

# **CSC 422/522**

# **Computer Vision and Pattern Recognition**

**Pinhole Camera and Perspective Projection**

**2026 Spring**



**SOUTH DAKOTA  
STATE UNIVERSITY**

# Today

- Learn basics of image formation
  - An overlapping area for computer vision and computer graphics (especially rendering)
- Focusing on the geometric side
  - Pinhole camera
  - Coordinate system
  - Perspective projection





**SOUTH DAKOTA  
STATE UNIVERSITY**

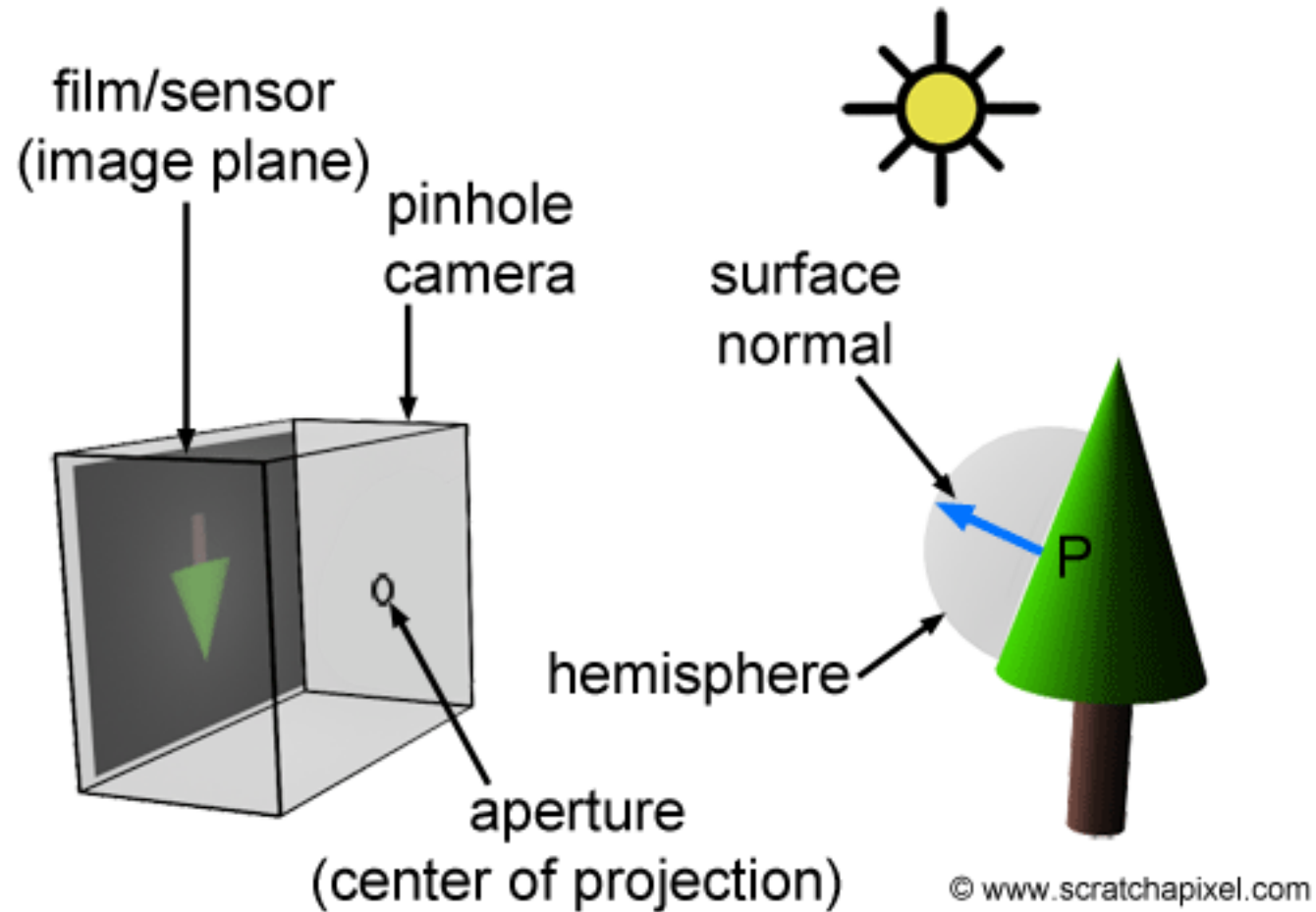


Source: <https://en.wikipedia.org/wiki/Photography>

# How is an Image Formed?

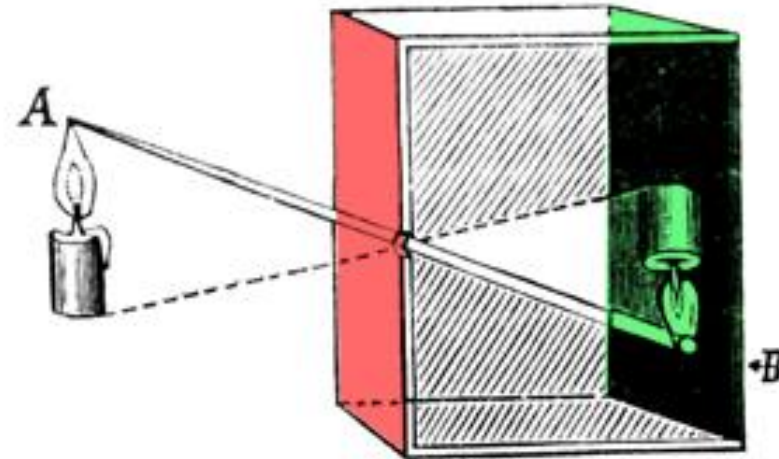
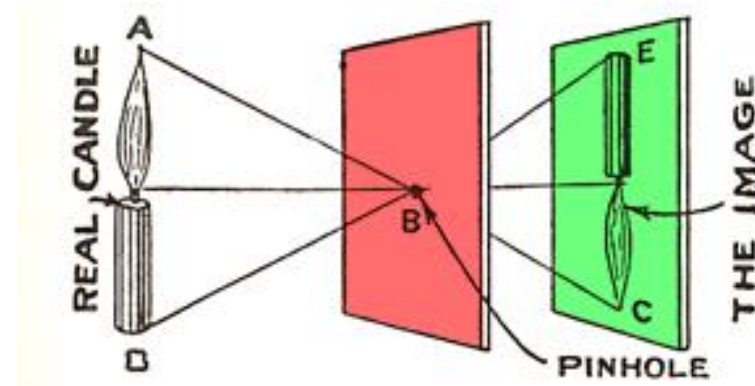
# How Does a Real Camera Work?

- In a real camera, images are created when light falls on a surface sensitive to light.
  - Film camera: film
  - Digital camera: sensor (or CCD).
- In the real world, when light from a light source reaches an object, it is reflected in many directions. However, only one ray travels in the direction of the camera and hits the film's surface or the CCD, as illustrated.



# Pinhole Cameras

- The pinhole camera and camera obscura principle illustrated in 1925, in *The Boy Scientist*.
- This illustration depicts the simplicity and fundamental mechanics of the pinhole camera model, highlighting its significance in the evolution of photographic techniques.



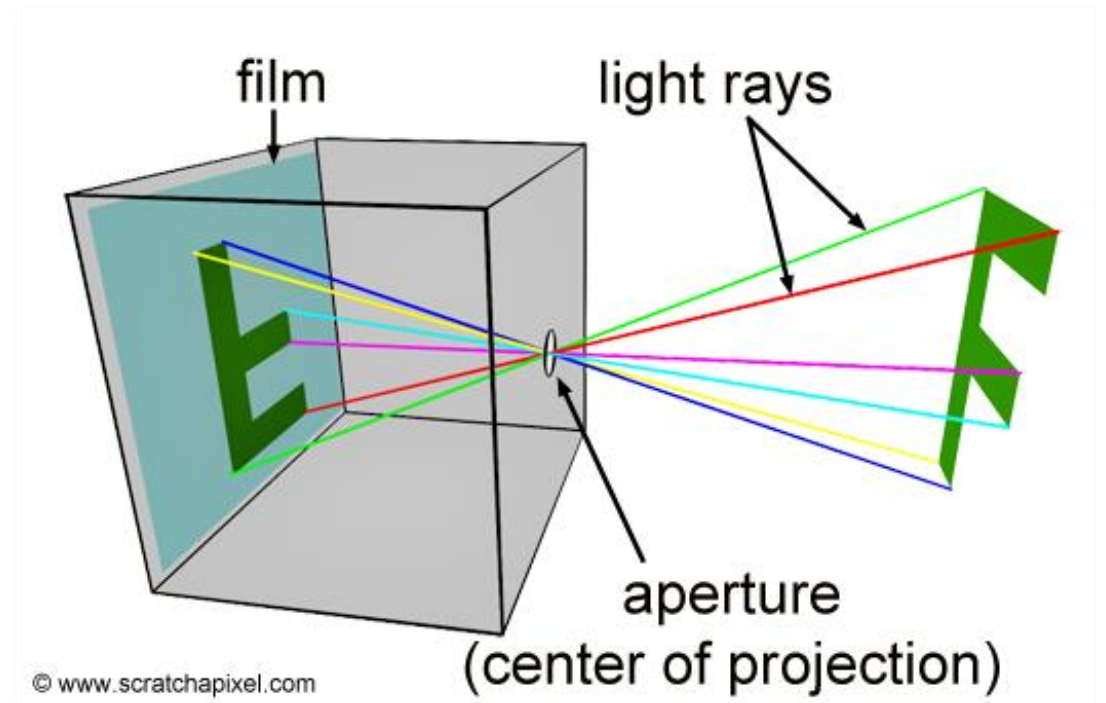
Reference: <https://www.scratchapixel.com/lessons/3d-basic-rendering/3d-viewing-pinhole-camera/how-pinhole-camera-works-part-1.html>



SOUTH DAKOTA  
STATE UNIVERSITY

# Pinhole Cameras

- The simplest type of camera.
- It consists of a simple, lightproof box with a very small hole in the front, known as an **aperture**, and some light-sensitive film or paper placed inside the box on the side facing this pinhole.
- To take a picture, you simply open the aperture to expose the film to light.





**SOUTH DAKOTA  
STATE UNIVERSITY**

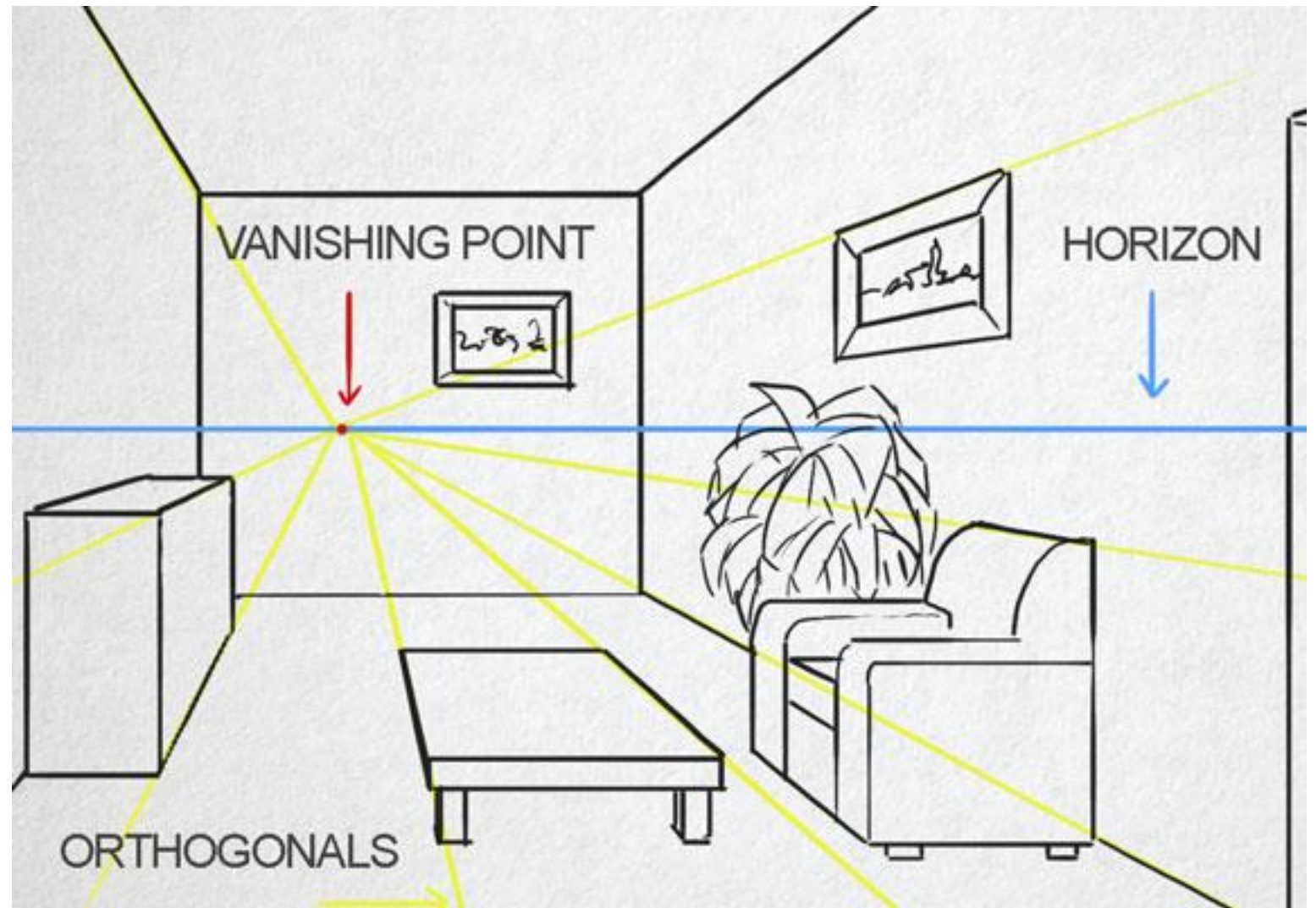
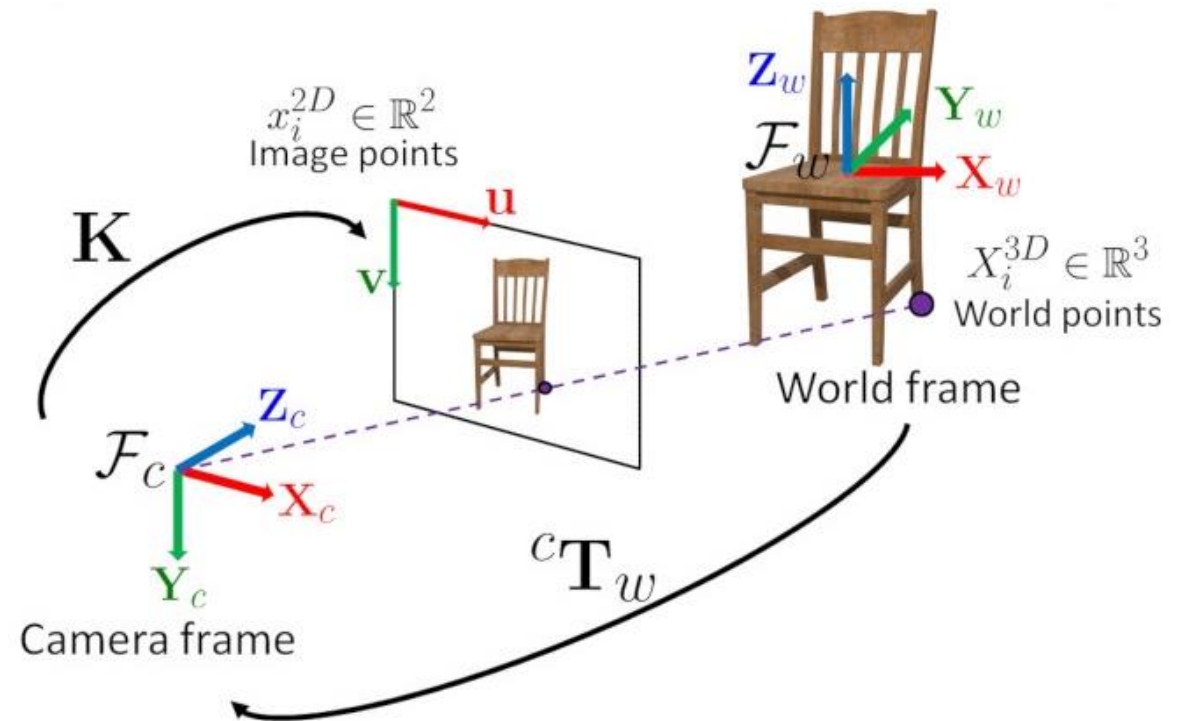
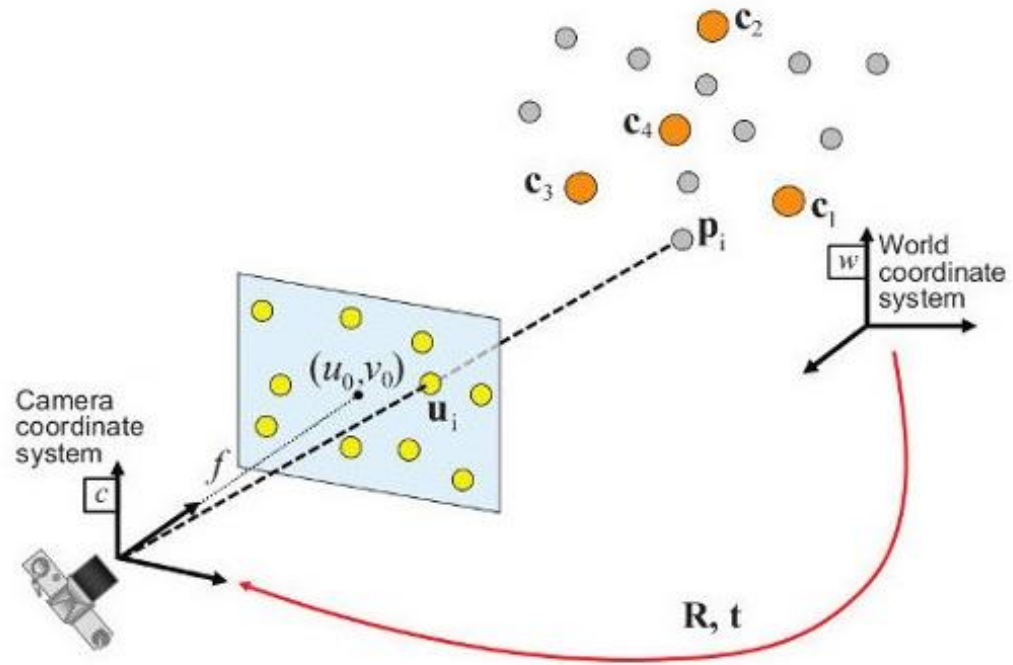


Image source: <https://thevirtualinstructor.com/onepointperspective.html>

# Perspective Projection

# Coordinate System

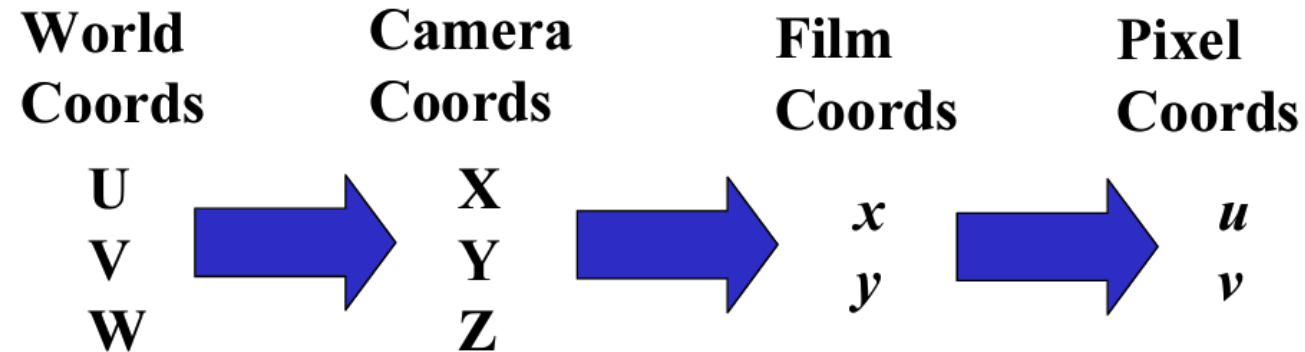


Reference: [https://docs.opencv.org/4.x/d5/d1f/calib3d\\_solvePnP.html](https://docs.opencv.org/4.x/d5/d1f/calib3d_solvePnP.html)



SOUTH DAKOTA  
STATE UNIVERSITY

# Forward Projection



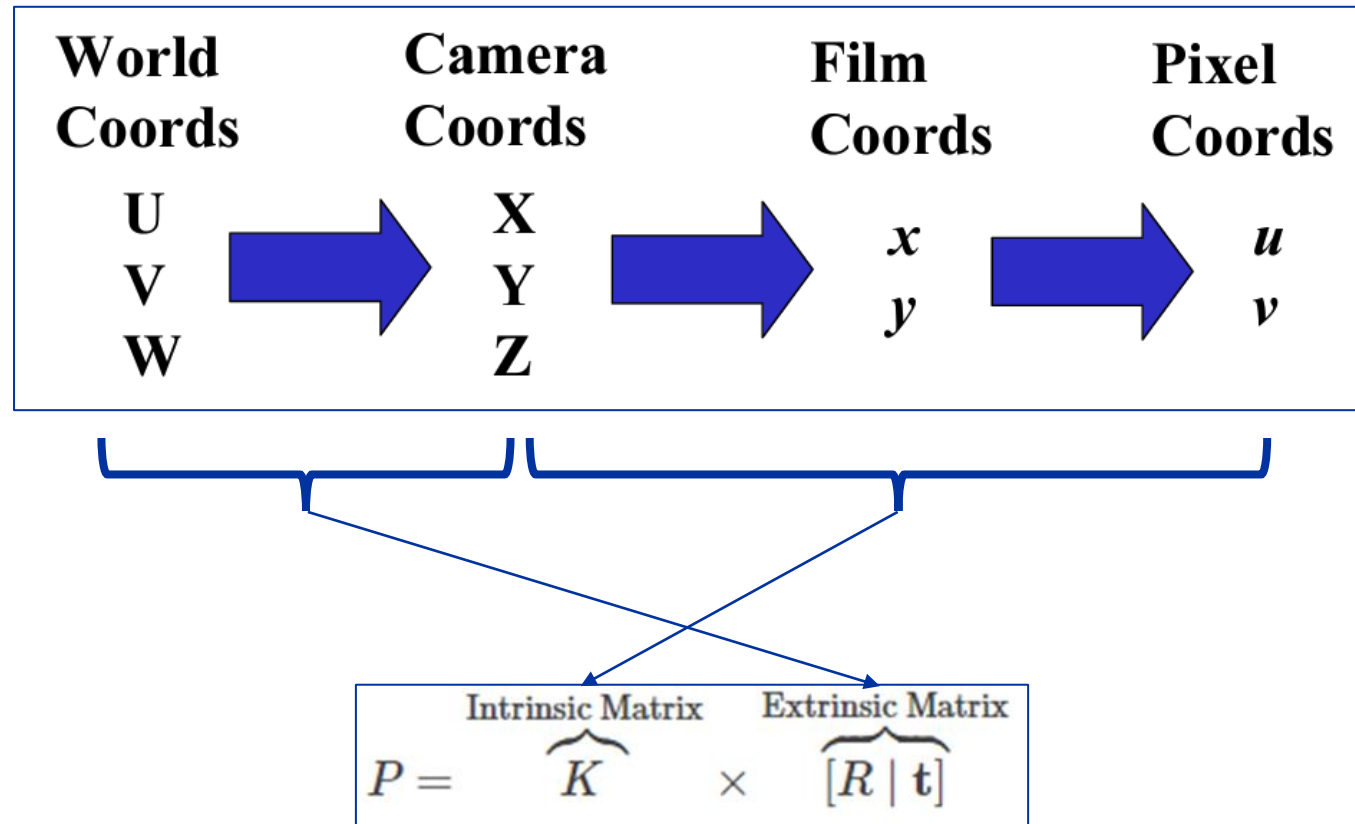
We want a mathematical model to describe how 3D World points get projected into 2D Pixel coordinates.

**Our goal: describe this sequence of transformations by a big matrix equation!**

Slide Credit: CSC492/592, 2024 Spring



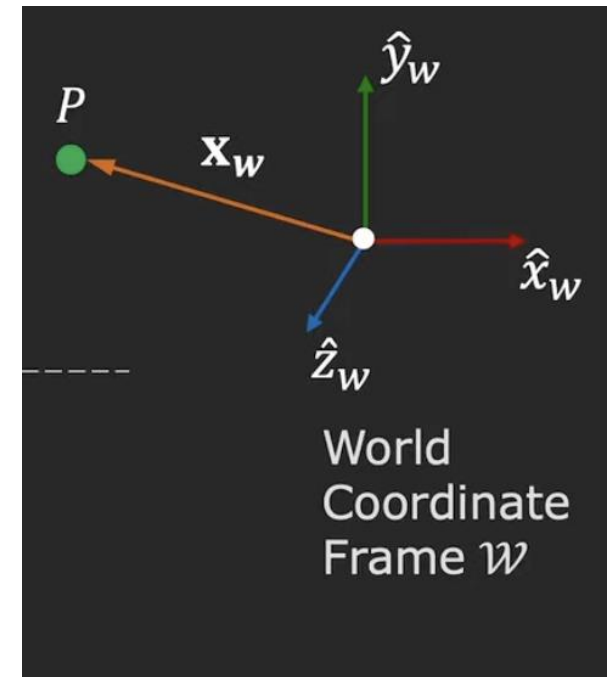
# Forward Projection



Slide Credit: CSC492/592, 2024 Spring



# Step-by-Step Mathematical Derivation

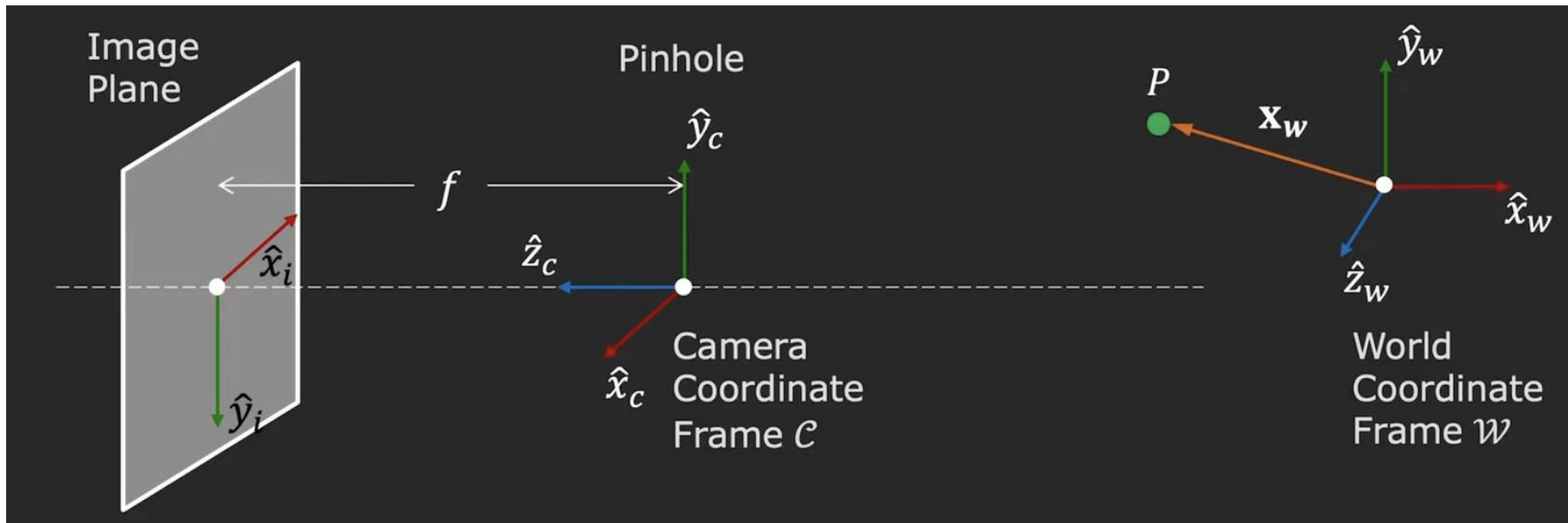


Note: some illustrations taken from <https://www.youtube.com/watch?v=qByYk6JggQU>



SOUTH DAKOTA  
STATE UNIVERSITY

# Step-by-Step Mathematical Derivation



Note: some illustrations taken from <https://www.youtube.com/watch?v=qByYk6JggQU>



SOUTH DAKOTA  
STATE UNIVERSITY

# Step-by-Step Mathematical Derivation

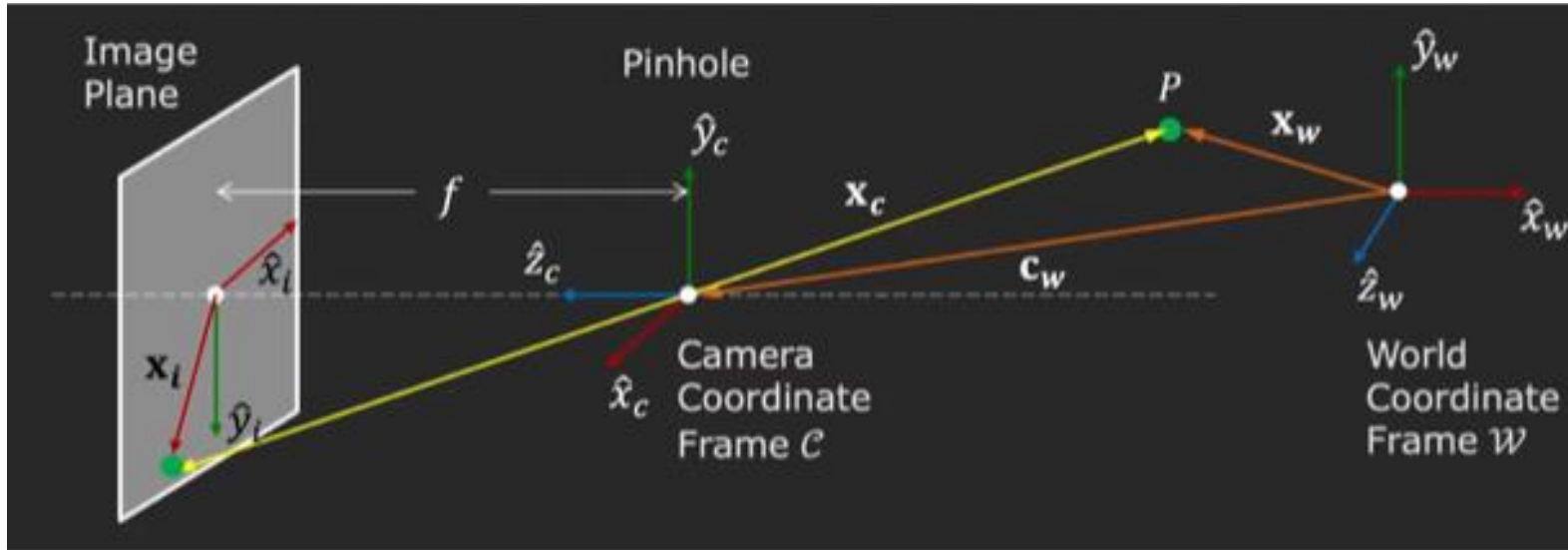


Image Coordinates

$$X_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$



Perspective Projection

Camera Coordinates

$$X_c = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$$



Coordinate Transformation

World Coordinates

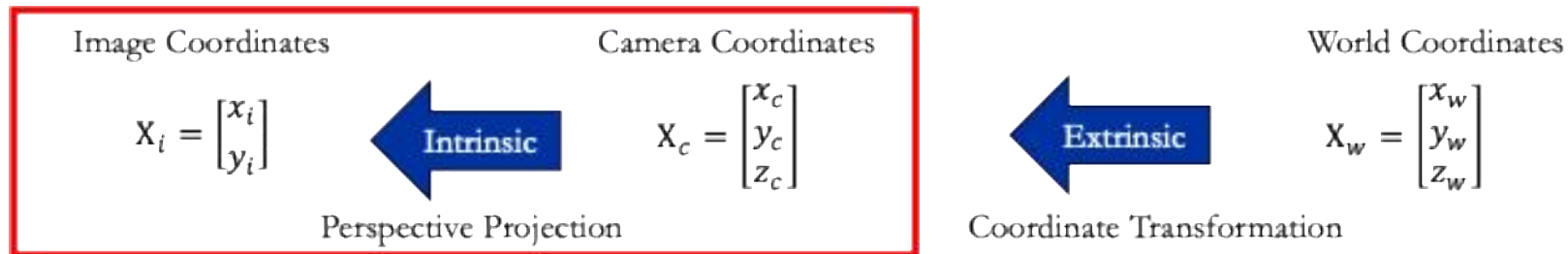
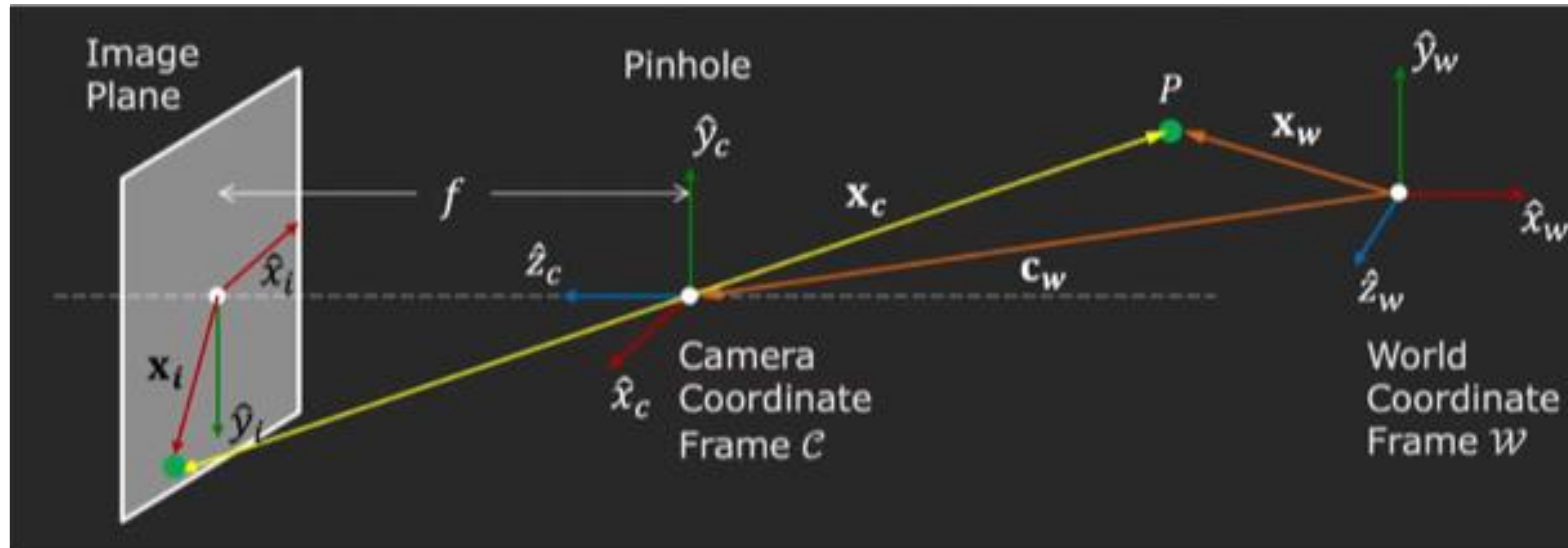
$$X_w = \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix}$$

Note: some illustrations taken from <https://www.youtube.com/watch?v=qByYk6JggQU>



SOUTH DAKOTA  
STATE UNIVERSITY

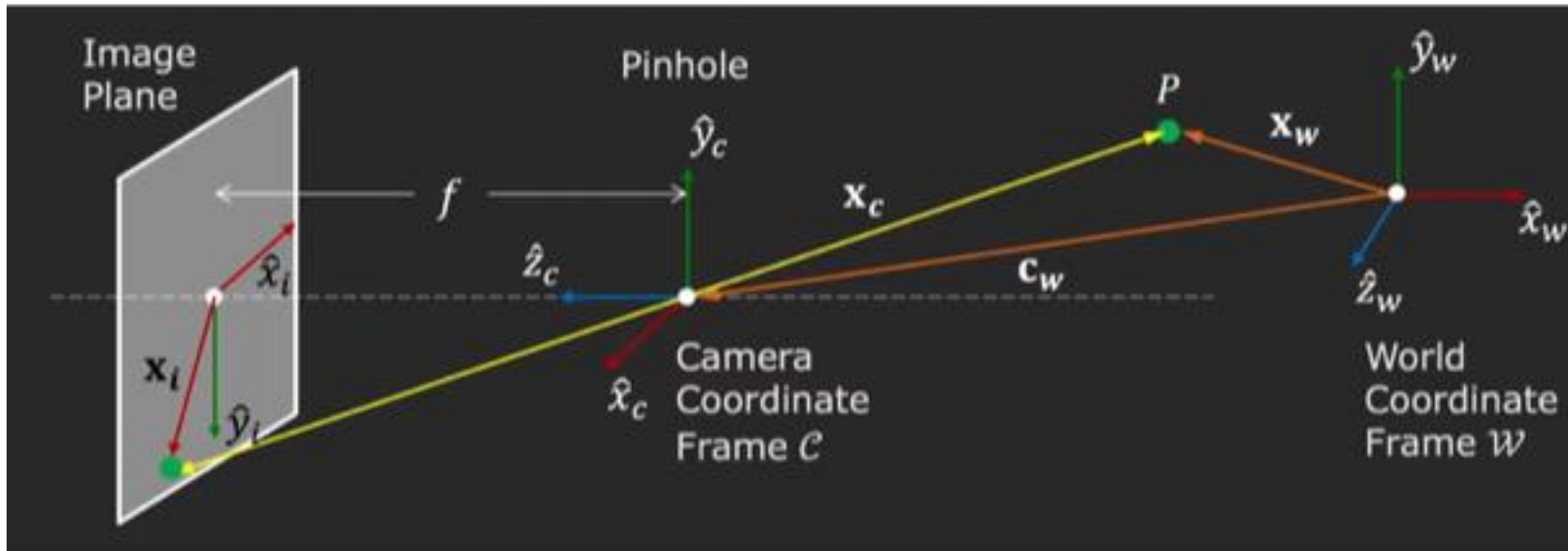
# Step-by-Step Mathematical Derivation



Note: some illustrations taken from <https://www.youtube.com/watch?v=qByYk6JggQU>



# Step-by-Step Mathematical Derivation

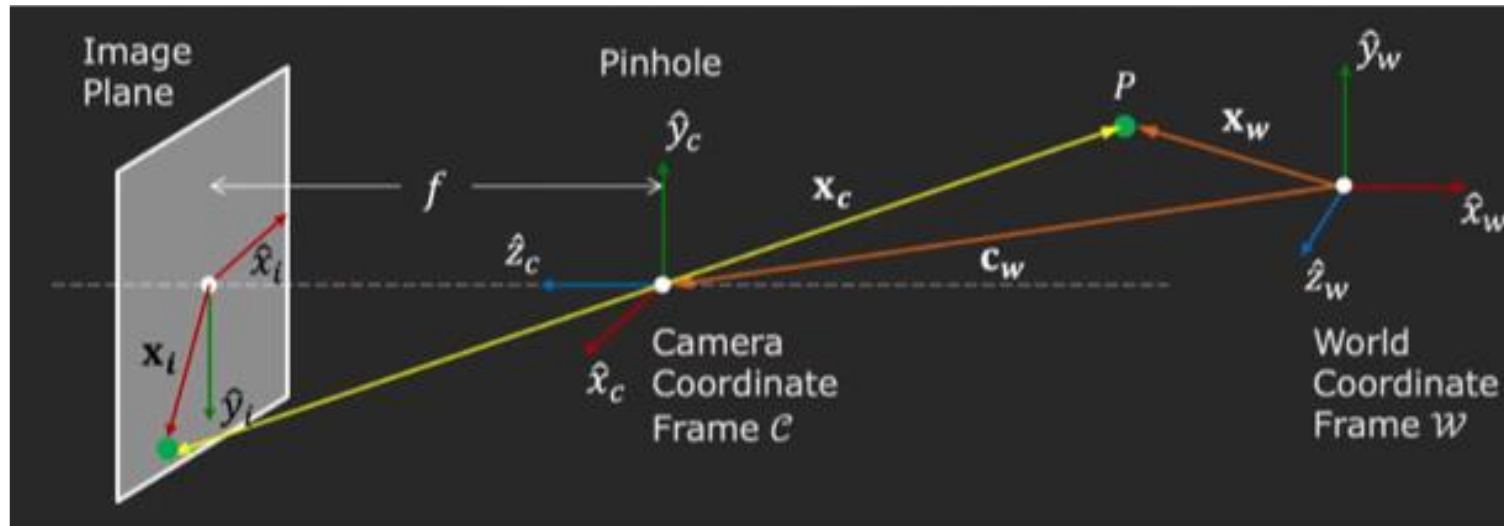
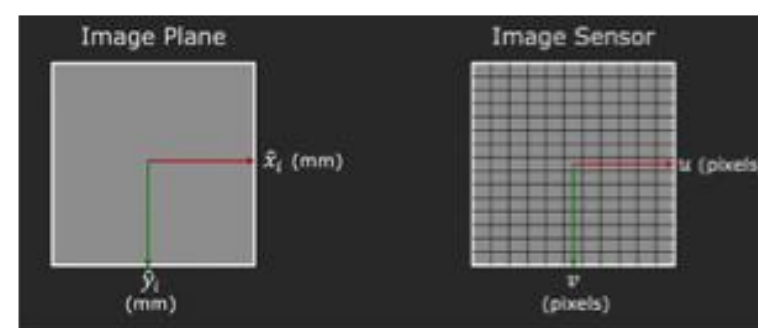


$$\frac{x_i}{f} = \frac{x_c}{z_c} \quad \text{and} \quad \frac{y_i}{f} = \frac{y_c}{z_c} \quad \Rightarrow \quad x_i = f \frac{x_c}{z_c} \quad \text{and} \quad y_i = f \frac{y_c}{z_c}$$

Note: some illustrations taken from <https://www.youtube.com/watch?v=qByYk6JggQU>



# Step-by-Step Mathematical Derivation



Given that  $m_x$  and  $m_y$  are the pixel densities (pixels/mm):

$$u = m_x f \frac{x_c}{z_c} \quad \text{and} \quad v = m_y f \frac{y_c}{z_c}$$

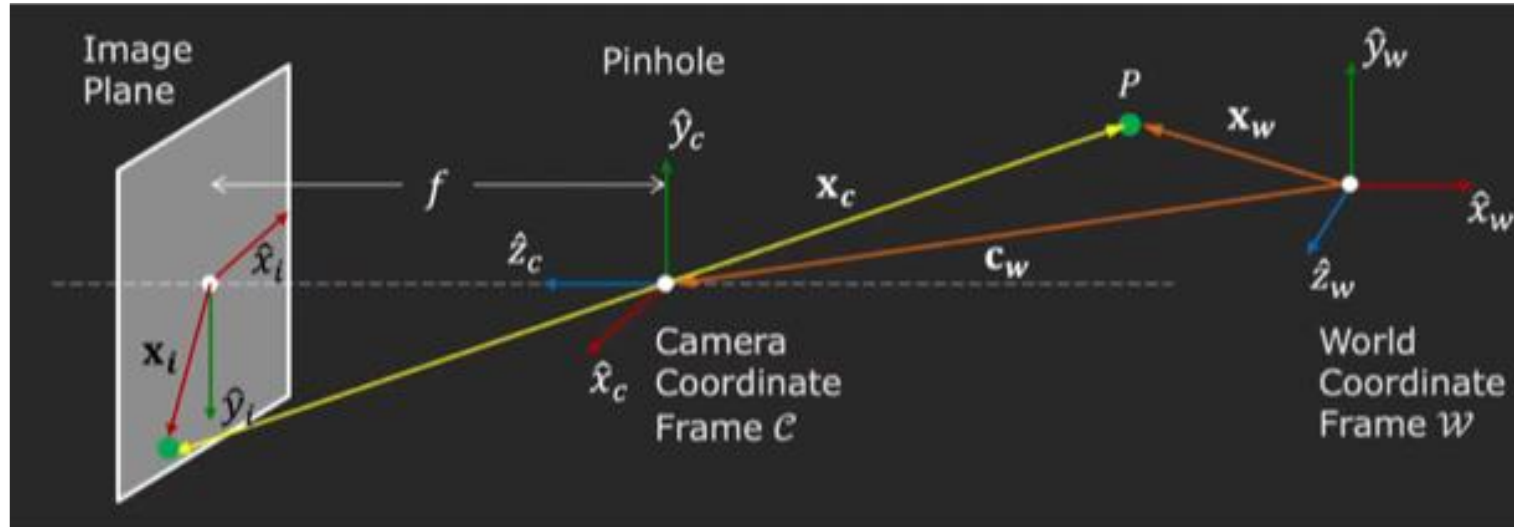
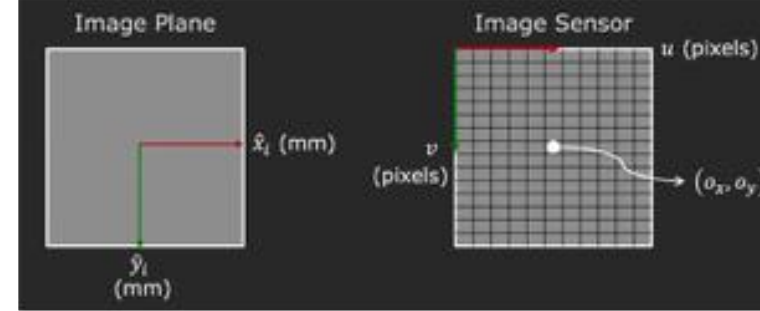
$$x_i = f \frac{x_c}{z_c} \quad \text{and} \quad y_i = f \frac{y_c}{z_c}$$

Note: some illustrations taken from <https://www.youtube.com/watch?v=qByYk6JggQU>



**SOUTH DAKOTA  
STATE UNIVERSITY**

# Step-by-Step Mathematical Derivation



Given that  $m_x$  and  $m_y$  are the pixel densities (pixels/mm):

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

$$x_i = f \frac{x_c}{z_c} \quad \text{and} \quad y_i = f \frac{y_c}{z_c}$$

Note: some illustrations taken from <https://www.youtube.com/watch?v=qByYk6JggQU>



**SOUTH DAKOTA  
STATE UNIVERSITY**

# Step-by-Step Mathematical Derivation

Perspective projection equations:

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

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

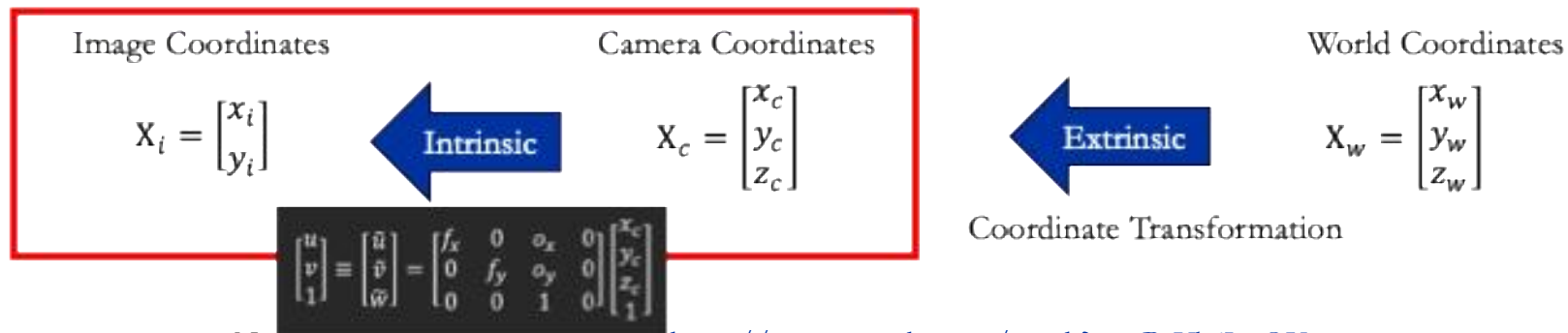
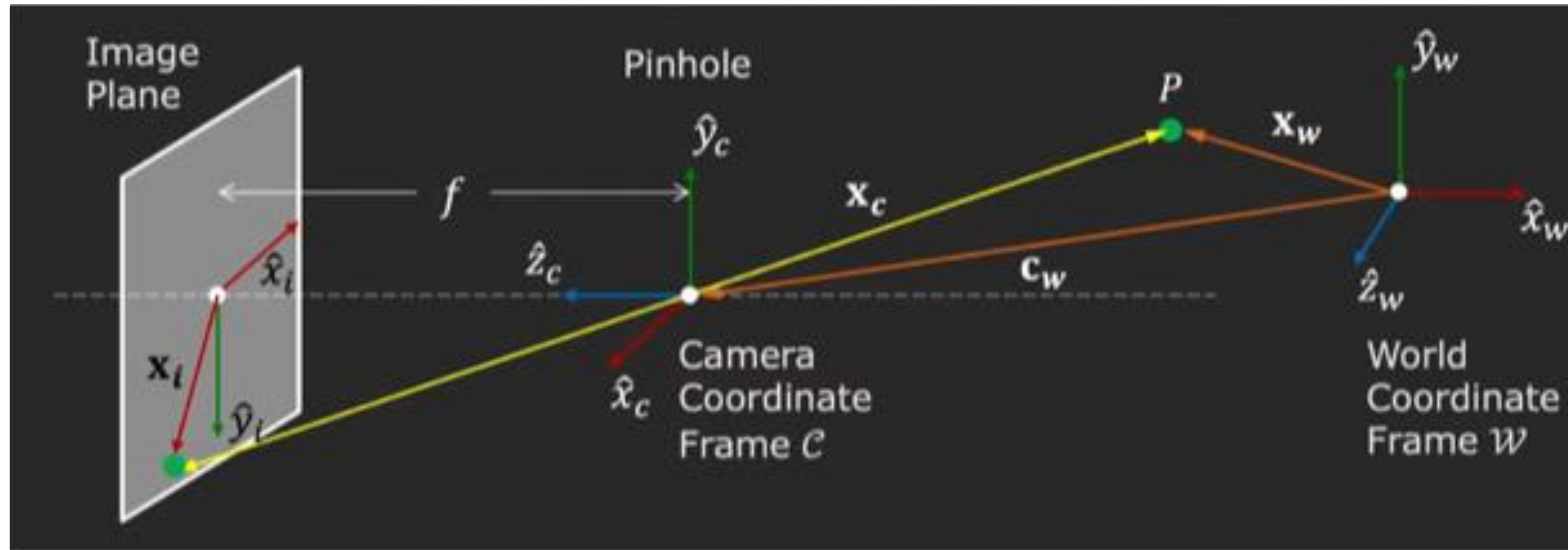
Homogenous coordinates of  $(u, v)$  :

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

Note: some illustrations taken from <https://www.youtube.com/watch?v=qByYk6JggQU>



# Now We Have Obtained Intrinsic Matrix



Note: some illustrations taken from <https://www.youtube.com/watch?v=qByYk6JggQU>



# Todo: World to Camera Transformation

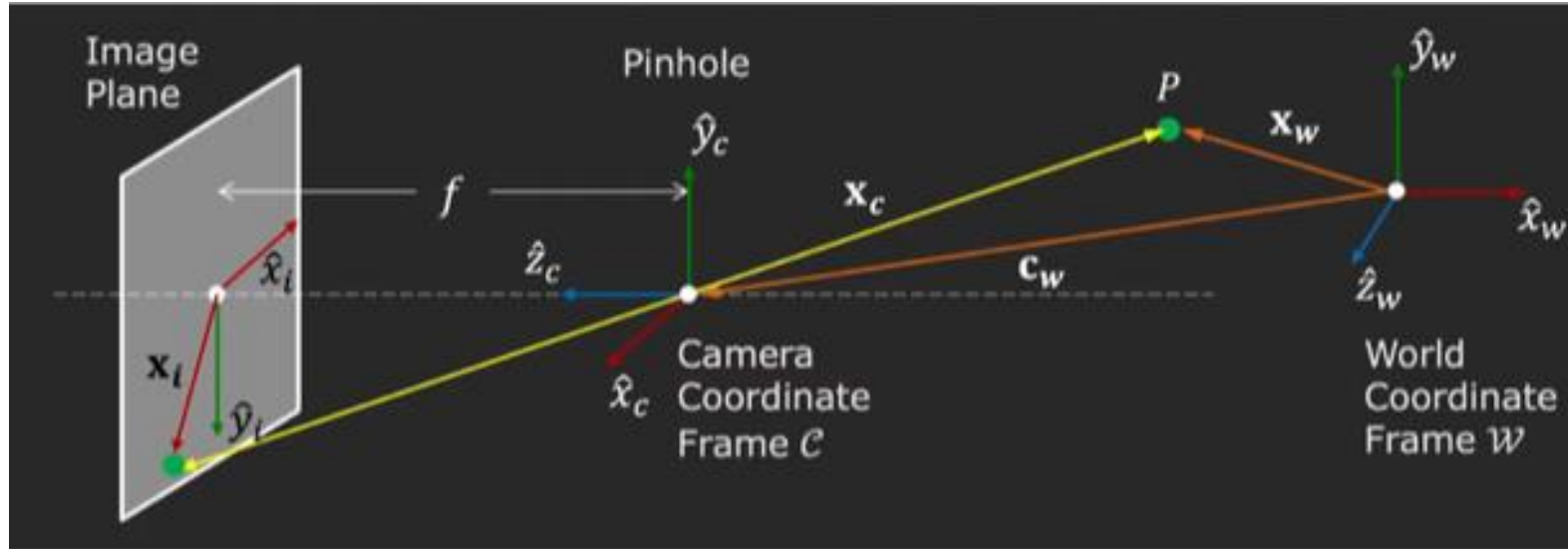


Image Coordinates

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

Intrinsic

Perspective Projection

Camera Coordinates

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

Extrinsic

Coordinate Transformation

World Coordinates

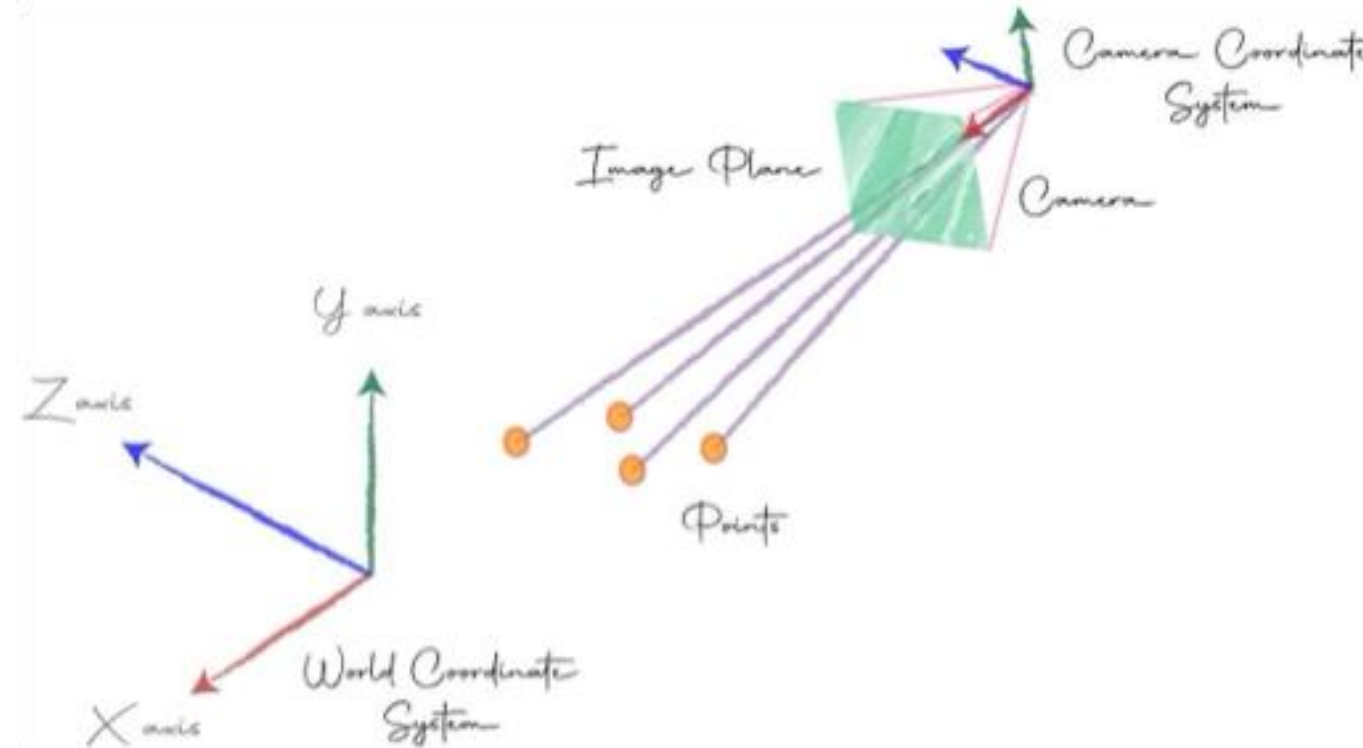
$$\mathbf{X}_w = \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix}$$

Note: some illustrations taken from <https://www.youtube.com/watch?v=qByYk6JggQU>



SOUTH DAKOTA  
STATE UNIVERSITY

# Todo: World to Camera Transformation



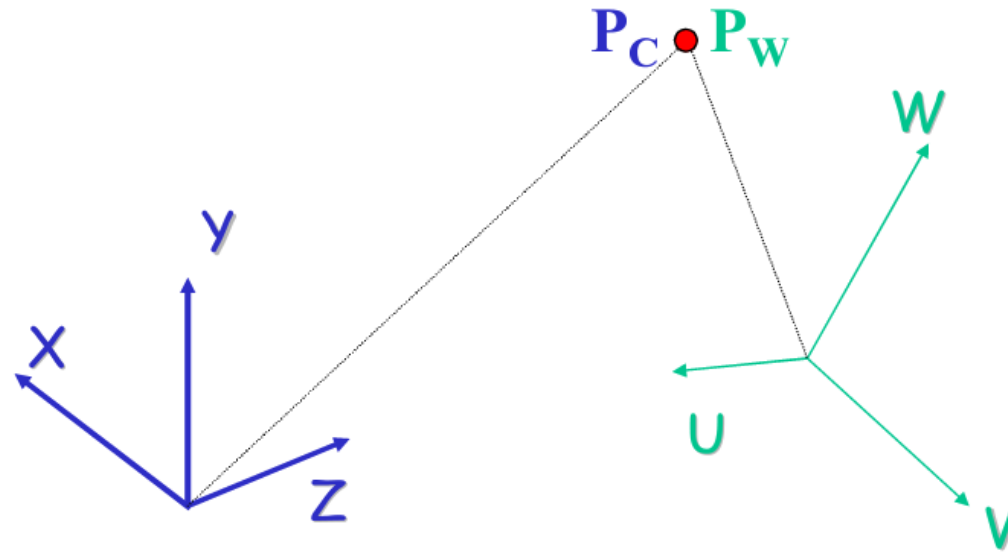
- How can we find the camera pose?
  - Rotation → orient the camera
  - Translation → move the camera

Slide Credit: CSC 492/592, 2024 Spring



SOUTH DAKOTA  
STATE UNIVERSITY

# World to Camera Transformation

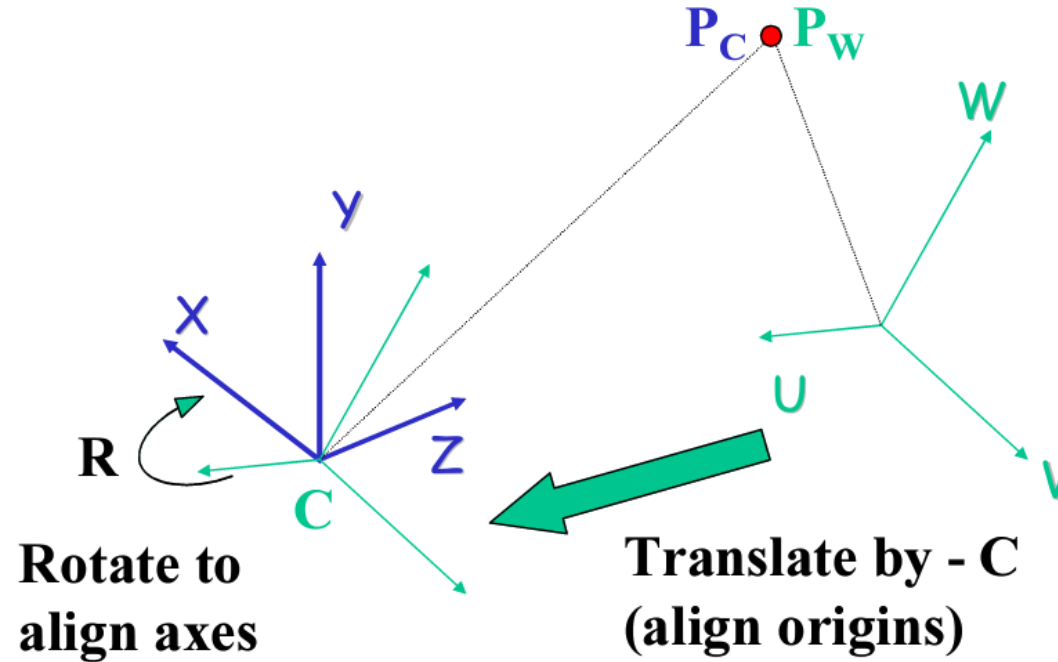


Slide Credit: CSC 492/592, 2024 Spring



SOUTH DAKOTA  
STATE UNIVERSITY

# World to Camera Transformation



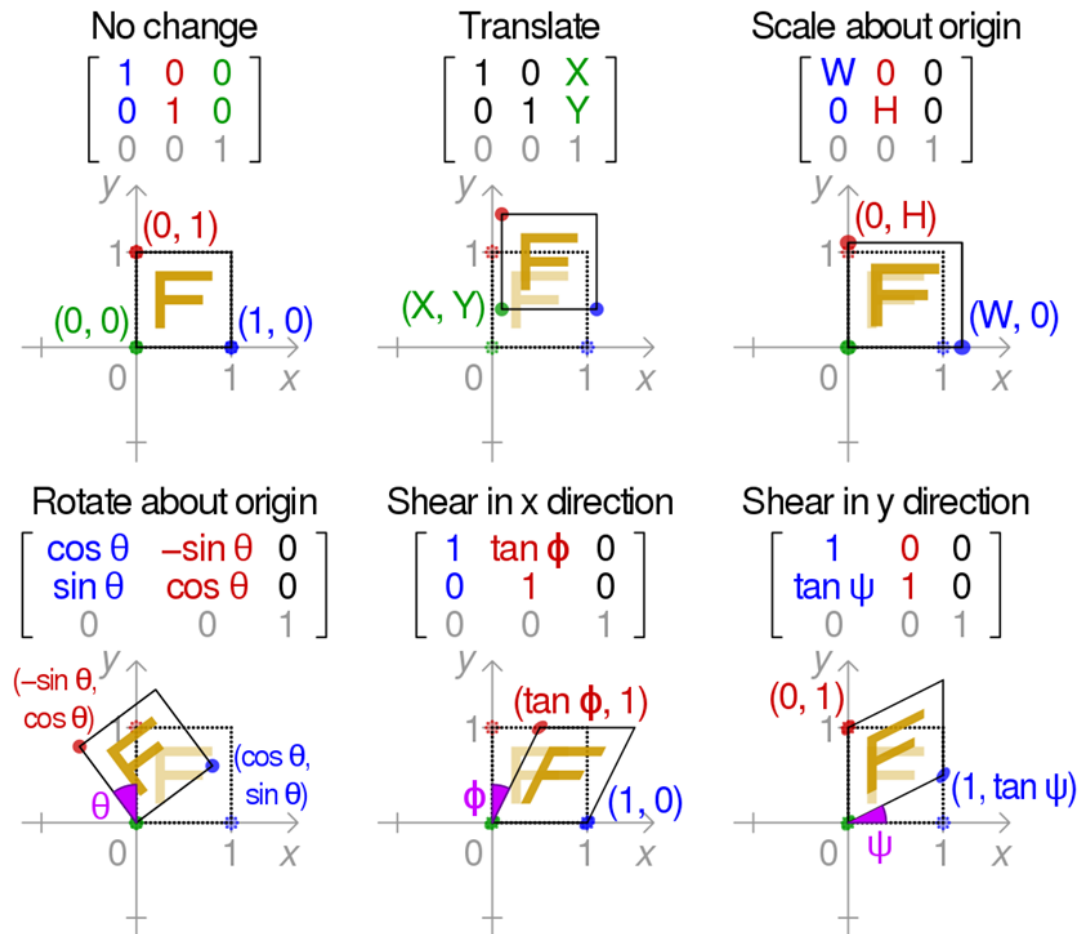
$$P_C = R ( P_W - C )$$

Slide Credit: CSC 492/592, 2024 Spring



SOUTH DAKOTA  
STATE UNIVERSITY

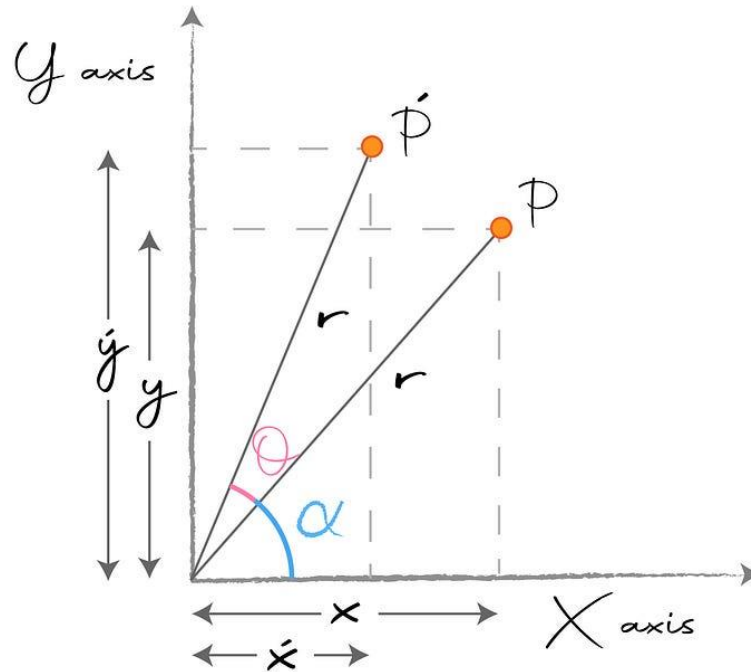
# Basic of Transformation Matrix (2D Version)



Slide Credit: CSC 492/592, 2024 Spring



# Basic of Transformation Matrix: 2D Rotation



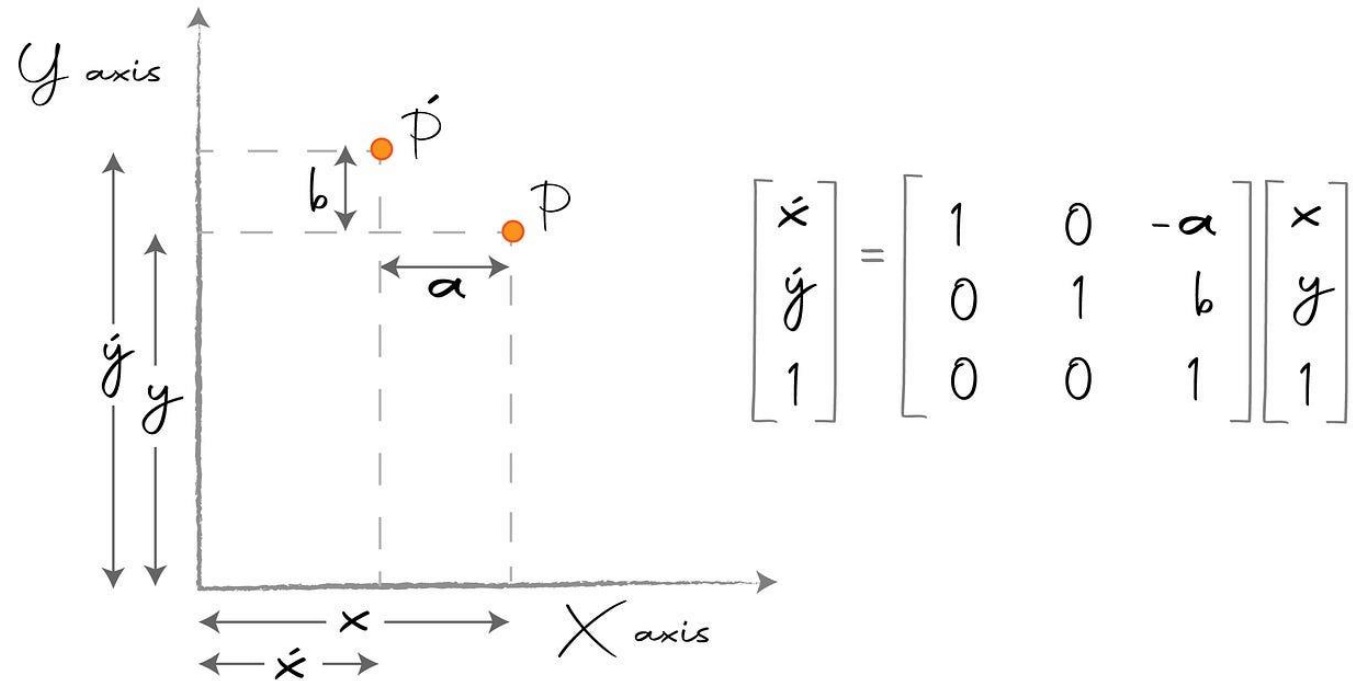
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Slide Credit: CSC 492/592, 2024 Spring



SOUTH DAKOTA  
STATE UNIVERSITY

# Basic of Transformation Matrix: 2D Translation



Slide Credit: CSC 492/592, 2024 Spring



SOUTH DAKOTA  
STATE UNIVERSITY

# Basic of Transformation Matrix (2D Version)






Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} \mathbf{I} & \mathbf{t} \end{bmatrix}_{2 \times 3}$	2	orientation	
rigid (Euclidean)	$\begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}_{2 \times 3}$	3	lengths	
similarity	$\begin{bmatrix} s\mathbf{R} & \mathbf{t} \end{bmatrix}_{2 \times 3}$	4	angles	
affine	$\begin{bmatrix} \mathbf{A} \end{bmatrix}_{2 \times 3}$	6	parallelism	
projective	$\begin{bmatrix} \tilde{\mathbf{H}} \end{bmatrix}_{3 \times 3}$	8	straight lines	

Table 2.1 from Computer Vision: Algorithms and Applications, 2nd ed



# Basic of Transformation Matrix (3D Version)






Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} \mathbf{I} & \mathbf{t} \end{bmatrix}_{3 \times 4}$	3	orientation	
rigid (Euclidean)	$\begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}_{3 \times 4}$	6	lengths	
similarity	$\begin{bmatrix} s\mathbf{R} & \mathbf{t} \end{bmatrix}_{3 \times 4}$	7	angles	
affine	$\begin{bmatrix} \mathbf{A} \end{bmatrix}_{3 \times 4}$	12	parallelism	
projective	$\begin{bmatrix} \tilde{\mathbf{H}} \end{bmatrix}_{4 \times 4}$	15	straight lines	

Table 2.2 from Computer Vision: Algorithms and Applications, 2nd ed



# Basic of Transformation Matrix (3D Version)

**Translation.** 3D translations can be written as  $\mathbf{x}' = \mathbf{x} + \mathbf{t}$  or

$$\mathbf{x}' = \begin{bmatrix} \mathbf{I} & \mathbf{t} \end{bmatrix} \bar{\mathbf{x}}, \quad (2.23)$$

where  $\mathbf{I}$  is the  $(3 \times 3)$  identity matrix.


**Rotation + translation.** Also known as 3D *rigid body motion* or the 3D *Euclidean transformation* or  $SE(3)$ , it can be written as  $\mathbf{x}' = \mathbf{R}\mathbf{x} + \mathbf{t}$  or

$$\mathbf{x}' = \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \bar{\mathbf{x}}, \quad (2.24)$$

where  $\mathbf{R}$  is a  $3 \times 3$  orthonormal rotation matrix with  $\mathbf{R}\mathbf{R}^T = \mathbf{I}$  and  $|\mathbf{R}| = 1$ . Note that sometimes it is more convenient to describe a rigid motion using

$$\mathbf{x}' = \mathbf{R}(\mathbf{x} - \mathbf{c}) = \mathbf{R}\mathbf{x} - \mathbf{R}\mathbf{c}, \quad (2.25)$$

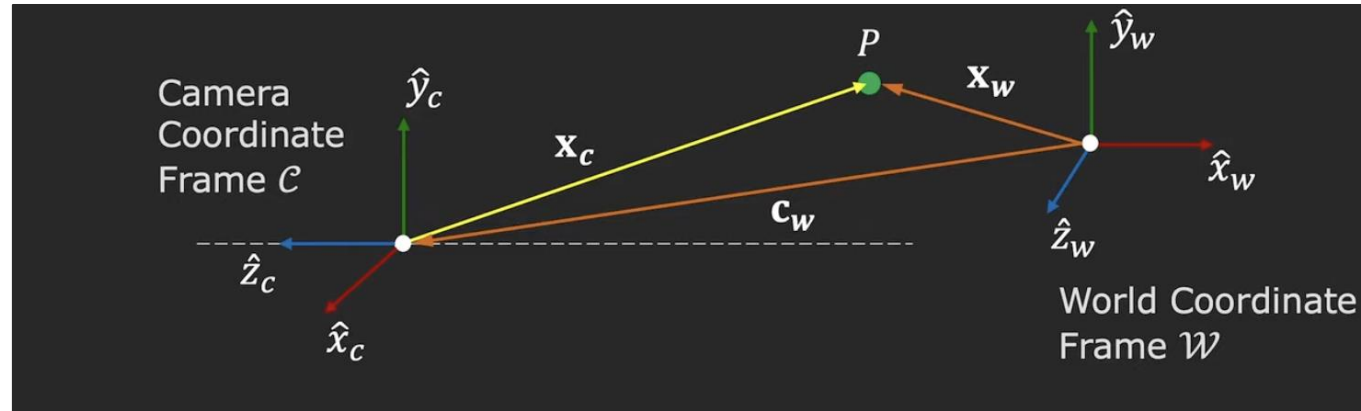
where  $\mathbf{c}$  is the center of rotation (often the camera center).

The biggest difference between 2D and 3D coordinate transformations is that the parameterization of the 3D rotation matrix  $\mathbf{R}$  is not as straightforward, as several different possibilities exist. 

from Computer Vision: Algorithms and Applications, 2nd ed



# Step-by-Step Mathematical Derivation



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:

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

Note: some illustrations taken from <https://www.youtube.com/watch?v=qByYk6JggQU>



# Step-by-Step Mathematical Derivation

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

Note: some illustrations taken from <https://www.youtube.com/watch?v=qByYk6JggQU>



# Full 3D to 2D Projection

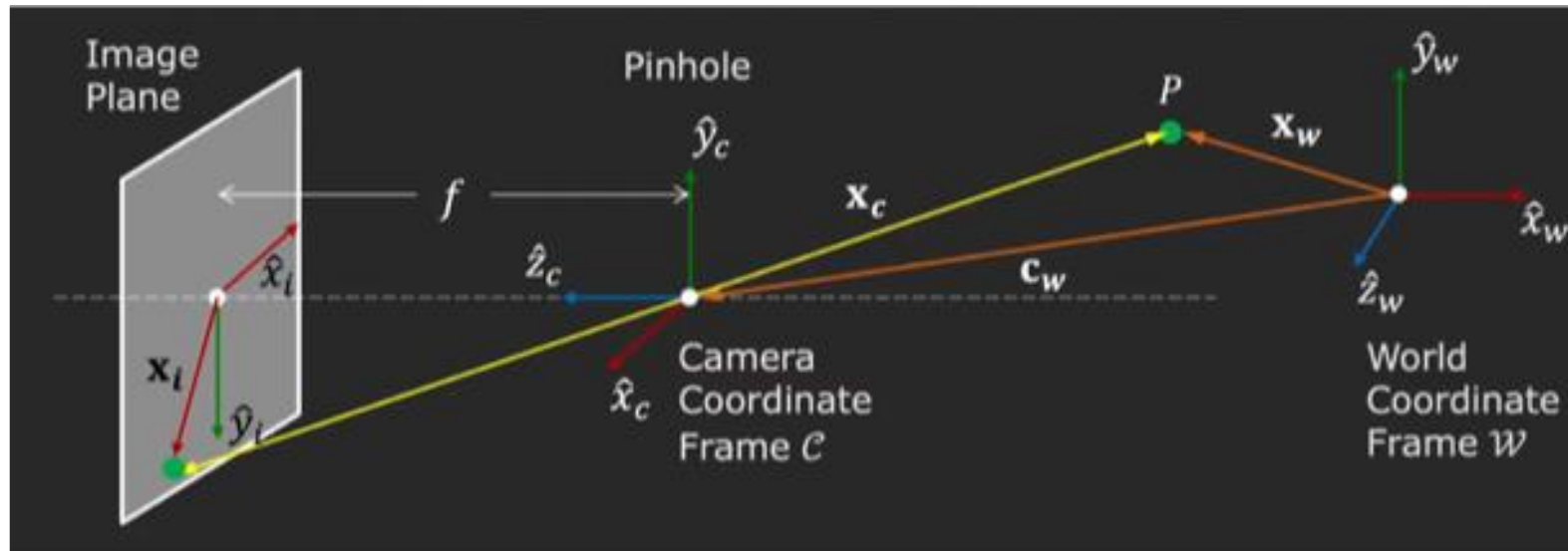


Image Coordinates

$$X_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$

**Intrinsic**

Camera Coordinates

$$X_c = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$$

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

**Extrinsic**

World Coordinates

$$X_w = \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix}$$

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

Note: some illustrations taken from <https://www.youtube.com/watch?v=qByYk6lggOU>



**SOUTH DAKOTA  
STATE UNIVERSITY**

# 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  $P$ :

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

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

Note: some illustrations taken from <https://www.youtube.com/watch?v=qByYk6JggQU>



# Photometric Image Formation?

- We have described how 3D geometric features in the world are projected into 2D features in an image.
- However, images are not composed of 2D features. Instead, they are made up of discrete color or intensity values.
- Where do these values come from?

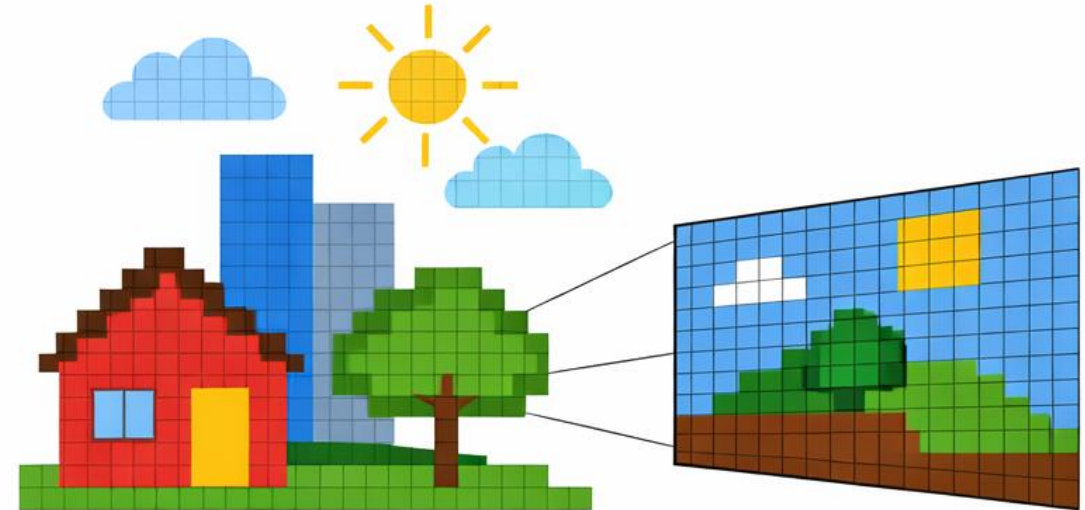
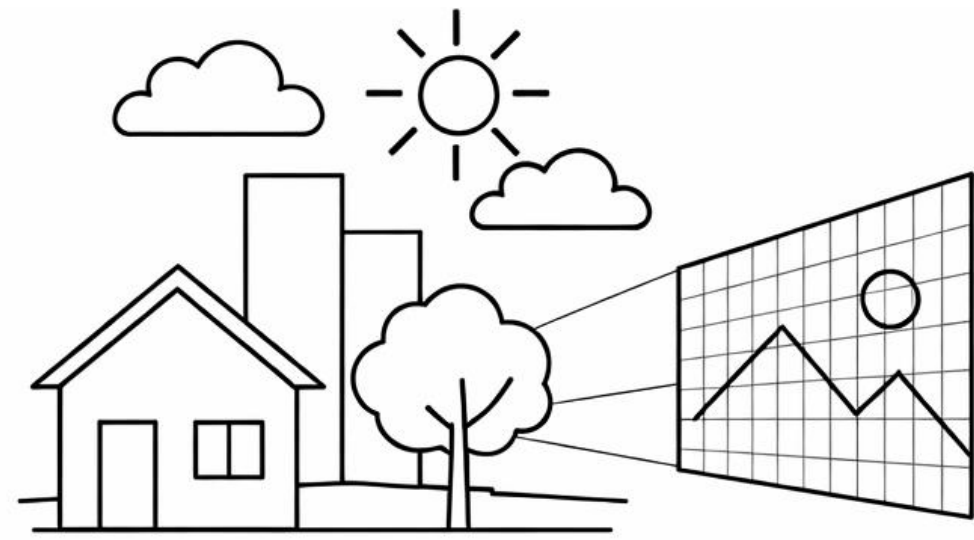


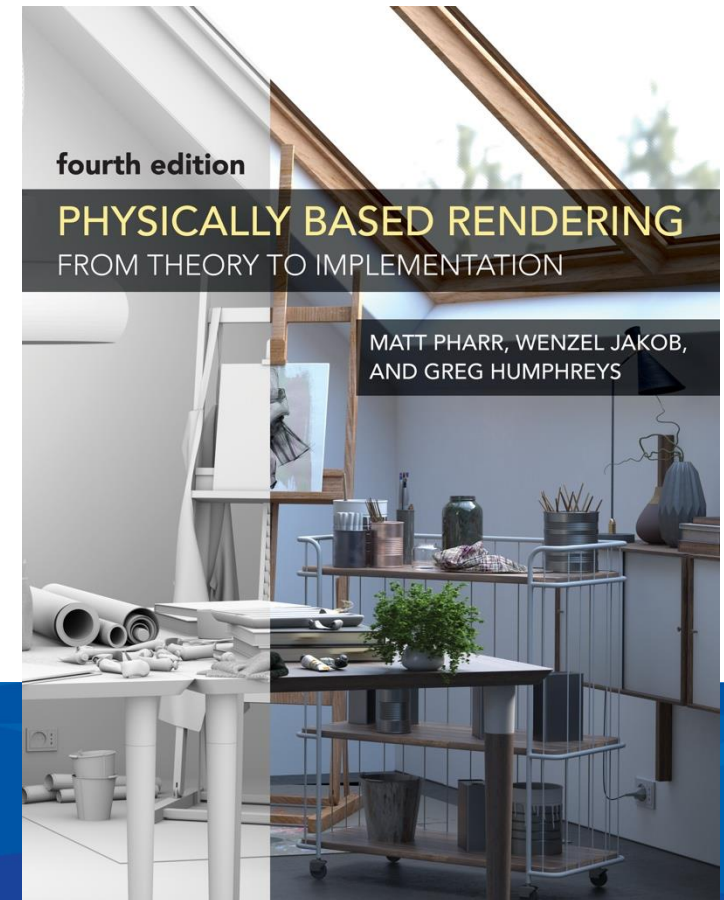
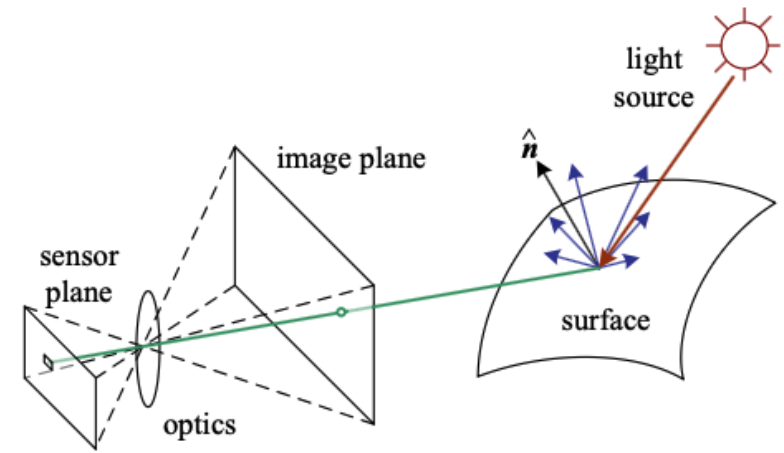
Illustration generated by ChatGPT. Is it good or not?



SOUTH DAKOTA  
STATE UNIVERSITY

# Photometric Image Formation?

- A simplified model of photometric image formation. Light is emitted by one or more light sources and is then reflected from an object's surface.
- A portion of this light is directed towards the camera. This simplified model ignores multiple reflections, which often occur in real-world scenes.
- Optional reading: physically based rendering (e.g., <https://pbrt.org>), if you are interested!



# Takeaway

- Image formation?
- Perspective projection / Coordinate transformation?
- Intrinsic / Extrinsic matrix?
- Projection matrix?
- Interactive demonstration of extrinsic parameters:  
<https://ksimek.github.io/2012/08/22/extrinsic/>

