

# Mathematical Foundations for Cameras in Computer Vision

Amir

Department of Computer Science

# Why Start With Mathematics?

- Computer vision operates on images
- Images come from cameras
- Cameras obey geometric and physical rules

**Mathematics lets us express these rules precisely.**

# What Goes Wrong Without Math?

- Same image can represent many 3D scenes
- Size, distance, and depth become ambiguous
- Algorithms fail silently

# Learning Philosophy of This Lecture

- We build math tools only when needed
- Every symbol will be explained
- Every equation will solve a camera problem

# Scalars: The Simplest Mathematical Objects

A **scalar** is a single number.

Examples:

- Distance
- Time
- Pixel intensity

# Why Scalars Are Not Enough

Cameras measure:

- Horizontal location
- Vertical location
- Depth

We need multiple numbers together.

# Coordinates Describe Position

A coordinate system answers:

*Where is this point?*

Example (2D):

$(x, y)$

# 2D Coordinates and Images

- Images live in 2D
- Pixel locations use  $(x, y)$

# 3D Coordinates and the Real World

- Real scenes are 3D
- Depth matters

$(x, y, z)$

# Vectors: Grouping Numbers

A **vector** stores multiple related values.

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$$

# Why Vectors Matter for Cameras

- A point is a vector
- A direction is a vector
- A camera ray is a vector

# Vector Addition (Motion)

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$$

Used for movement and translation.

# Matrices: Transforming Vectors

A **matrix** transforms vectors.

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

# Matrix–Vector Multiplication

$$\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \end{bmatrix} = \begin{bmatrix} 2 \\ 6 \end{bmatrix}$$

# Why This Matters

- Scaling
- Rotation
- Coordinate change

Cameras perform all three.

# What Does Linear Mean?

A transformation is linear if:

- Scaling inputs scales outputs
- Adding inputs adds outputs

# Why Linearity Is Powerful

- Predictable behavior
- Efficient computation
- Differentiable (learning-friendly)

# Rotation

Rotation changes orientation, not size.

$$R \in \mathbb{R}^{3 \times 3}$$

# Translation

Translation moves all points equally.

$$\mathbf{t} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

# World to Camera Transformation

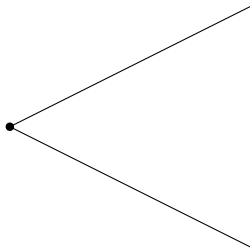
$$\mathbf{X}_c = R\mathbf{X}_w + \mathbf{t}$$

- $\mathbf{X}_w$ : world point
- $\mathbf{X}_c$ : camera point

# Why Projection Is Needed

- Images are 2D
- World is 3D

# Pinhole Camera Model



Based on geometric projection [?].

# Projection Equation

$$x = \frac{X}{Z}, \quad y = \frac{Y}{Z}$$

- $(X, Y, Z)$ : camera coordinates
- $(x, y)$ : normalized image coordinates

# Why Intrinsic Are Needed

- Pixels are discrete
- Sensors have scale and offset

# Intrinsic Matrix $K$

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

# What Is $K$ and Why Do We Care?

- $K$  maps camera coordinates to pixel coordinates
- Encodes camera internals
- Needed for calibration and measurement

# Meaning of Each Term in $K$

- $f_x, f_y$ : focal lengths (zoom)
- $c_x, c_y$ : principal point (image center)

# The Complete Camera Equation

$$\mathbf{x} = K[R \mid \mathbf{t}]\mathbf{X}$$

# What Each Symbol Represents

- $\mathbf{X}$ : 3D world point
- $R, \mathbf{t}$ : camera pose
- $K$ : camera internals
- $\mathbf{x}$ : image pixel

# Why Computer Vision Needs This

- Pose estimation
- 3D reconstruction
- Visual odometry

**Cameras turn geometry into images. Mathematics lets us reverse that process.**

# References I