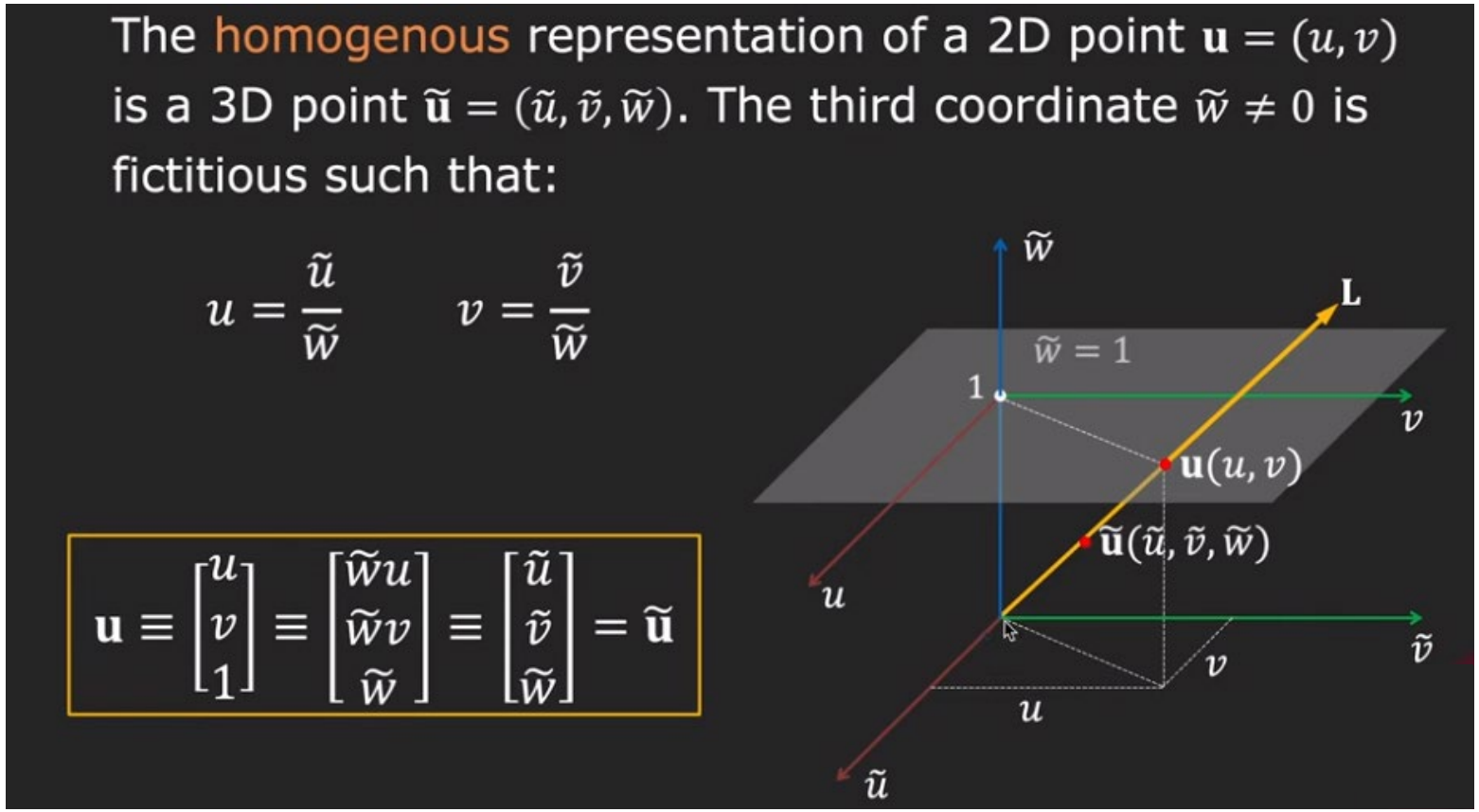
$$u = m_x f \frac{x_c}{z_c} + o_x \qquad v = m_y f \frac{y_c}{z_c} + o_y$$

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

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

Equations for perspective projections are non linear
It is convenient to express them as linear equations
With homogeneous coordinates

Homogeneous Coordinates



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

Homogeneous 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

- Linear model for perspective projection

Perspective projection equations:

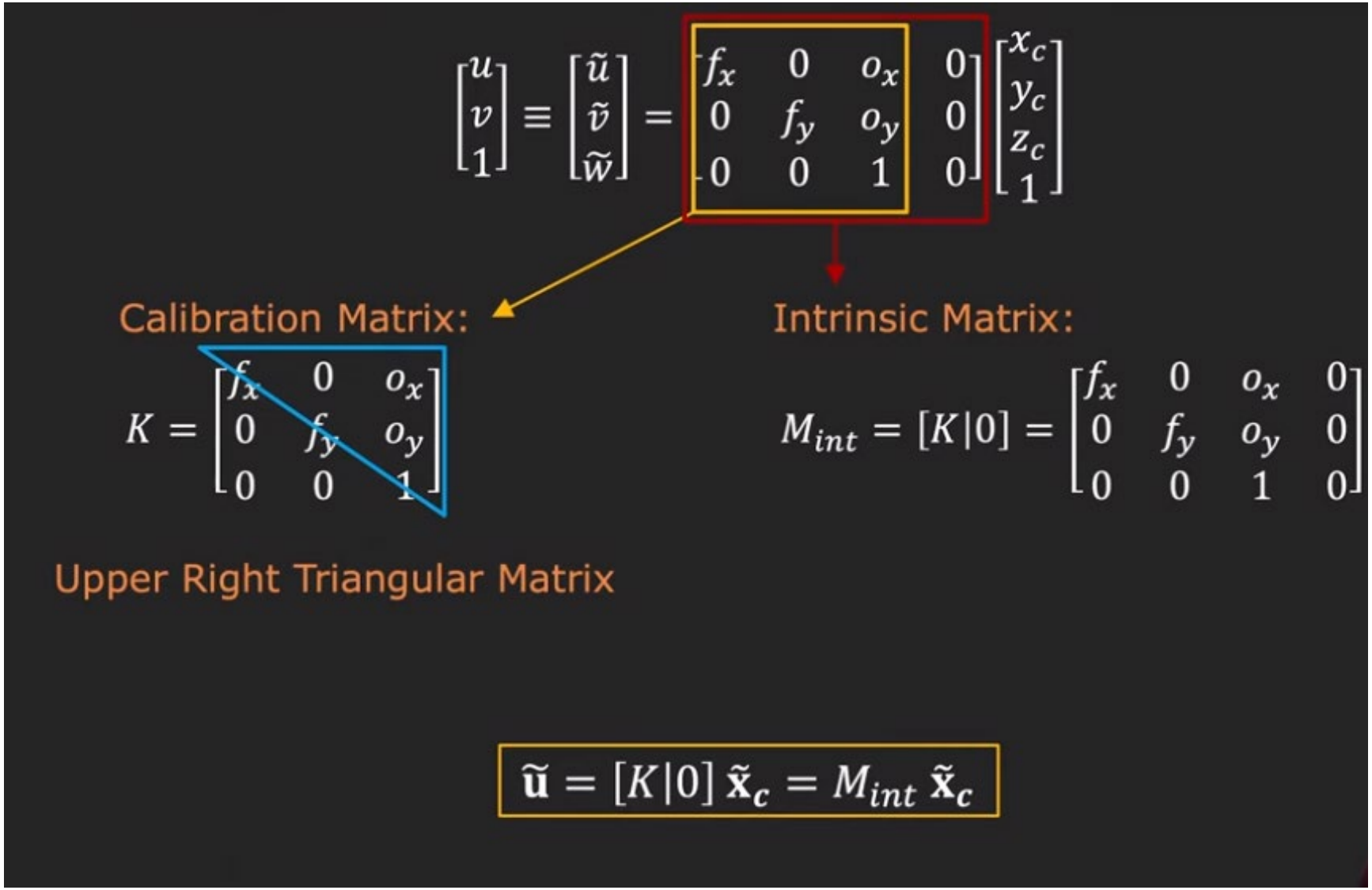
$$u = f_x \frac{x_c}{z_c} + o_x \qquad v = f_y \frac{y_c}{z_c} + o_y$$

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

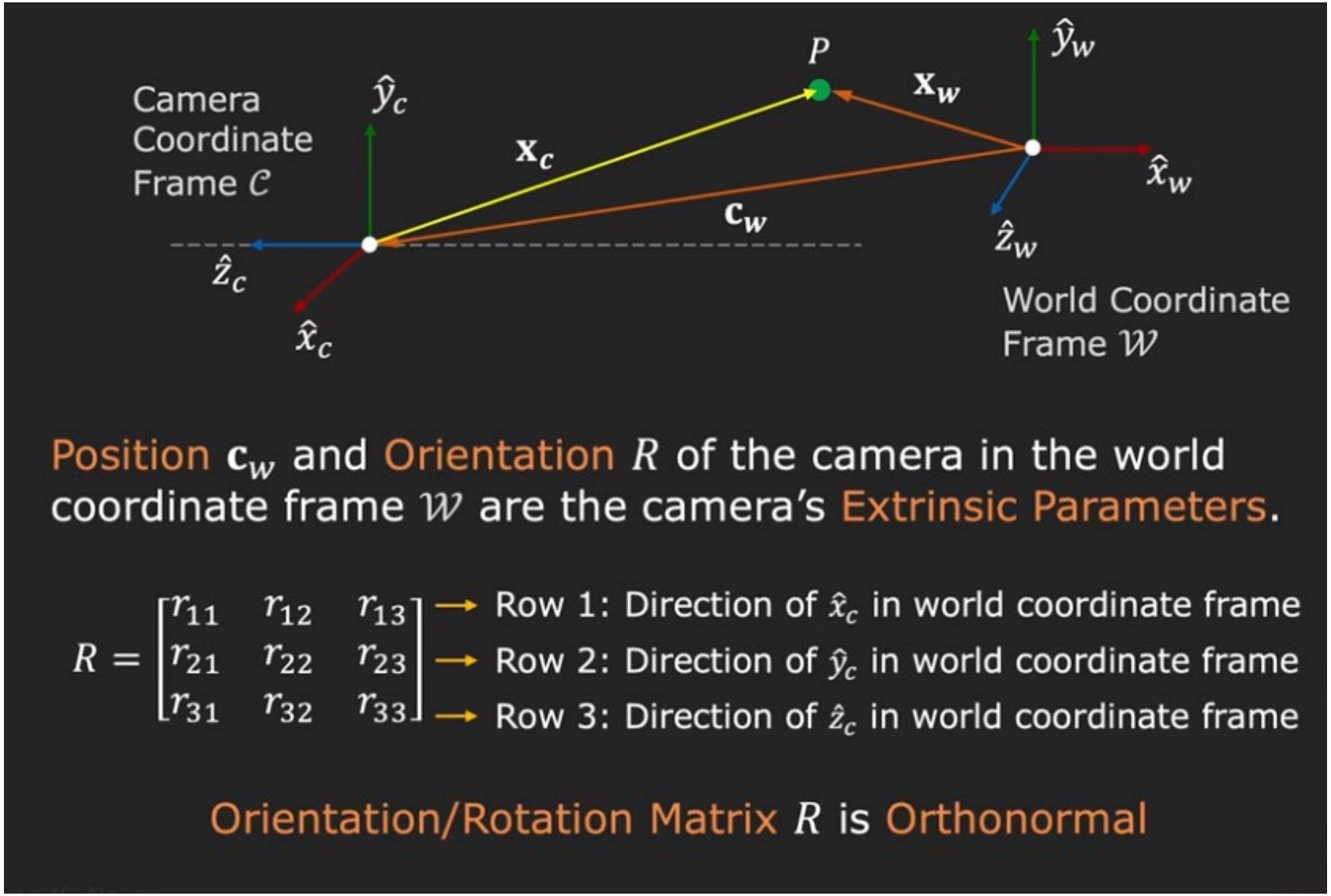
Intrinsic Matrix



f: focal lengths in pixels

o: coordinates of the principal point

Extrinsic Parameters



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)

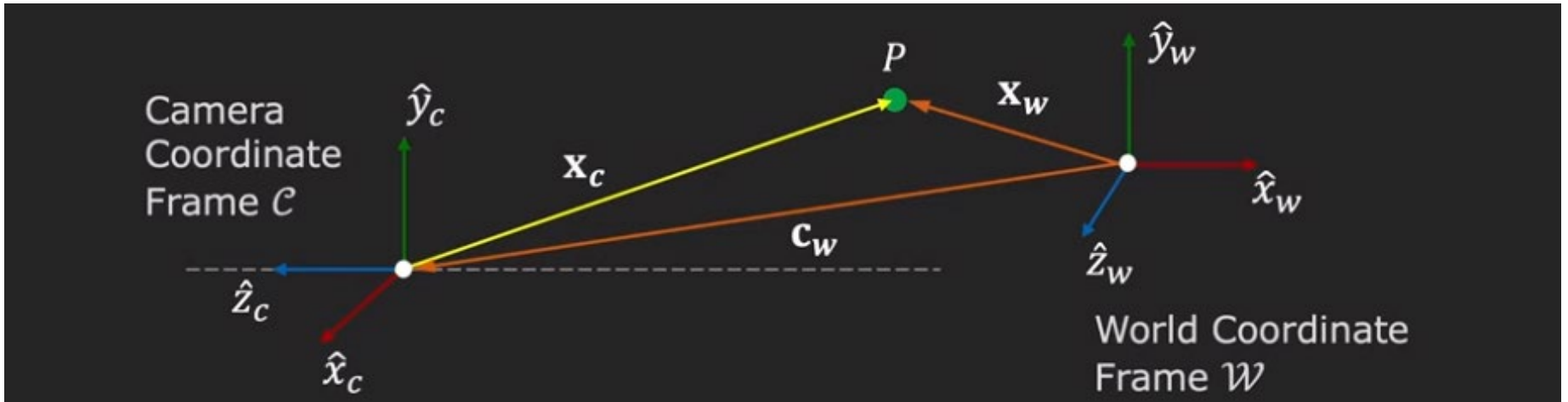
Example: The x-, y- and z-axes of $\mathbb{R}^3$ Euclidean space

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

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

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:

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

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

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

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

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

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

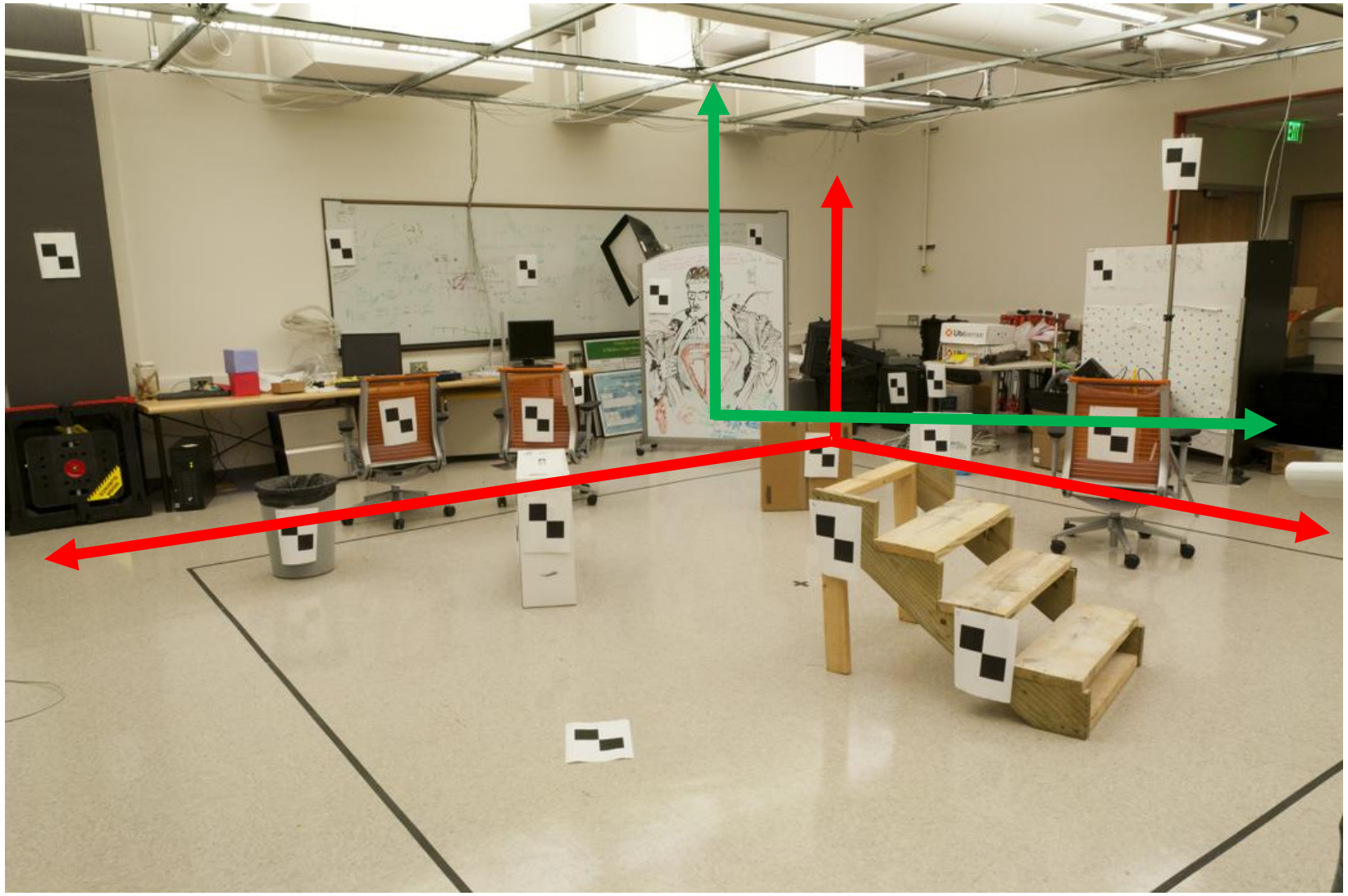
$$\tilde{\mathbf{x}}_c = M_{ext} \tilde{\mathbf{x}}_w$$

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

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

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

World vs Image Coordinates

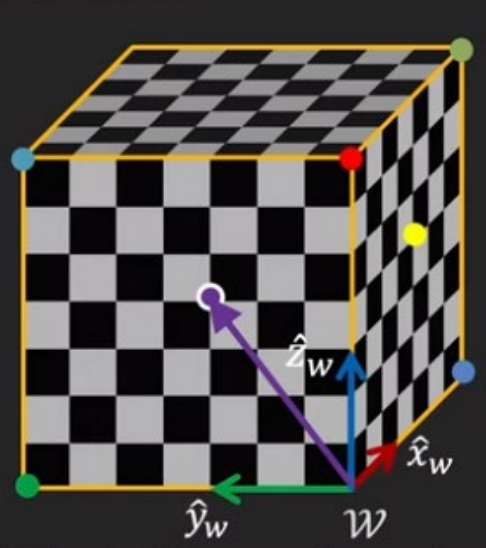


James Hays

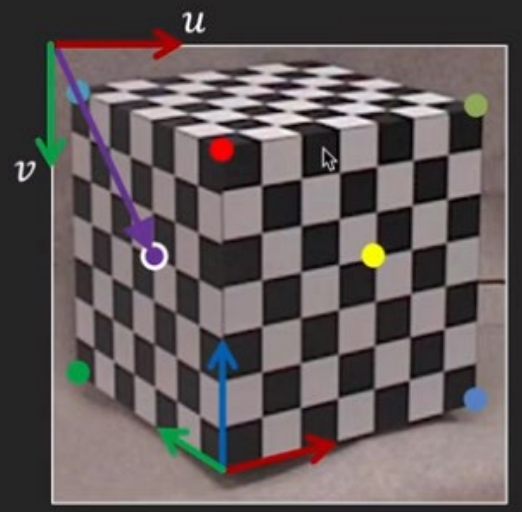
Camera Calibration Procedure

Step 1: Capture an image of an object of known geometry

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

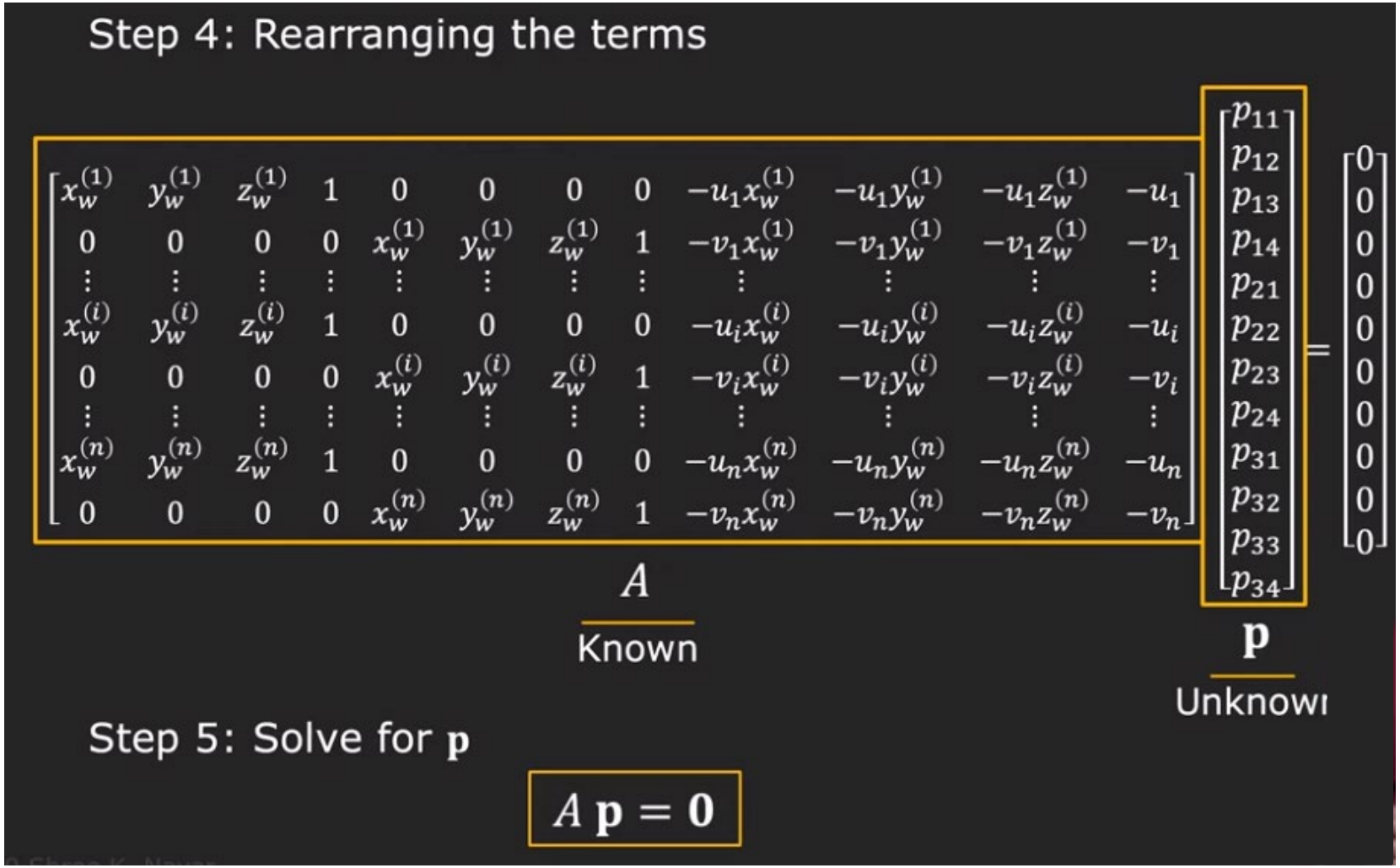
$$\underbrace{\begin{bmatrix} u^{(i)} \\ v^{(i)} \\ 1 \end{bmatrix}}_{\text{Known}} \equiv \underbrace{\begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}}_{\text{Unknown}} \underbrace{\begin{bmatrix} x_w^{(i)} \\ y_w^{(i)}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

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



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.

Least Square Solution for matrix P

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

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

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 Square 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.

Extracting Intrinsic/Extrinsic Parameters

- Assuming you've already obtained P

We know that:

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

M_{int} M_{ext}

That is:

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

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

Extracting Intrinsic/Extrinsic Parameters

We know that:

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

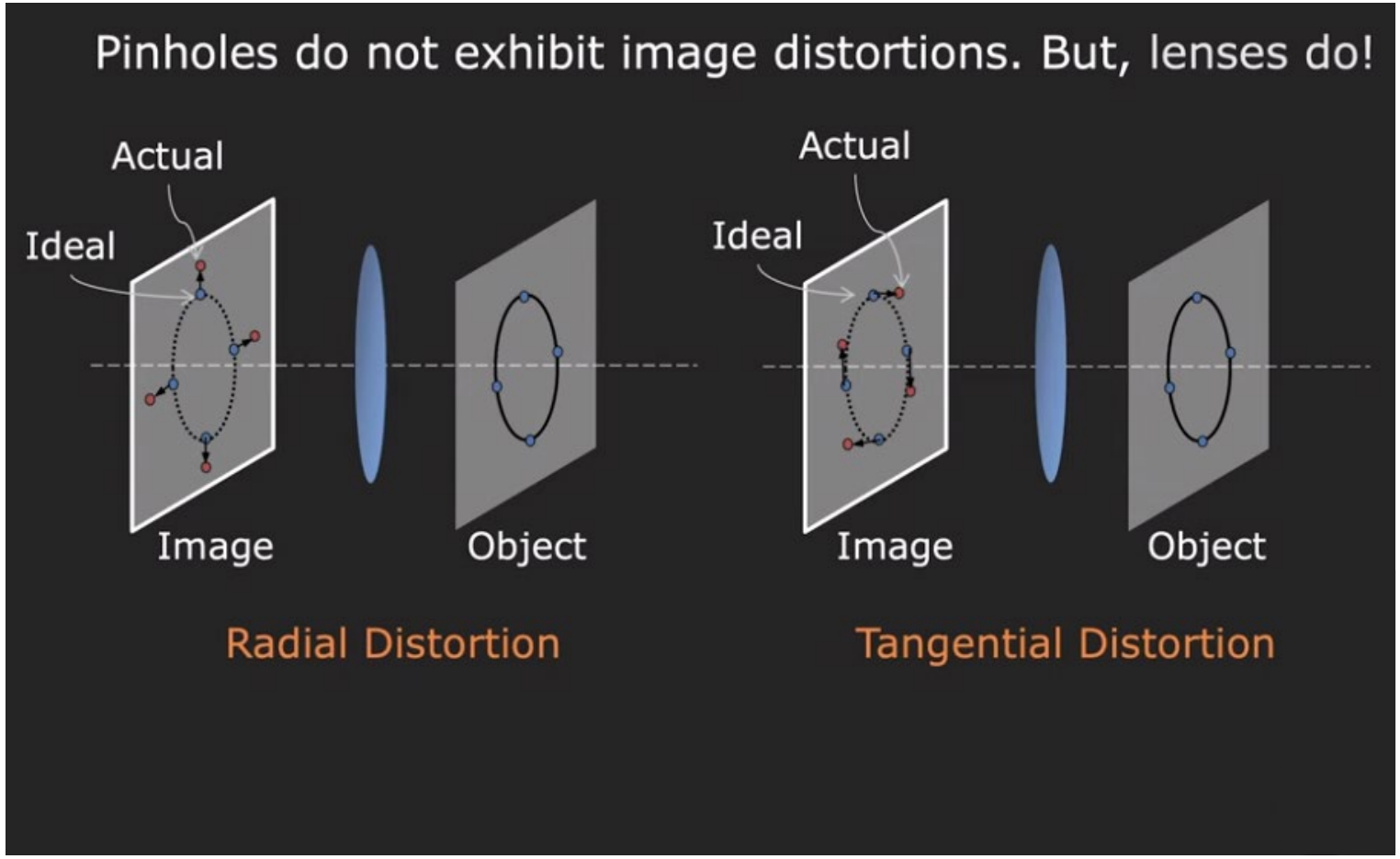
That is:

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

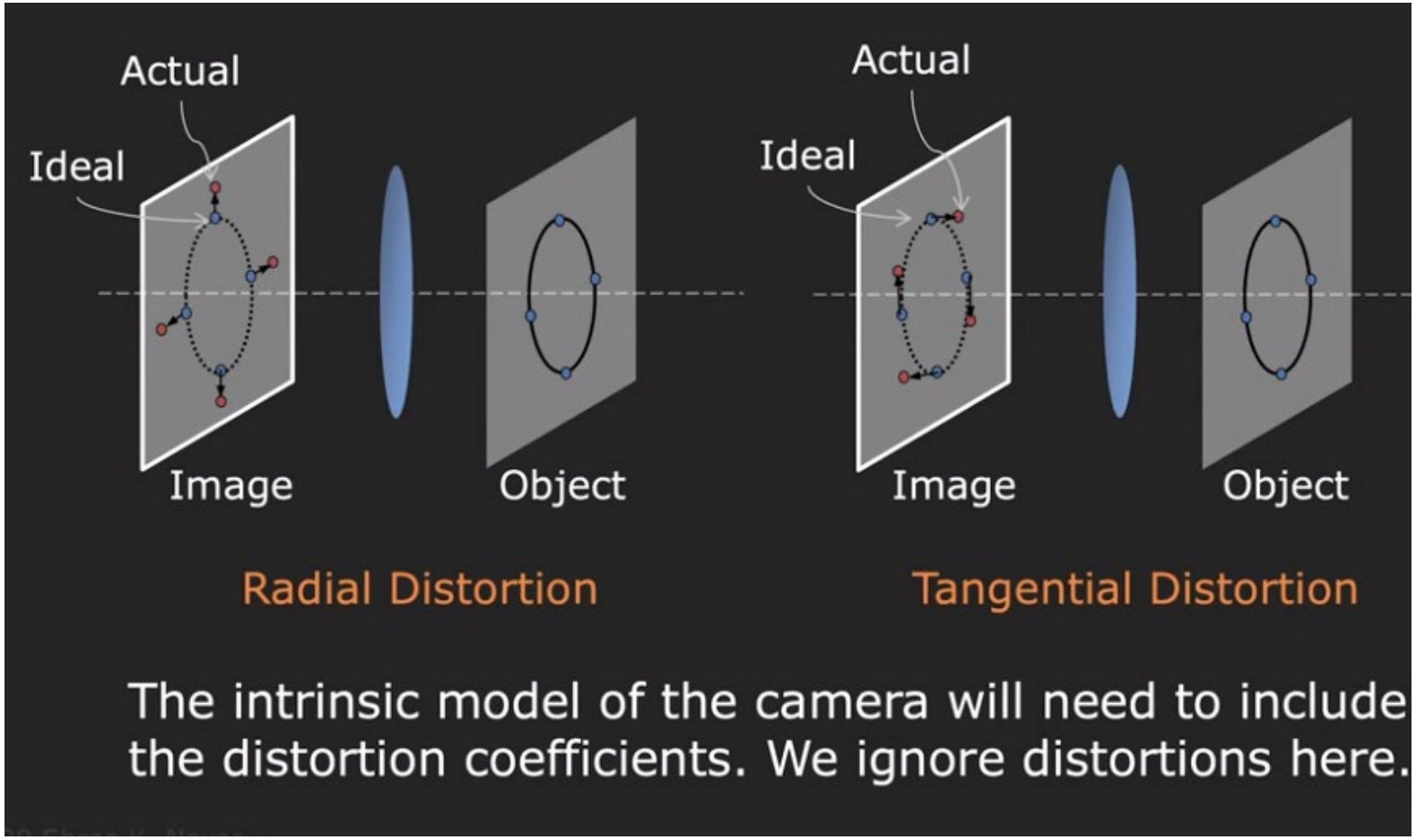
Therefore:

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

Other Intrinsic Parameters



Other Intrinsic Parameters



Lens Distortion

The assumption of linear projection (straight lines remain straight) is violated in practice due to the properties of the camera lens which introduces distortions.
Both **radial and tangential distortion** effects can be modeled relatively easily:
Let $x = x_c/z_c$, $y = y_c/z_c$ and $r^2 = x^2 + y^2$. The distorted point is obtained as:

$$\mathbf{x}' = \underbrace{(1 + \kappa_1 r^2 + \kappa_2 r^4)}_{\text{Radial Distortion}} \begin{pmatrix} x \\ y \end{pmatrix} + \underbrace{\begin{pmatrix} 2\kappa_3 x y + \kappa_4(r^2 + 2x^2) \\ 2\kappa_4 x y + \kappa_3(r^2 + 2y^2) \end{pmatrix}}_{\text{Tangential Distortion}}$$

$$\mathbf{x}_s = \begin{pmatrix} f_x x' + c_x \\ f_y y' + c_y \end{pmatrix}$$

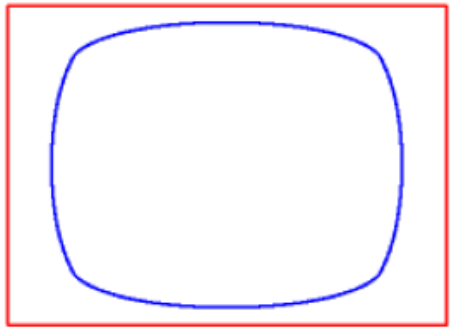
Images can be **undistorted** such that the perspective projection model applies.
More complex distortion models must be used for wide-angle lenses (e.g., fisheye).

Lens Distortion

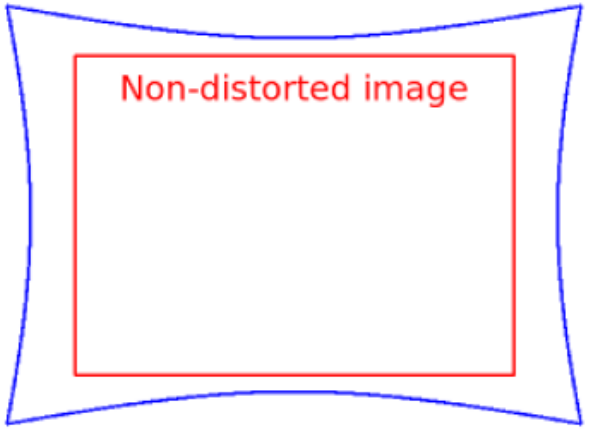
No distortion

Negative radial distortion
(Barrel distortion)

Positive radial distortion
(Pincushion distortion)



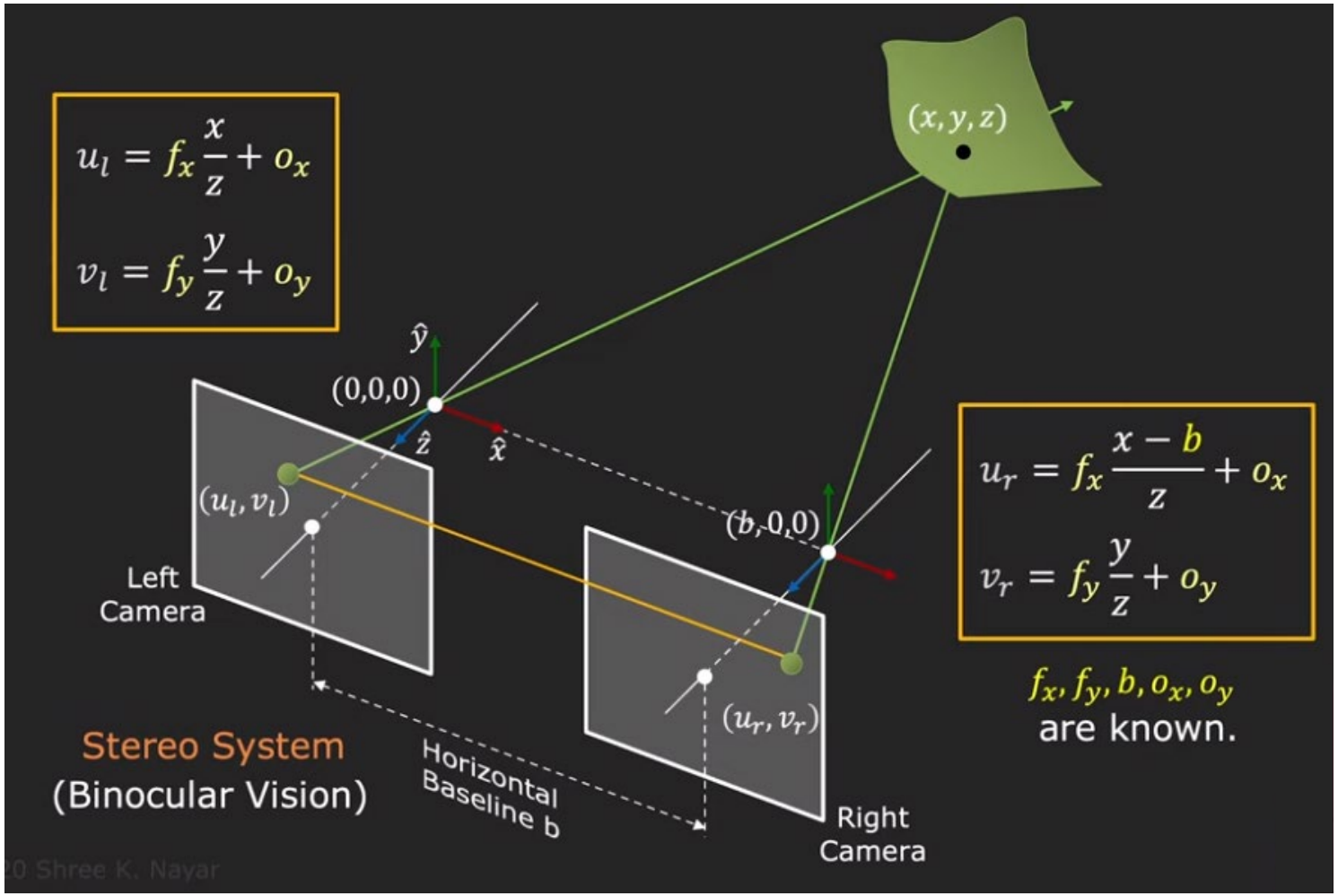
Negative radial distortion (k1=-1.5)
(Barrel distortion)



Positive radial distortion (k1=1.5)
(Pincushion distortion)

Stereo Vision

Triangulation using two cameras



- **Baseline** is the **physical distance** between the **centers of projection** (optical centers) of two cameras in a stereo setup.

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.

- **Disparity** is the **difference in pixel positions** of the same 3D point projected into the **left and right images**.

Stereo Matching: finding disparities

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

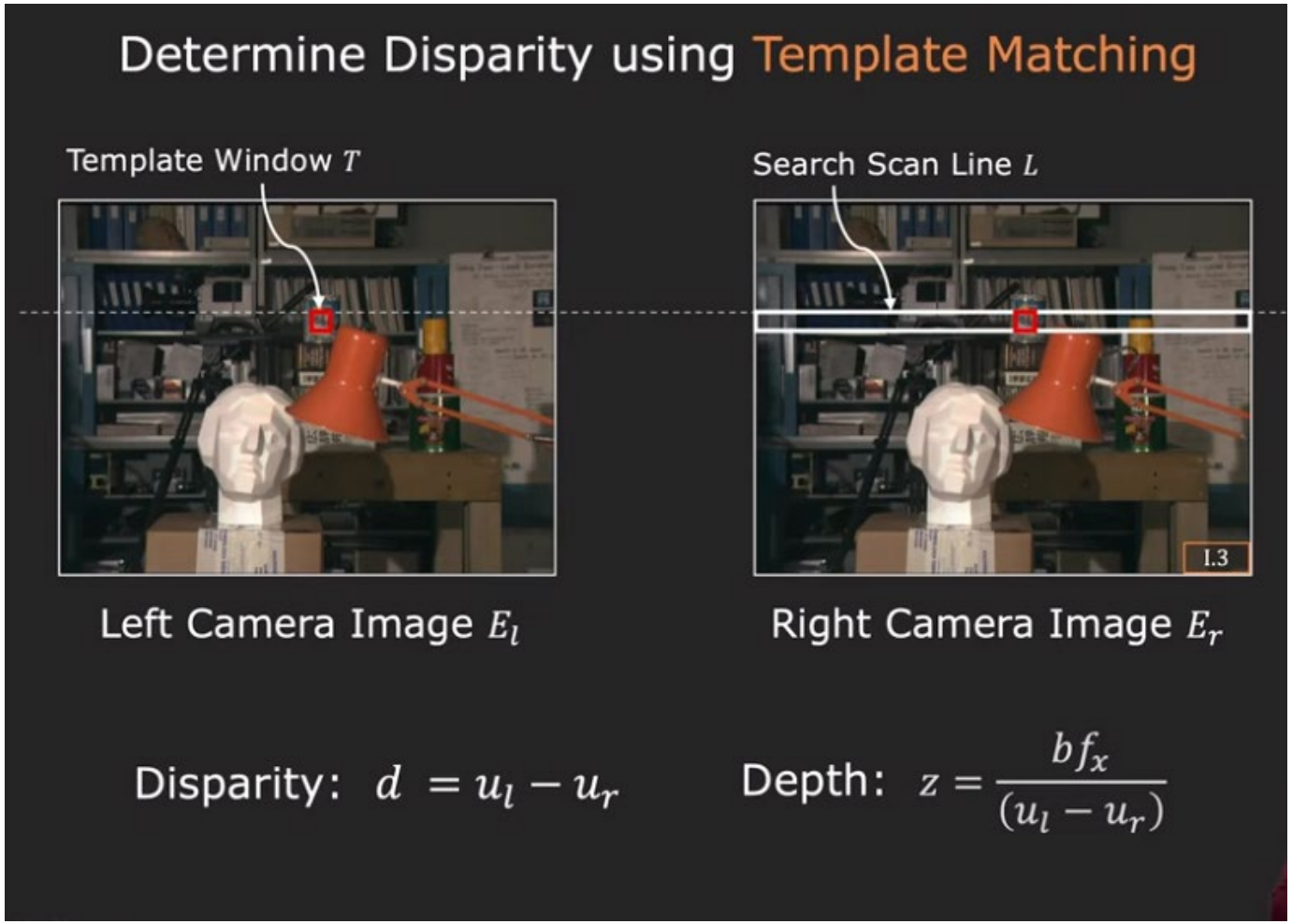
Left/Right Camera Images

Disparity Map (Ground Truth)

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

- Disparity is the horizontal shift between corresponding points in the left and right images
- Corresponding scene points lie on the same horizontal scan line (this assumes images are rectified, so corresponding points lie on the same row)

Window based methods



b: baseline (distance between cameras)
f: focal lenght in pixels
d: diparity in pixels

Similarity matrix 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:

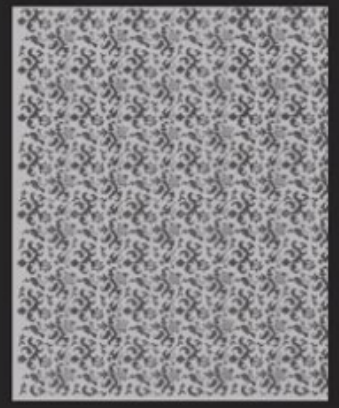
$$SSD(k,l) = \sum_{(i,j)\in T} |E_l(i,j) - E_r(i+k, j+l)|^2$$

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

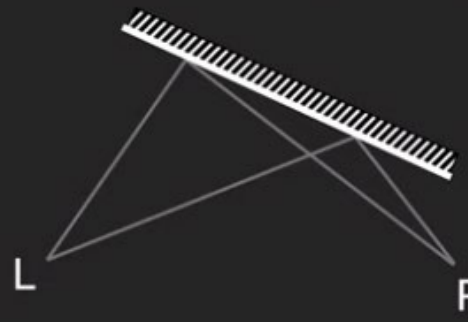
$$NCC(k,l) = \frac{\sum_{(i,j)\in T} E_l(i,j)\, E_r(i+k, j+l)}{\sqrt{\sum_{(i,j)\in T} E_l(i,j)^2 \sum_{(i,j)\in T} E_r(i+k, j+l)^2}}$$

Issues with Stereo Matching

- Surface must have (non-repetitive) texture

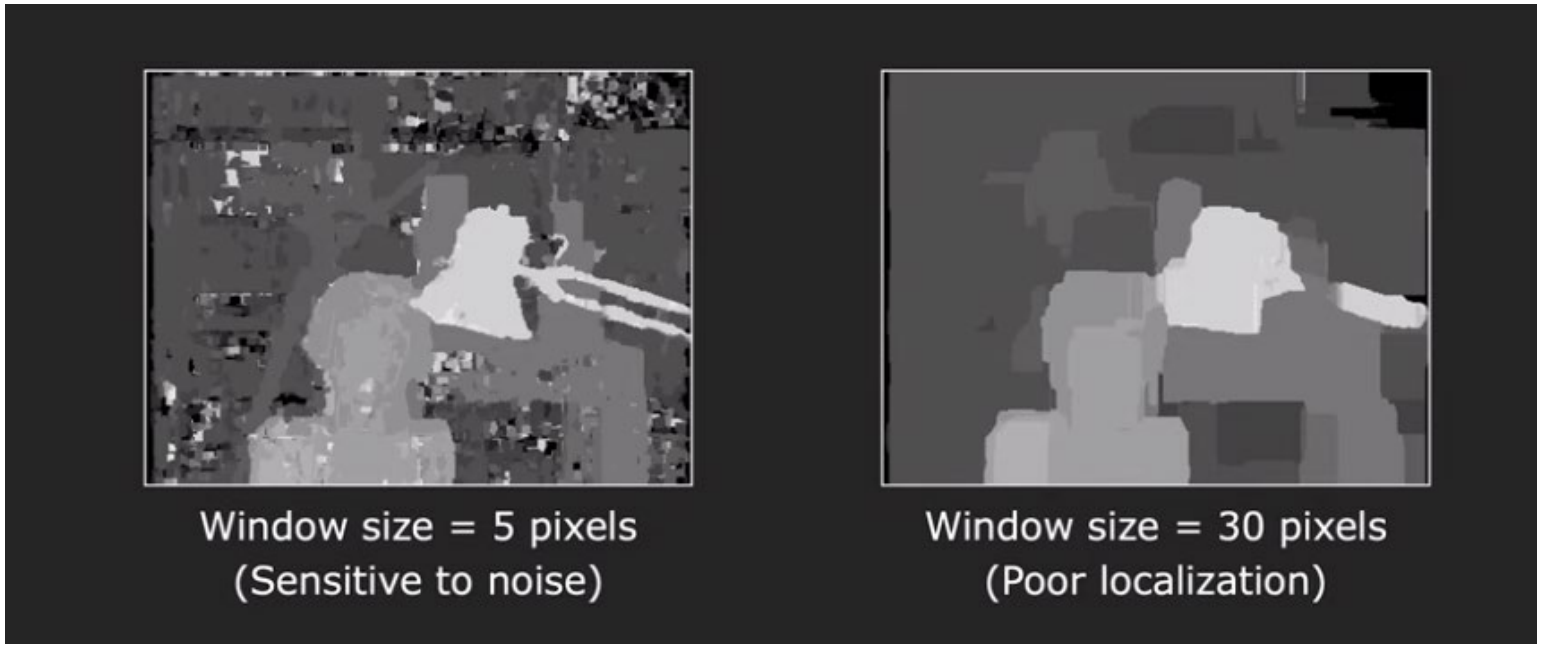


- Foreshortening effect makes matching challenging

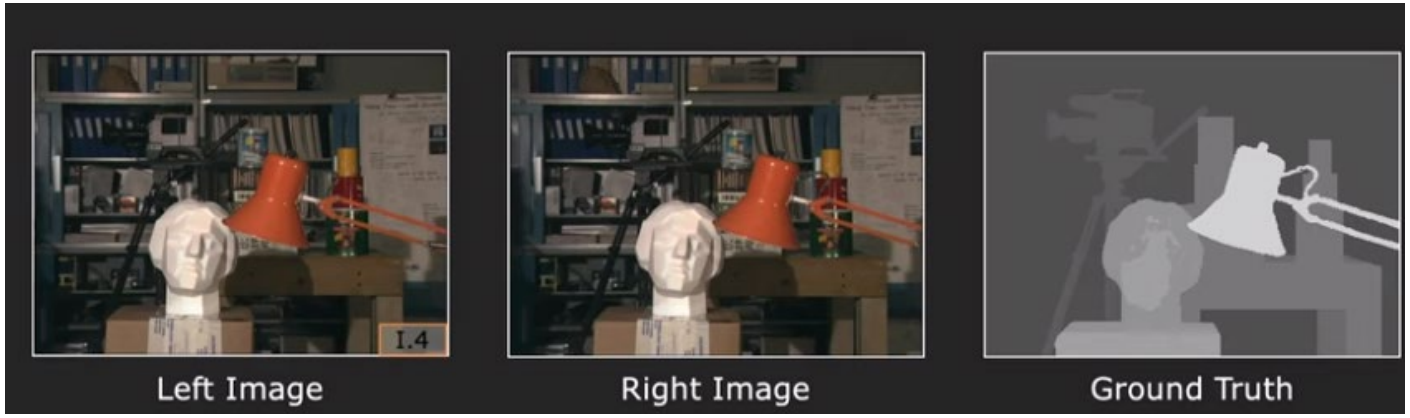


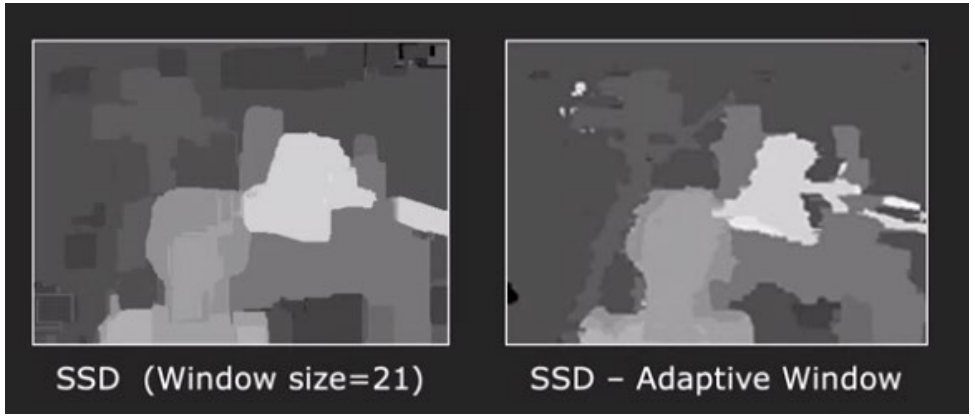
How large should window be?

- 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 method: results

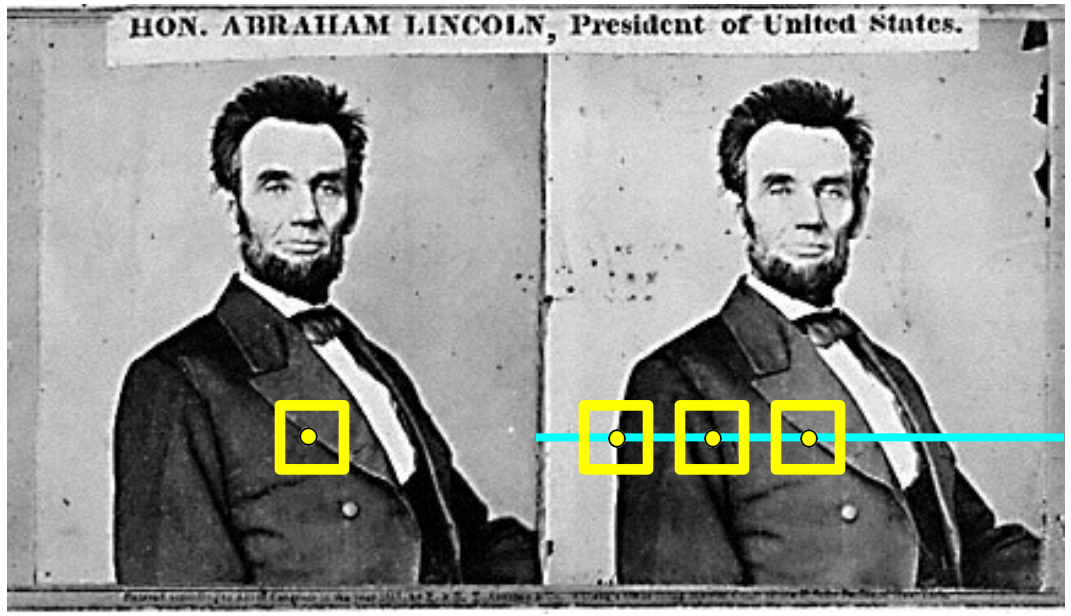




https://vision.middlebury.edu/stereo/

Figure 17: *Comparative results on the Tsukuba images. The results are shown in decreasing order of overall performance ($B_{\overline{\sigma}}$). Algorithms implemented by us are marked with a star.*

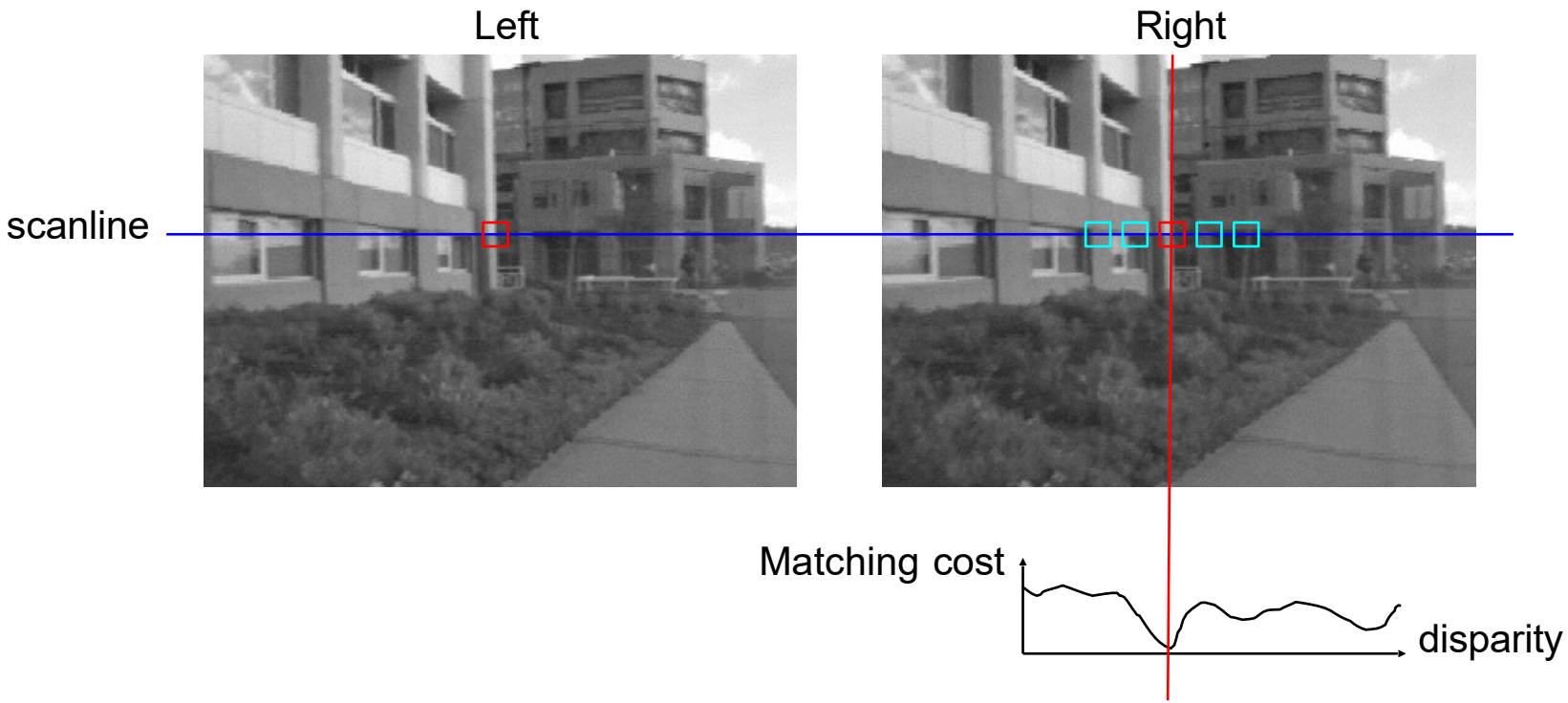
D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. International Journal of Computer Vision, 47(1/2/3):7-42, April-June 2002.



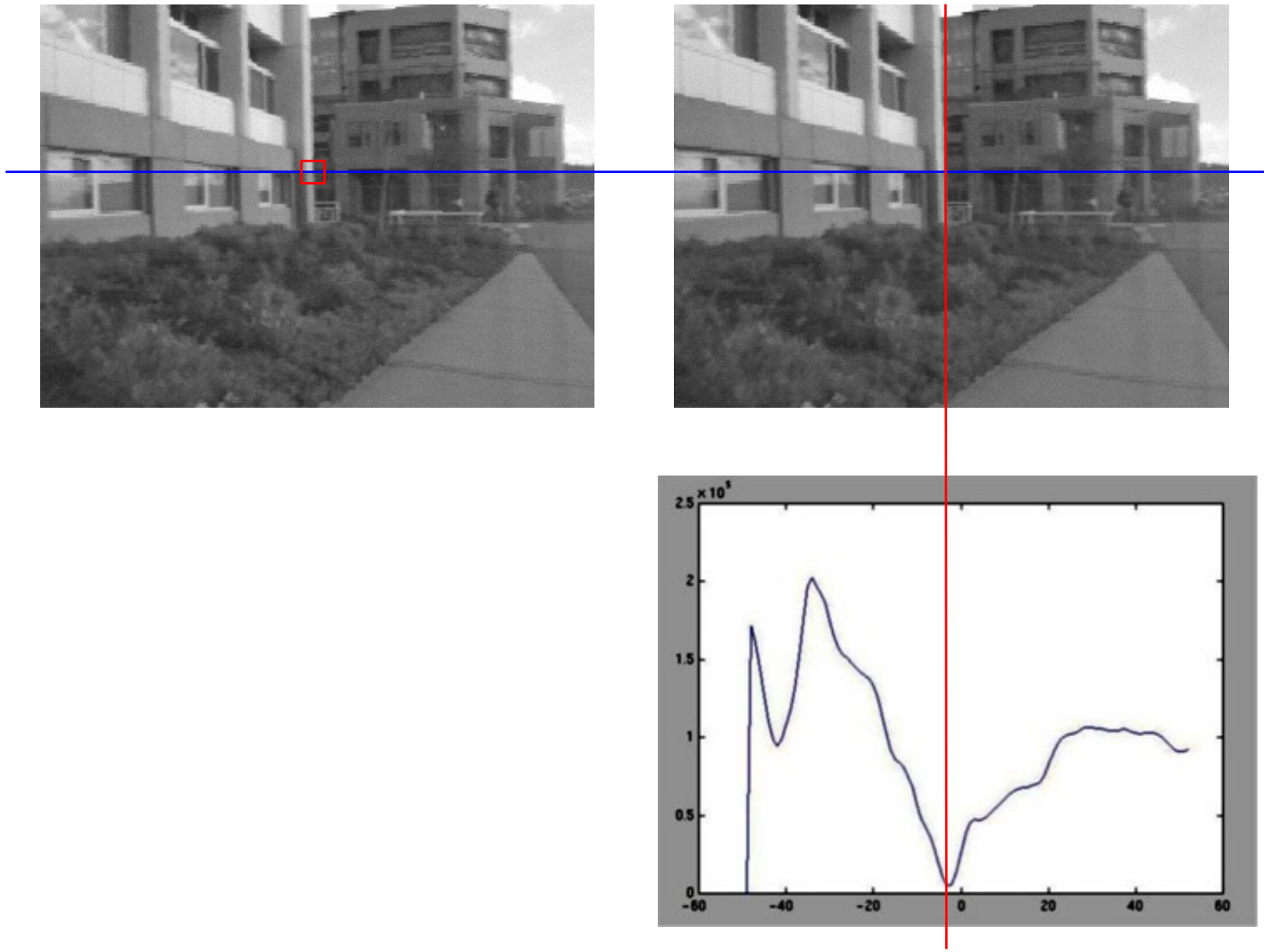
1. Rectify images
   (make epipolar lines horizontal)
2. For each pixel
   a. Find epipolar line
   b. Scan line for best match
   c. Compute depth from disparity

$$Z = \frac{bf}{d}$$

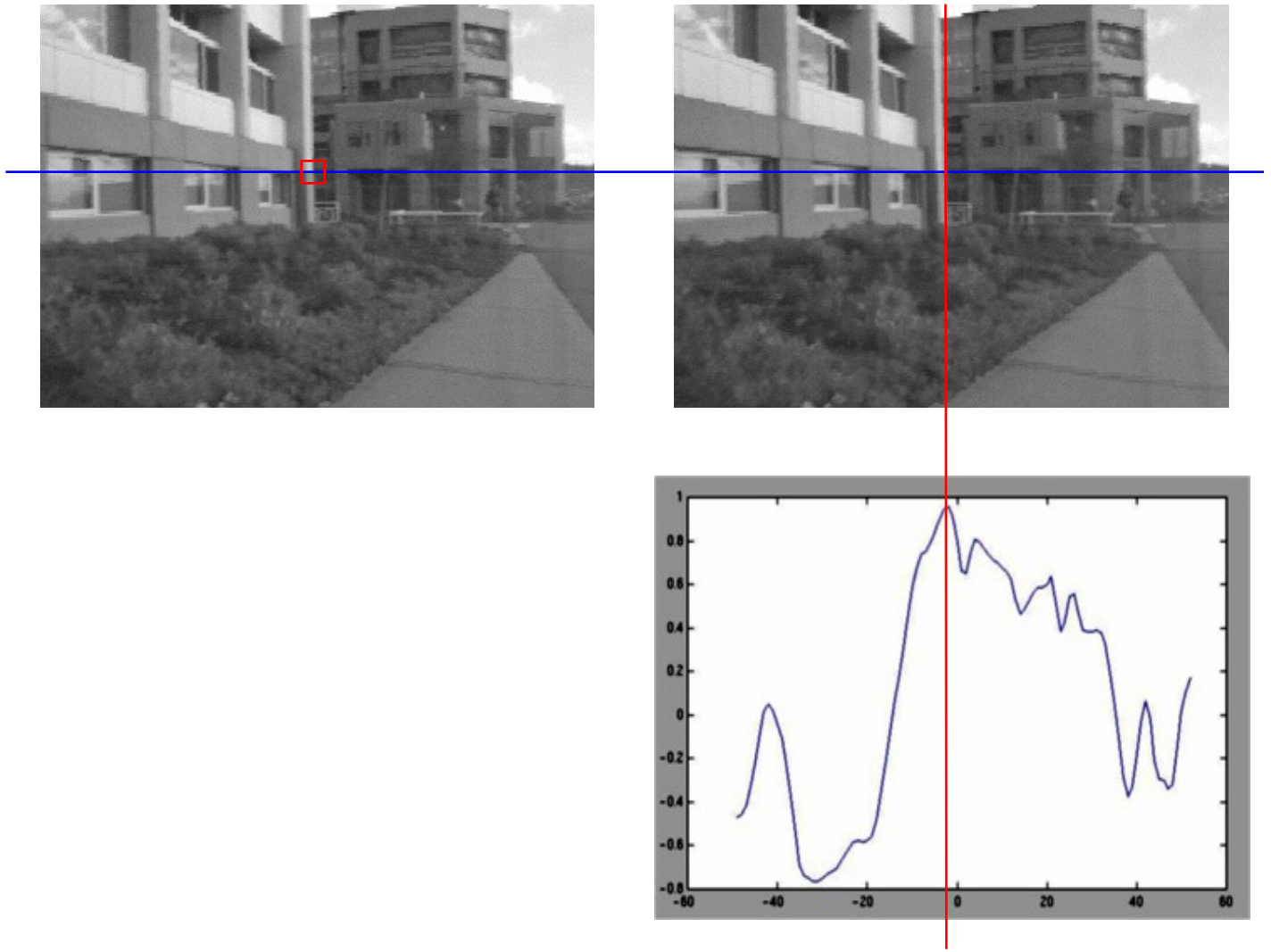
Stereo Block Matching



- Slide a window along the epipolar line and compare contents of that window with the reference window in the left image
- Matching cost: SSD or normalized correlation

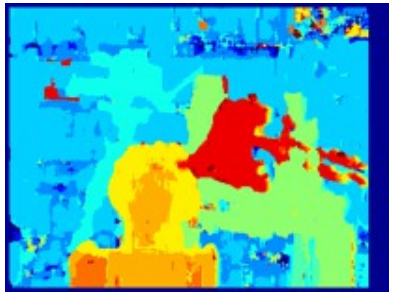


SSD

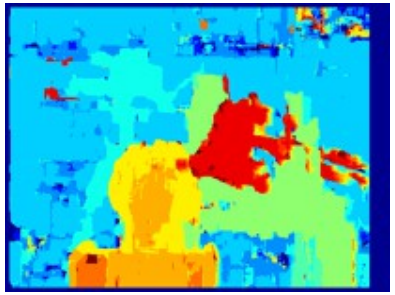


Normalized cross-correlation

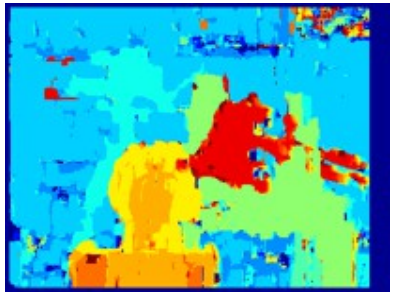
Similarity Measure	Formula
Sum of Absolute Differences (SAD)	$\sum_{(i,j)\in W} \lvert I_1(i,j) - I_2(x+i,y+j)\rvert$
Sum of Squared Differences (SSD)	$\sum_{(i,j)\in W} \left(I_1(i,j) - I_2(x+i,y+j)\right)^2$
Zero-mean SAD	$\sum_{(i,j)\in W} \lvert I_1(i,j) - \bar{I}_1(i,j) - I_2(x+i,y+j) + \bar{I}_2(x+i,y+j)\rvert$
Locally scaled SAD	$\sum_{(i,j)\in W} \lvert I_1(i,j) - \frac{\bar{I}_1(i,j)}{\bar{I}_2(x+i,y+j)} I_2(x+i,y+j)\rvert$
Normalized Cross Correlation (NCC)	$\frac{\sum_{(i,j)\in W} I_1(i,j).I_2(x+i,y+j)}{\sqrt[2]{\sum_{(i,j)\in W} I_1^2(i,j).\sum_{(i,j)\in W} I_2^2(x+i,y+j)}}$



SAD



SSD

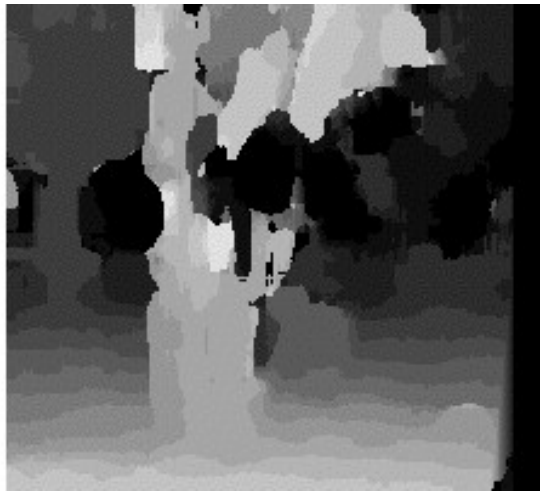


NCC

Ground truth

Effect of window size

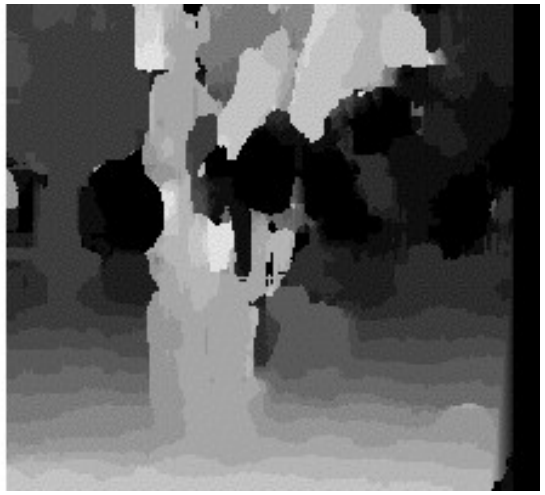
W = 3



W = 20

Effect of window size

W = 3



W = 20

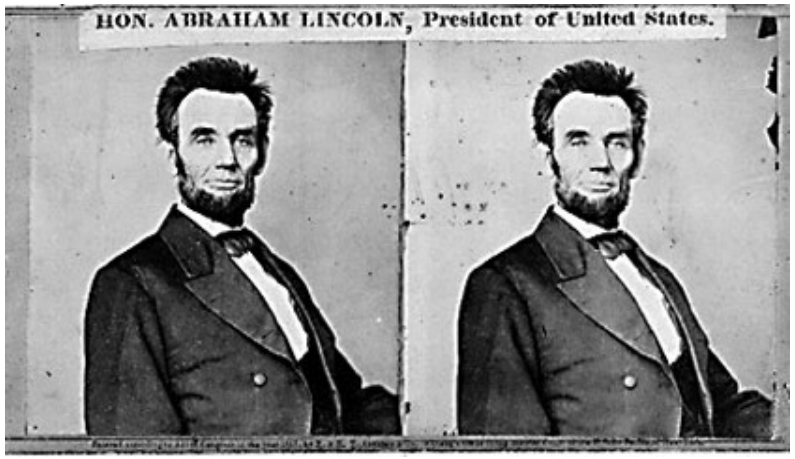
Smaller window

+ More detail
- More noise

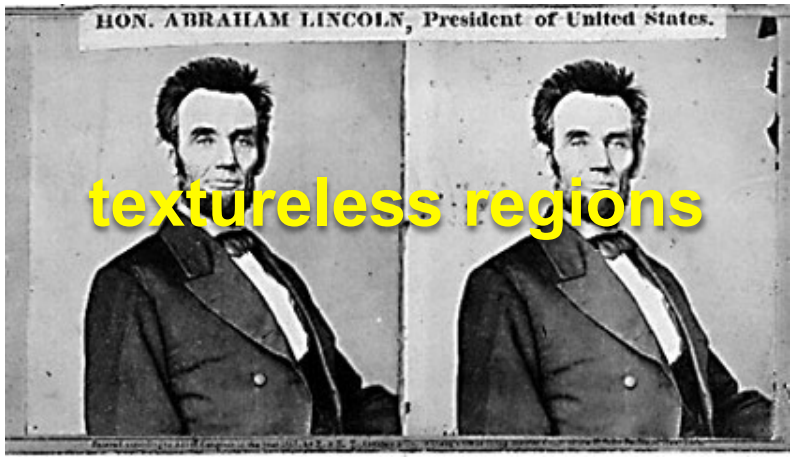
Larger window

+ Smoother disparity maps
- Less detail
- Fails near boundaries

When will stereo block matching fail?

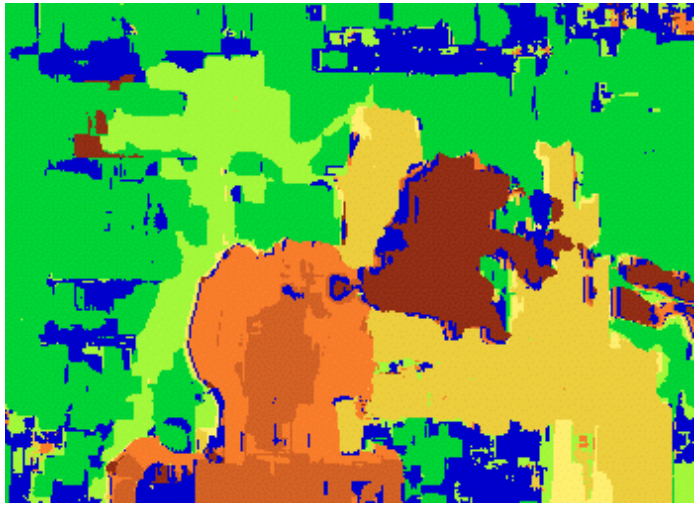


When will stereo block matching fail?



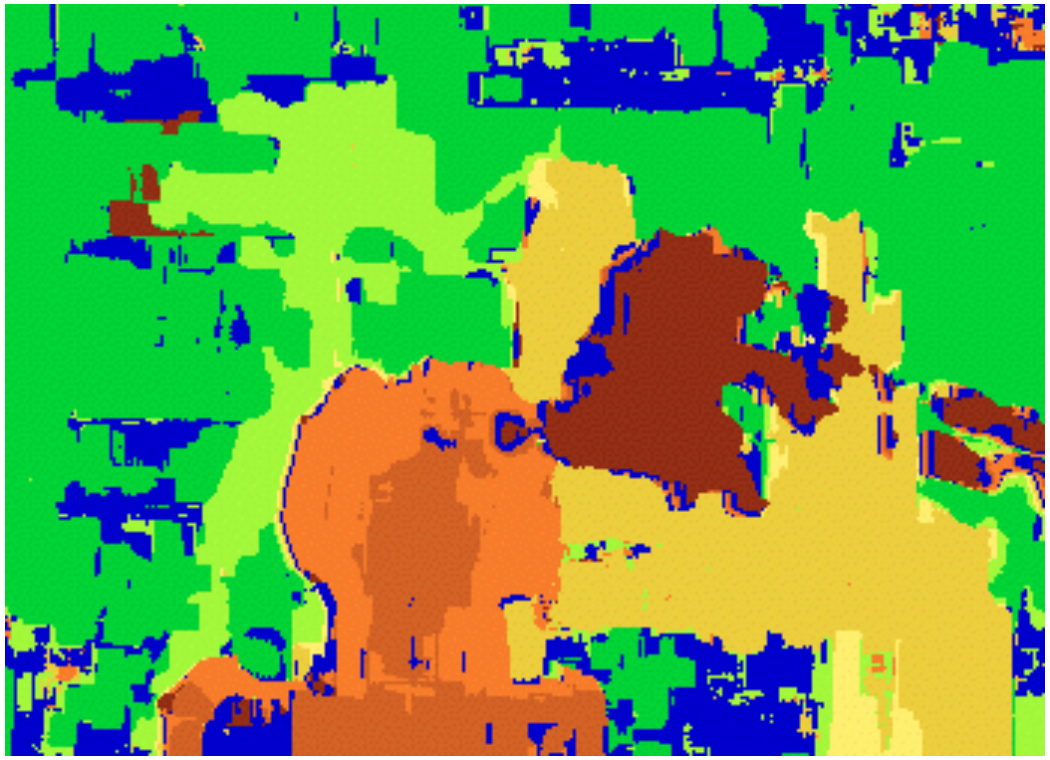
Improving Stereo Block Matching

Block matching



Ground truth

What are some problems with the result?

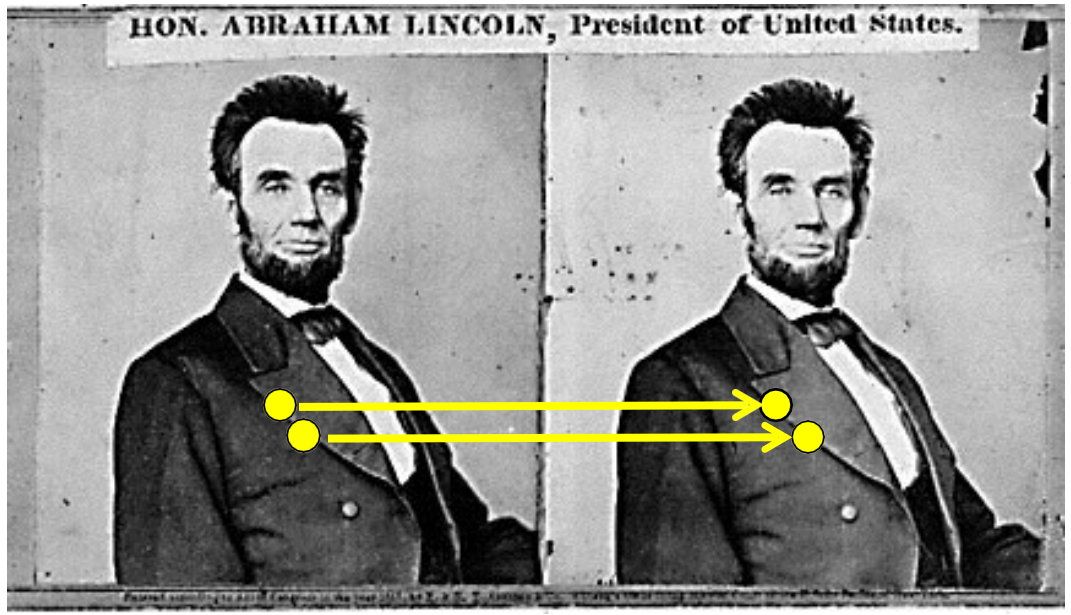


How can we improve depth estimation?

Too many discontinuities.
We expect disparity values to change slowly.

Let’s make an assumption:
depth should change smoothly

Stereo Matching as Energy Minimization



What defines a good stereo correspondence?

1. **Match quality**
   - Want each pixel to find a good match in the other image
2. **Smoothness**
   - If two pixels are adjacent, they should (usually) move about the same amount

energy function
(for one pixel)

$$E(d) = \underbrace{E_d(d)}_{\text{data term}} + \lambda \underbrace{E_s(d)}_{\text{smoothness term}}$$

energy function
(for one pixel)

$$E(d) = \underbrace{E_d(d)}_{\text{data term}} + \lambda \underbrace{E_s(d)}_{\text{smoothness term}}$$

Want each pixel to find a good match in the other image

(block matching result)

Adjacent pixels should (usually) move about the same amount

(smoothness function)

$$E(d) = E_d(d) + \lambda E_s(d)$$

$$E_d(d) = \sum_{(x,y)\in I} C(x, y, d(x, y))$$

data term

SSD distance between windows centered at I(x, y) and J(x+ d(x,y), y)

$$E(d) = E_d(d) + \lambda E_s(d)$$

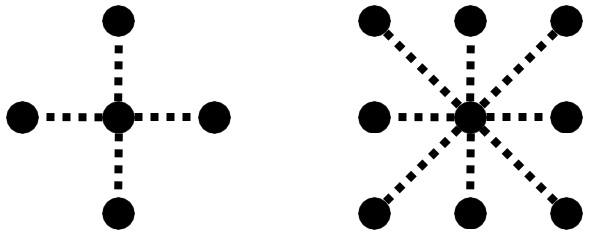
$$E_d(d) = \sum_{(x,y)\in I} C(x, y, d(x, y))$$

SSD distance between windows centered at I(x, y) and J(x+ d(x,y), y)

$$E_s(d) = \sum_{(p,q)\in\mathcal{E}} V(d_p, d_q)$$

smoothness term

$\mathcal{E}$: set of neighboring pixels



4–connected neighborhood

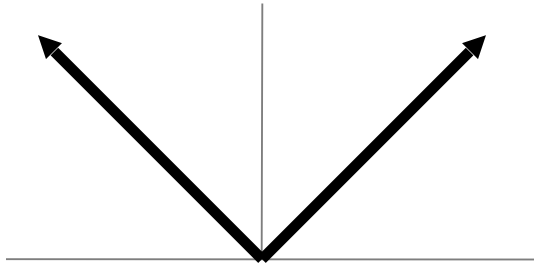
8–connected neighborhood

$$E_s(d) = \sum_{(p,q)\in\mathcal{E}} V(d_p, d_q)$$

smoothness term

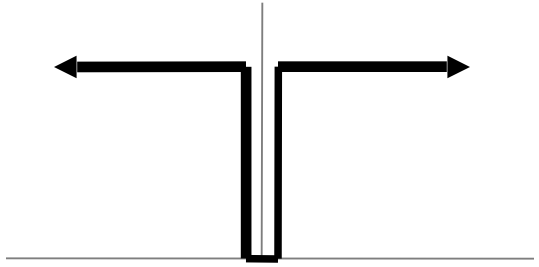
$$V(d_p, d_q) = |d_p - d_q|$$

L_1 distance



$$V(d_p, d_q) = \begin{cases} 0 & \text{if } d_p = d_q \\ 1 & \text{if } d_p \neq d_q \end{cases}$$

“Potts model”



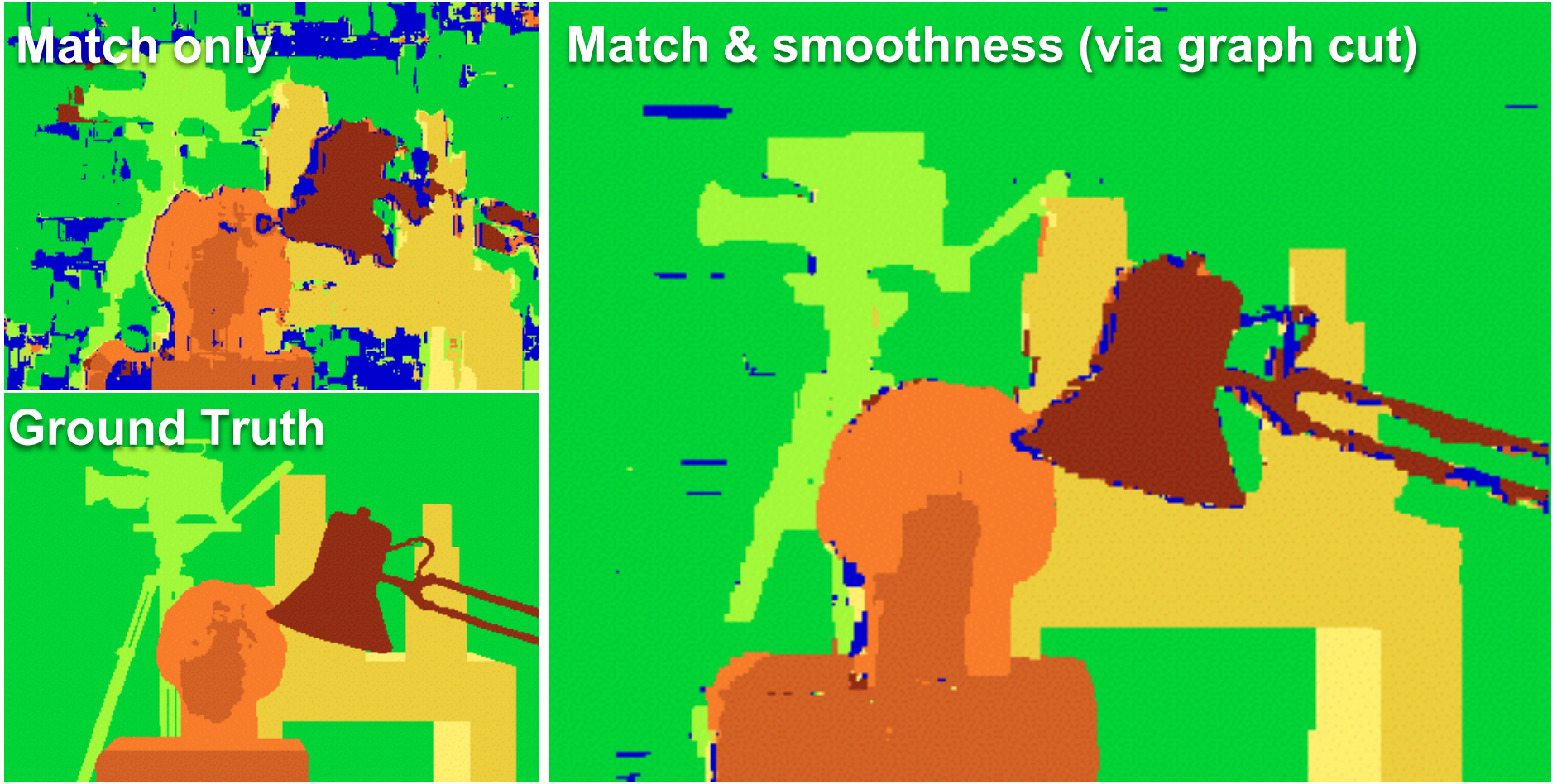
Dynamic Programming

$$E(d) = E_d(d) + \lambda E_s(d)$$

Can minimize this independently per scanline using dynamic programming (DP) ●···●···●

$D(x, y, d)$: minimum cost of solution such that d(x,y) = d

$$D(x, y, d) = C(x, y, d) + \min_{d'} \{D(x-1, y, d') + \lambda\,|d - d'|\}$$



Y. Boykov, O. Veksler, and R. Zabih, Fast Approximate Energy Minimization via Graph Cuts, PAMI 2001

Graph Cut

- Graph cuts solve this energy minimization efficiently (approximate minimum) by modeling it as a **graph.**

- The graph cut algorithm is a method for partitioning a graph into two disjoint sets by removing the minimum-weight set of edges. The basic idea is to represent the problem as a graph, where nodes represent elements to be partitioned (e.g., pixels in an image), and edges represent relationships or similarities between these elements.
- You now seek the labeling (disparity assignment) that minimizes the total energy.

Procedure:

- Graph Representation: Each pixel becomes a node, Each disparity label is a possible value, Edges: Between pixels (for smoothness term), To terminal nodes for data cost
- Edge Weight Assignment: Assign weights to the edges based on the dissimilarity or cost of separating the corresponding elements.
- Cut Cost Calculation: Define a cut in the graph by removing a set of edges. The cost of the cut is calculated by summing the weights of the removed edges.
- Minimum Cut: Find the cut with the minimum cost. This can be achieved using algorithms like the Max-Flow Min-Cut algorithm or other specialized graph-cut algorithms.
- Partitioning: After finding the minimum cut, the graph is partitioned into two disjoint sets, separating the elements into distinct segments.

Acknowledgement: some slides and material from Bernt Schiele, Mario Fritz, Michael Black, Bill Freeman, Fei-Fei, Justin Johnson, Serena Yeung, R. Szelisky, Ioannis Gkioulekas, Noah Snavely, Abe Davis, Kris Kitani, Xavier Giró-i-Nieto, Shree Nayar, Andreas Geiger