Semester Gasal 2024/2025
Topic #8

Multi-Camera Vision Systems

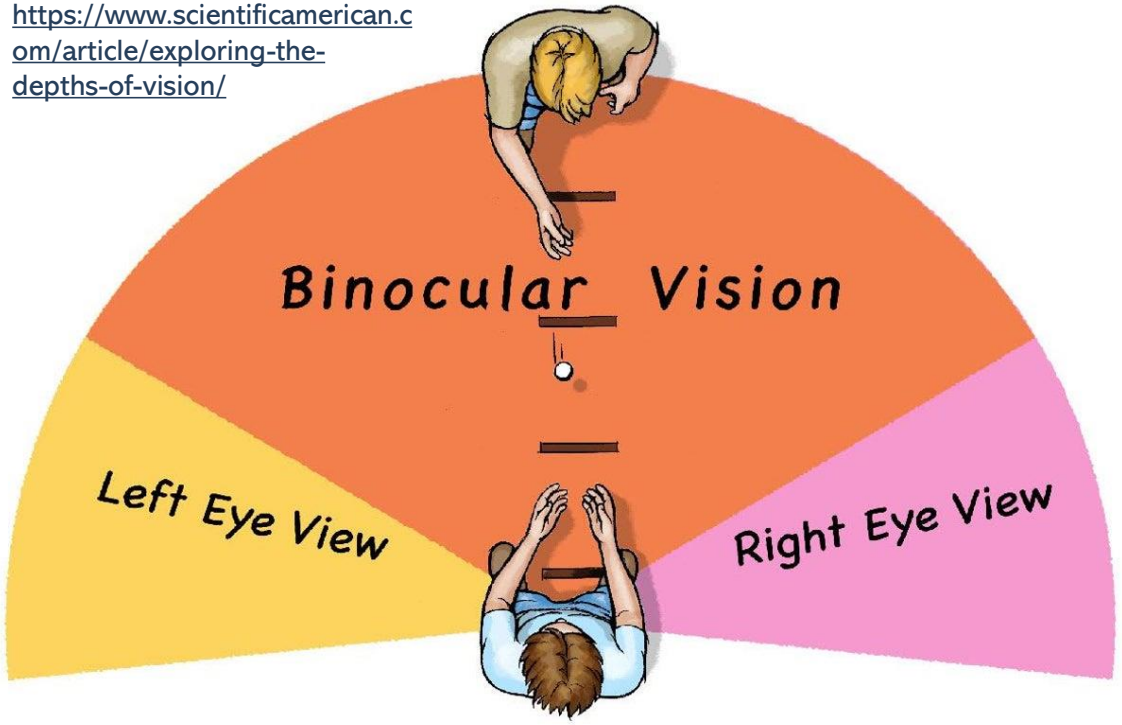
CSCE604133 Computer Vision
Faculty of Computer Science
Universitas Indonesia

Dr. Eng. Laksmita Rahadianti
Muhammad Febrian Rachmadi, Ph.D.
Dr. Dina Chahyati, Prof. Dr. Aniati M. Arymurthy

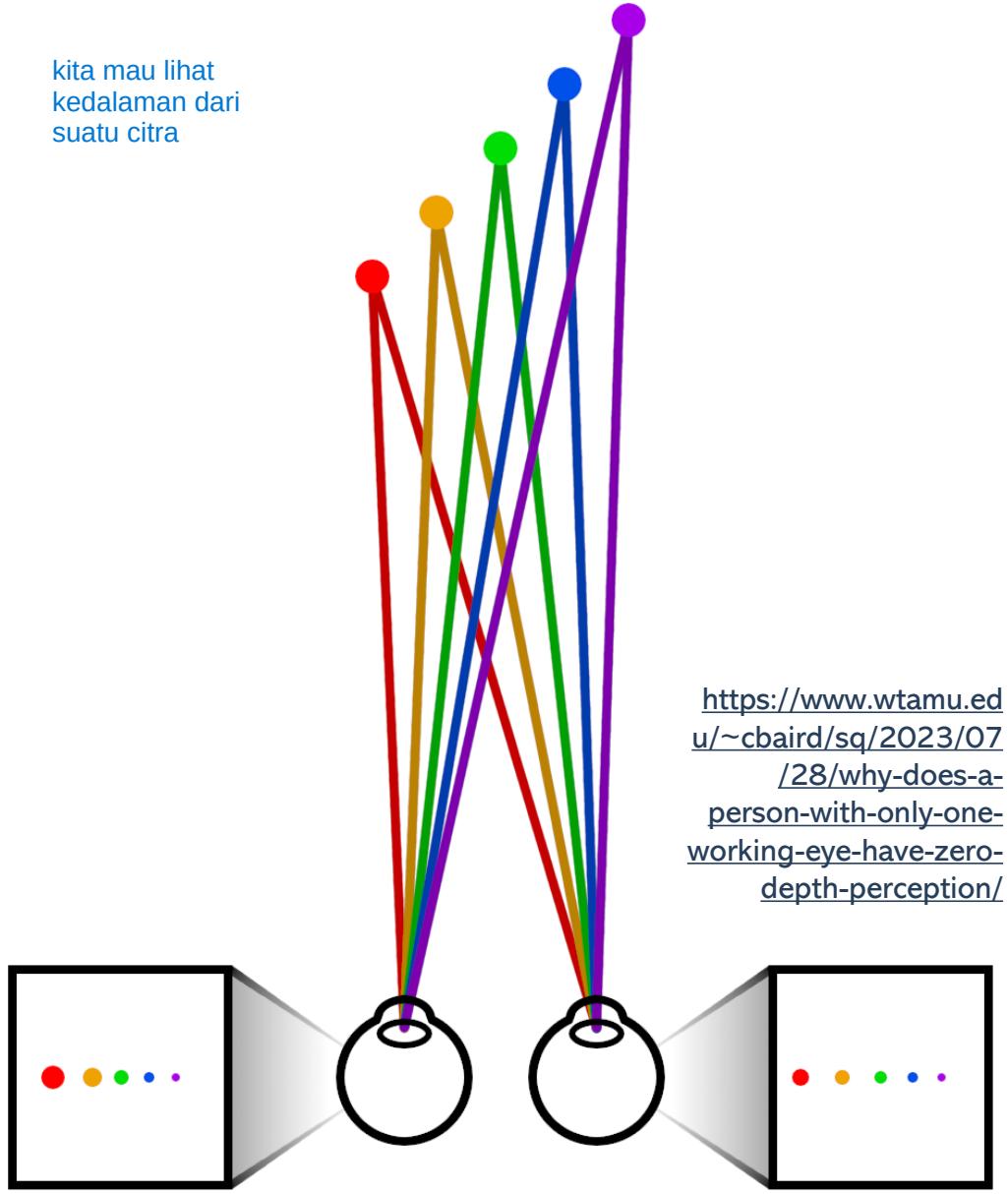
Motivation: The Perception of Depth (1)

- Q: How do humans perceive depth in our 3D world?
- A: Mostly because humans have two eyes!
- Binocular vision is the ability of the eyes to work together in coordination to perceive 3D image of surroundings.

https://www.scientificamerican.com/article/exploring-the-depths-of-vision/



kita mau lihat kedalaman dari suatu citra



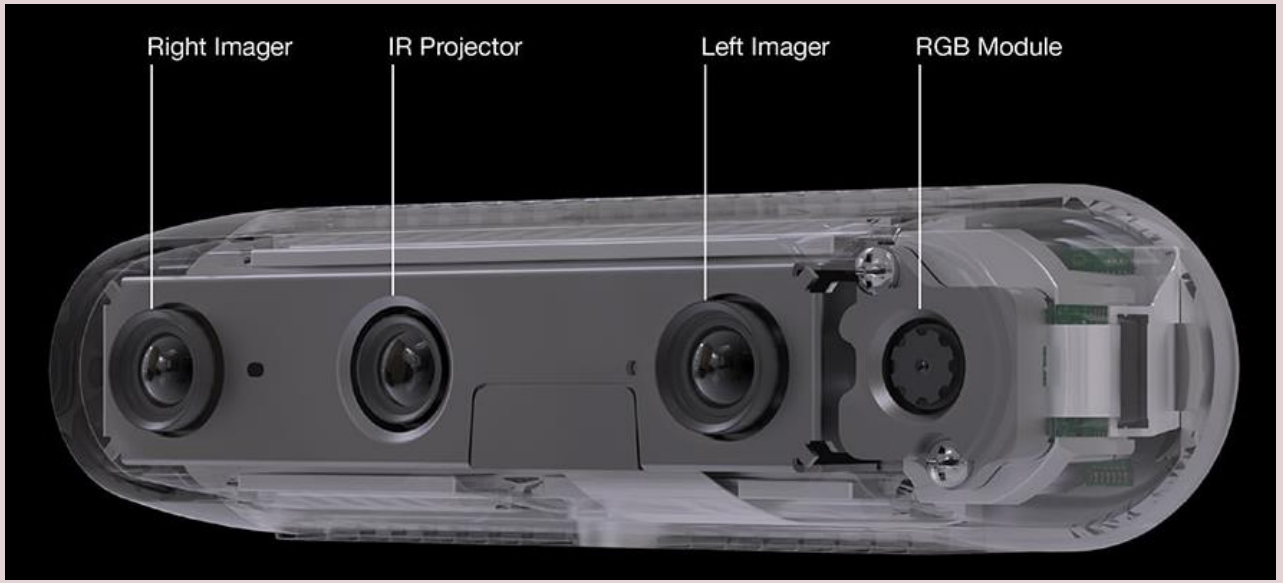
https://www.wtamu.edu/~cbaird/sq/2023/07/28/why-does-a-person-with-only-one-working-eye-have-zero-depth-perception/

Motivation: The Perception of Depth (2)

buat apa multi camera vision di CV? -> Misalnya kita mau buat 3D Body scanner (ada dibawah)

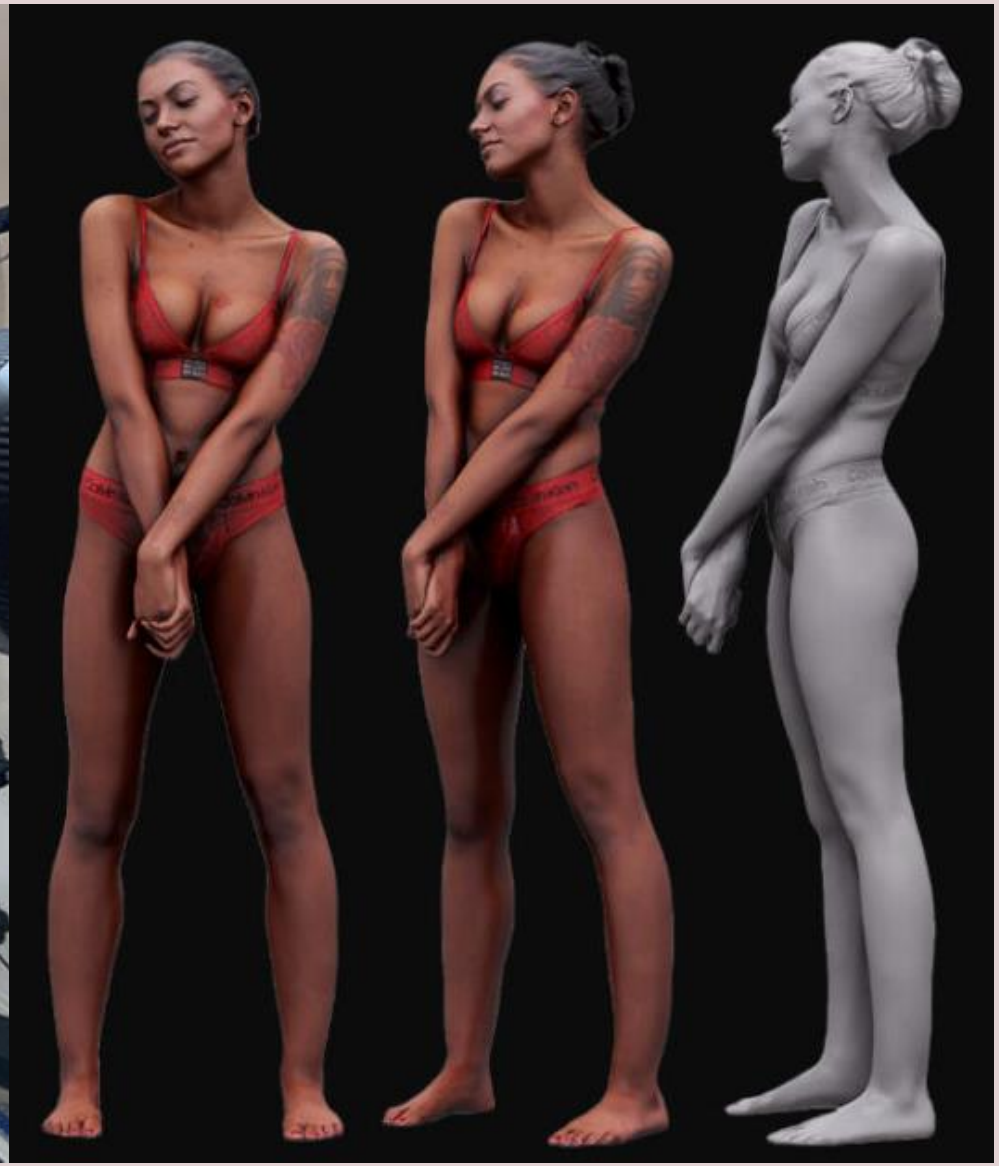
- We replicate this in computer vision!

https://rditechnologies.com/models/stereo-vision/



https://www.intelrealsense.com/depth-camera-d435/

Real World Example: 3D Body Scanner



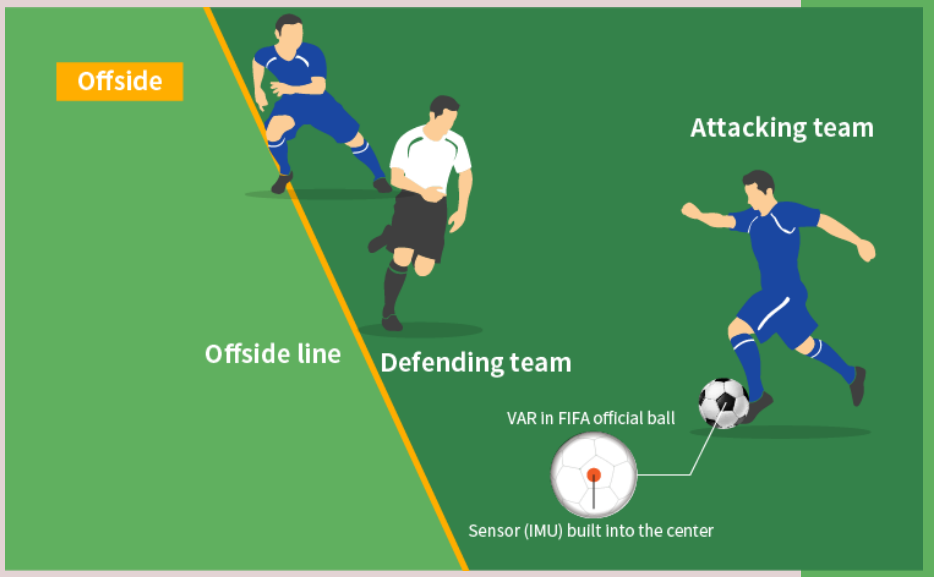
Keyword: Photogrammetry, i.e., the art and science of extracting 3D information from photographs.

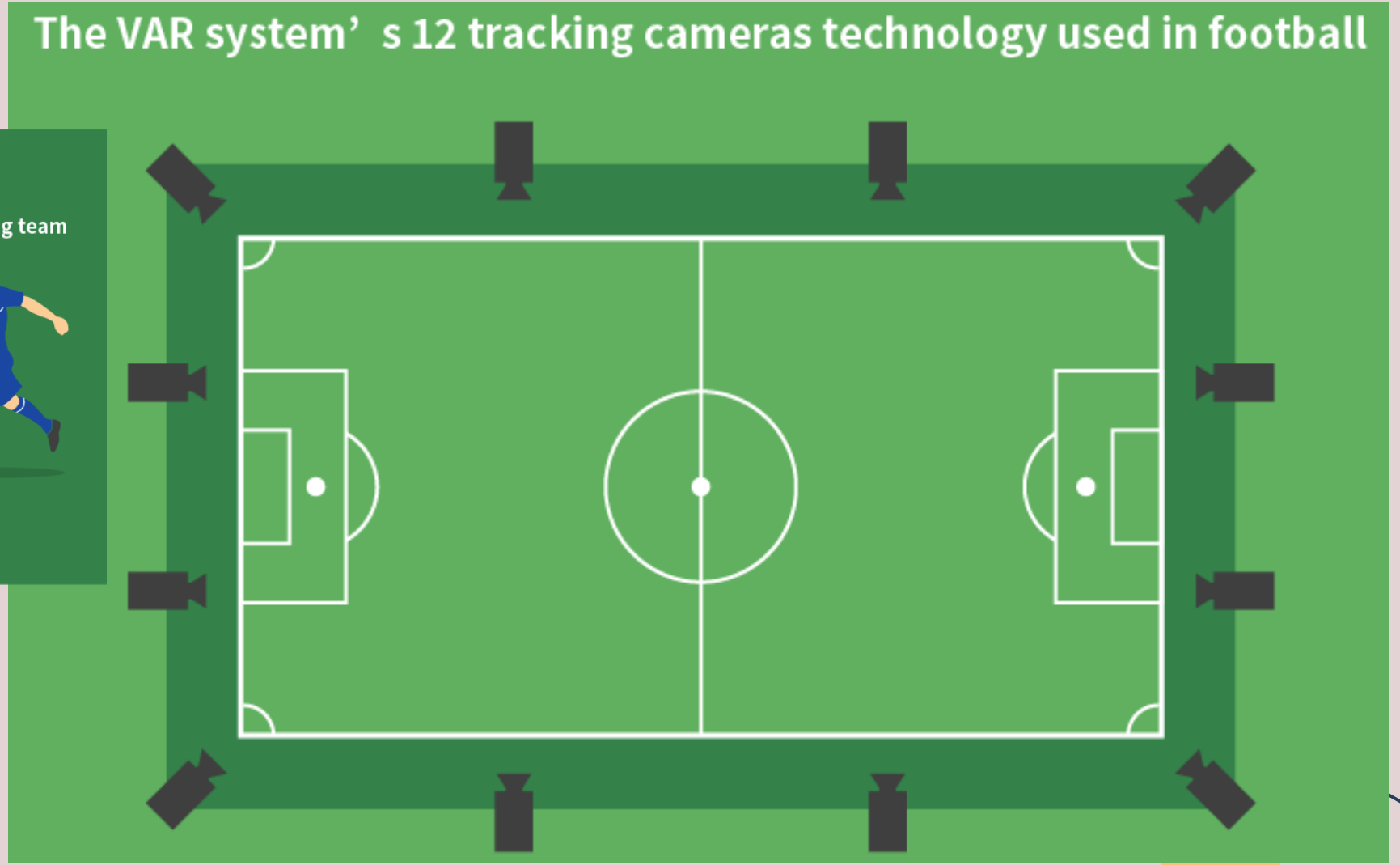
lihat keyword:

photogrammetry

https://www.scan3dbody.com/foto-rig/

Real World Example: Video Assistant Referee (VAR) in Football

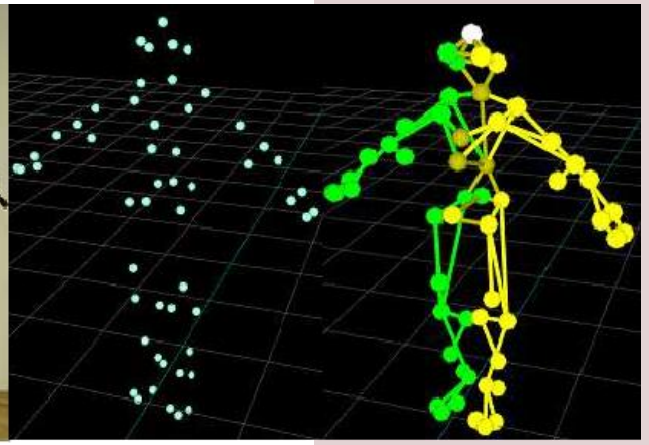




https://www.tdk.com/en/tech-mag/sportstech/002#:~:text=These%20video%20referee%20cameras%20track,are%20positioned%20around%20the%20field.

Real World Example: Motion Capture System

kiita menggunakan kayak marker, terus diatasnya ada namanya multi camera vision system. Jadi kalau misalnya modelnya bergerak, ktia bisa tahu animasinya kayak gimana



https://www.phasespace.com/impulse-motion-capture.html

Pawlyta, M., & Skurowski, P. (2016). A survey of selected machine learning methods for the segmentation of raw motion capture data into functional body mesh. In *Information Technologies in Medicine: 5th International Conference, ITIB 2016 Kamień Śląski, Poland, June 20-22, 2016 Proceedings, Volume 2* (pp. 321-336). Springer International Publishing.

Example: Motion Capture for Game Development of Assassin's Creed by Ubisoft

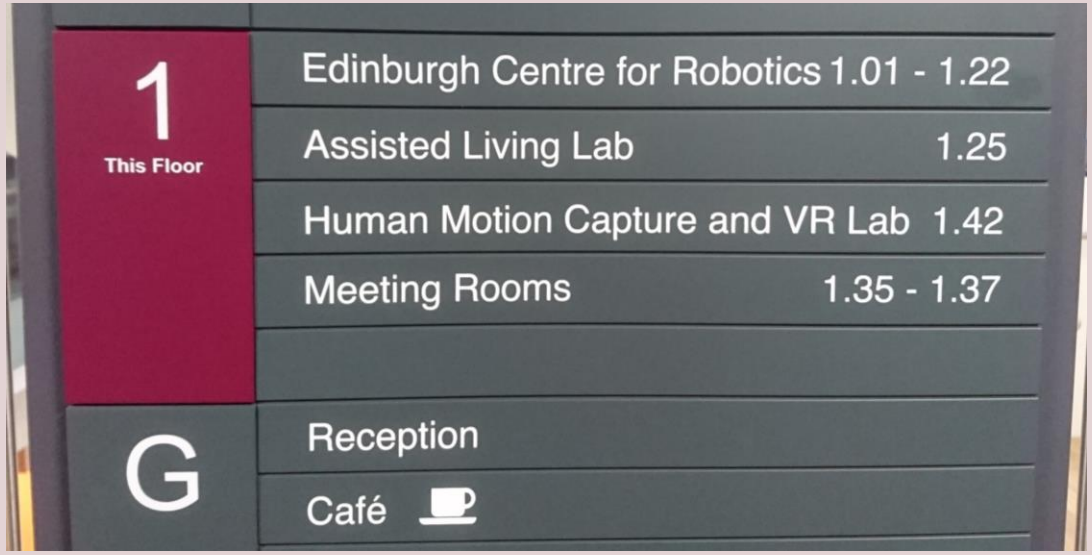
https://youtu.be/yQOH96pBlhg?si=sD2OQcar6TBFd_Wn

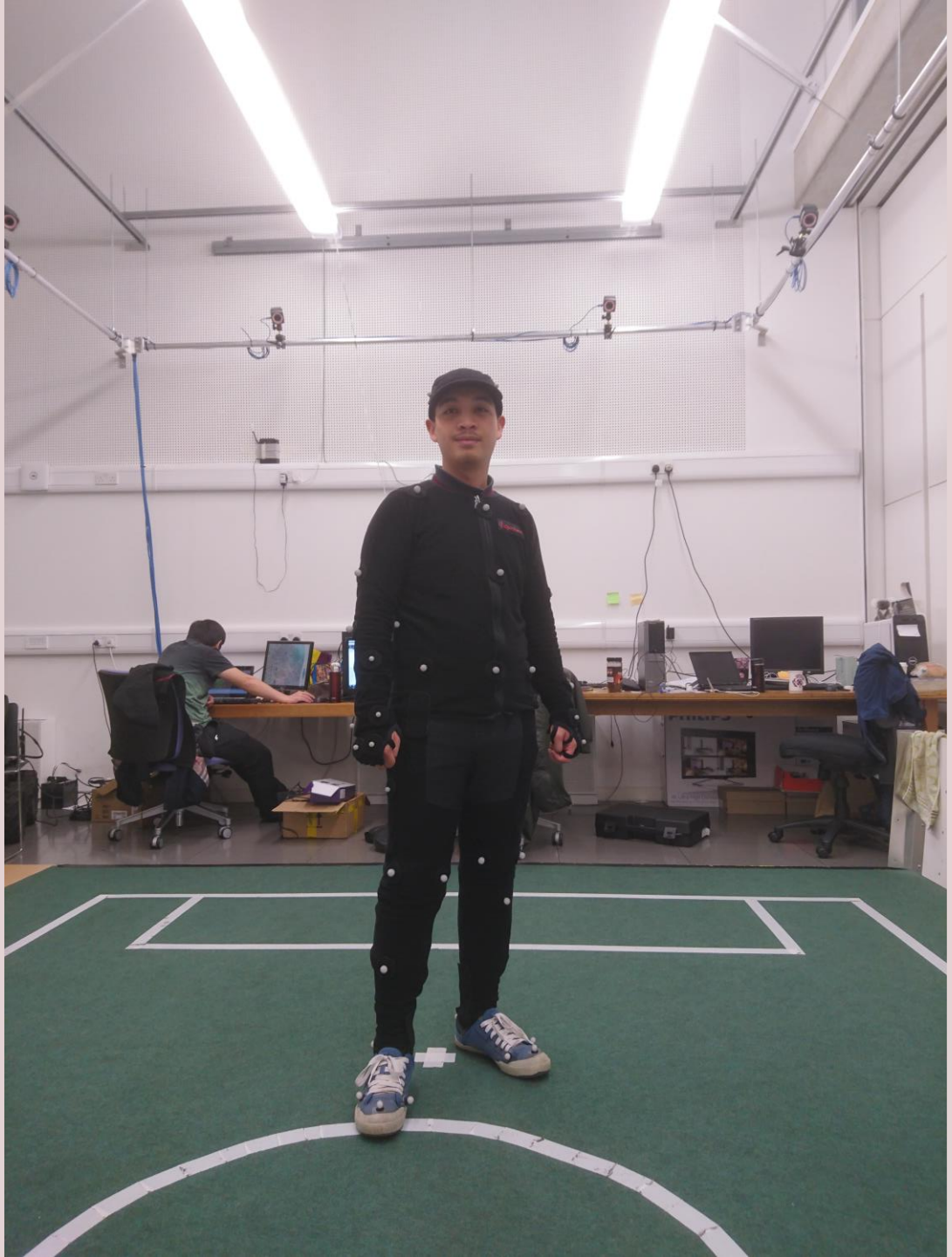
Example: Mocap of Cats for Video Game

https://youtu.be/zyq_LQrHpoo?si=lGms7BKVgOLj9bxG

MFR Acted as A Mocap Actor

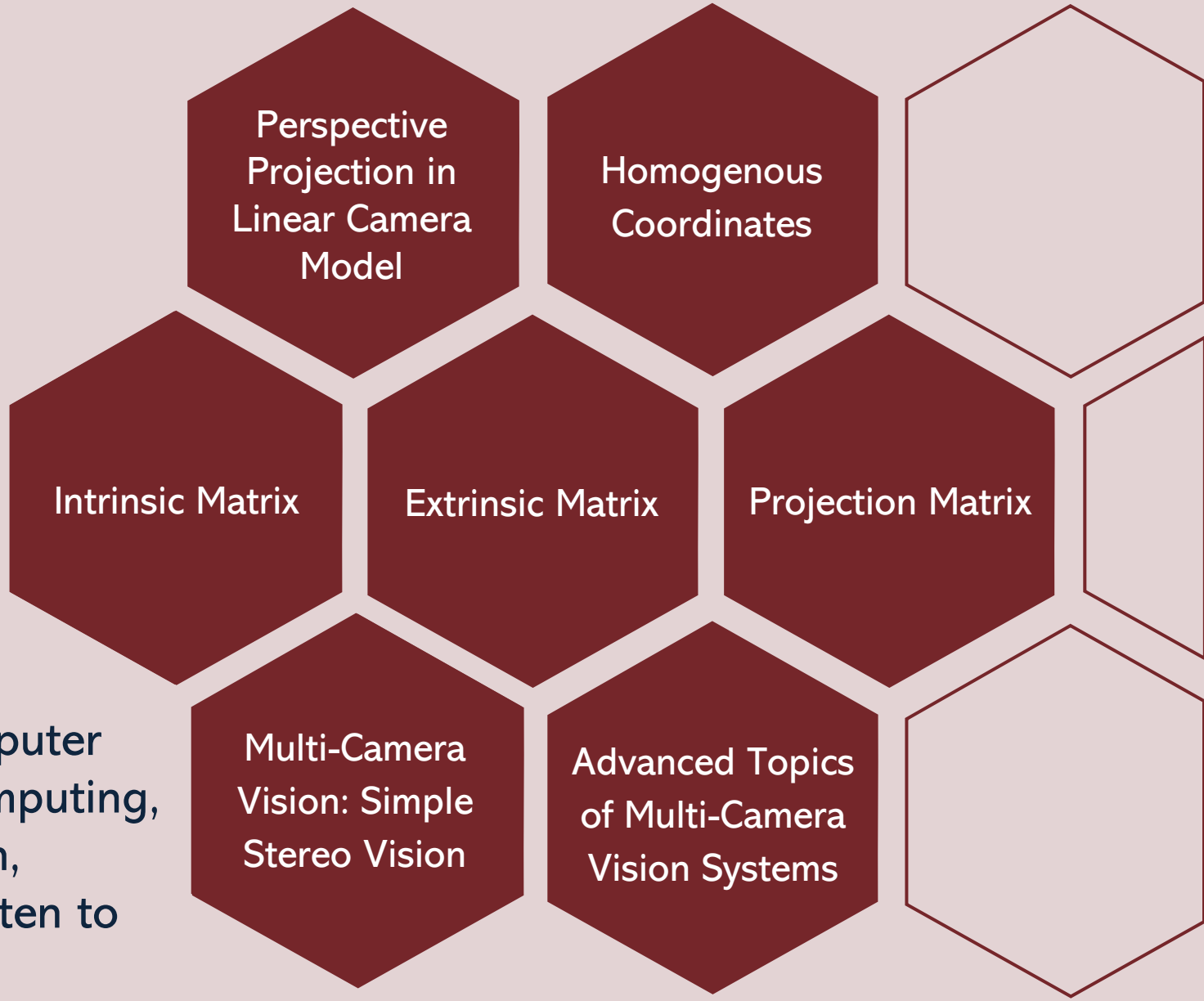
- October 2016, School of Informatics, the University of Edinburgh, UK

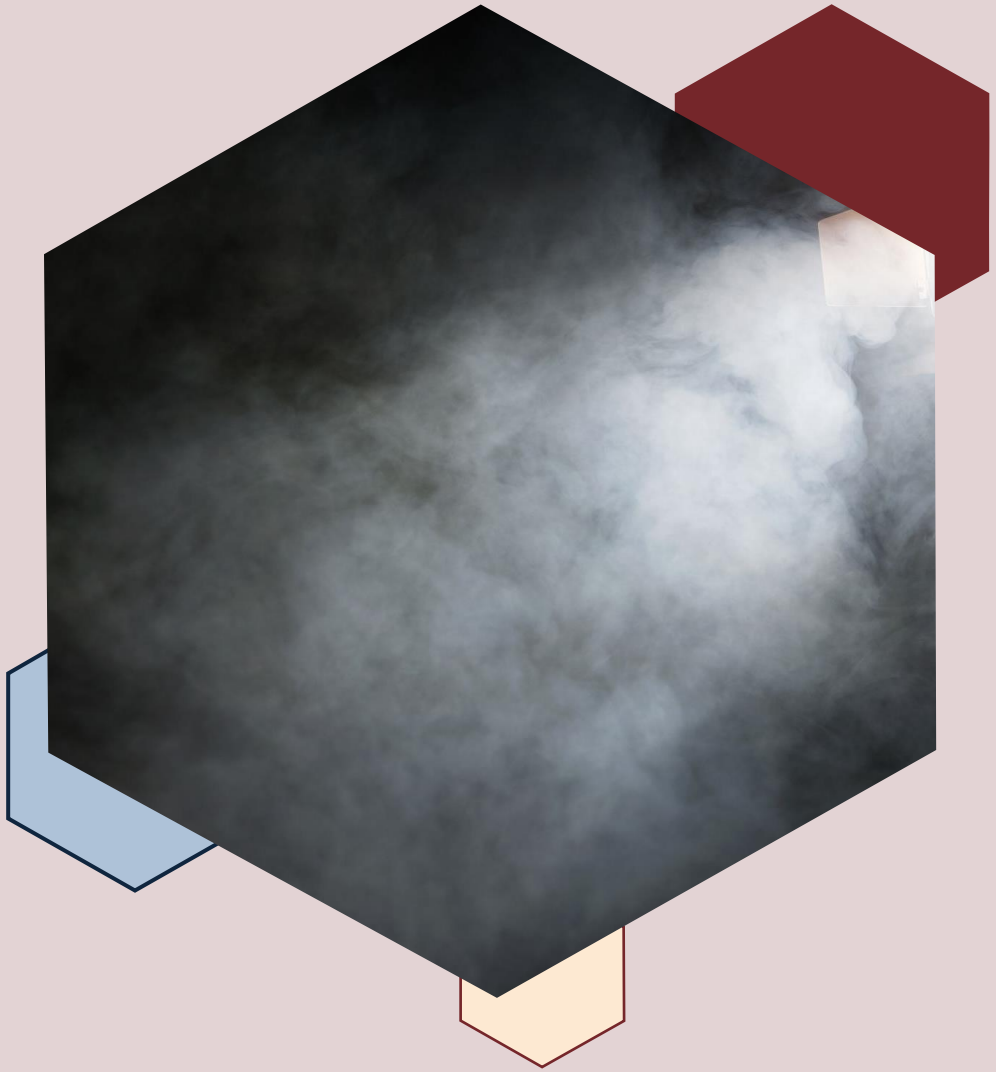




Agenda

- Warning: You will find a lot of math equations on this presentation!
- If you are interested in working on computer graphics, game development, visual computing, 3D animation, etc. after your graduation, we strongly suggest you to seriously listen to this lecture!



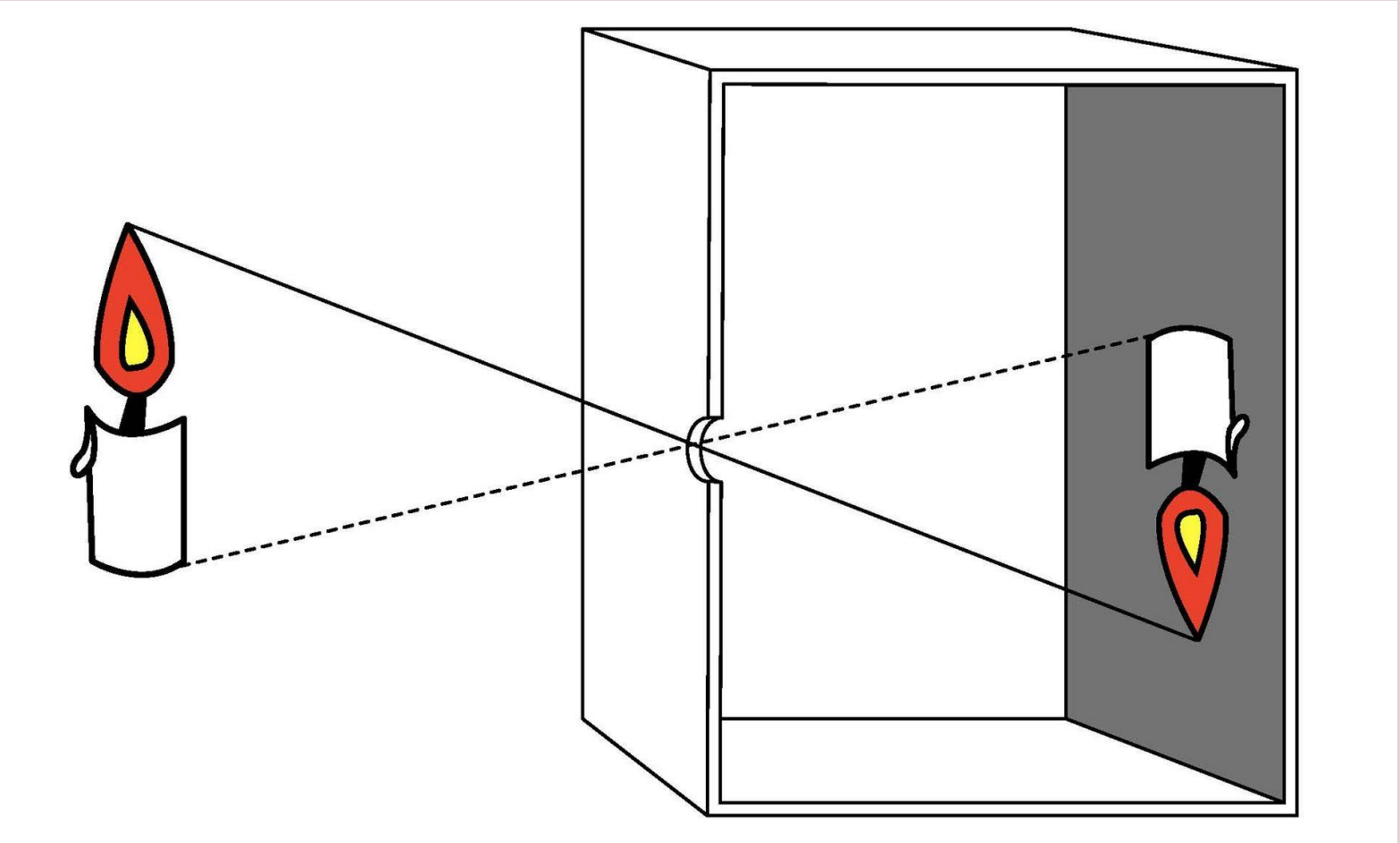


Perspective Projection in Linear Camera Model

Section 1

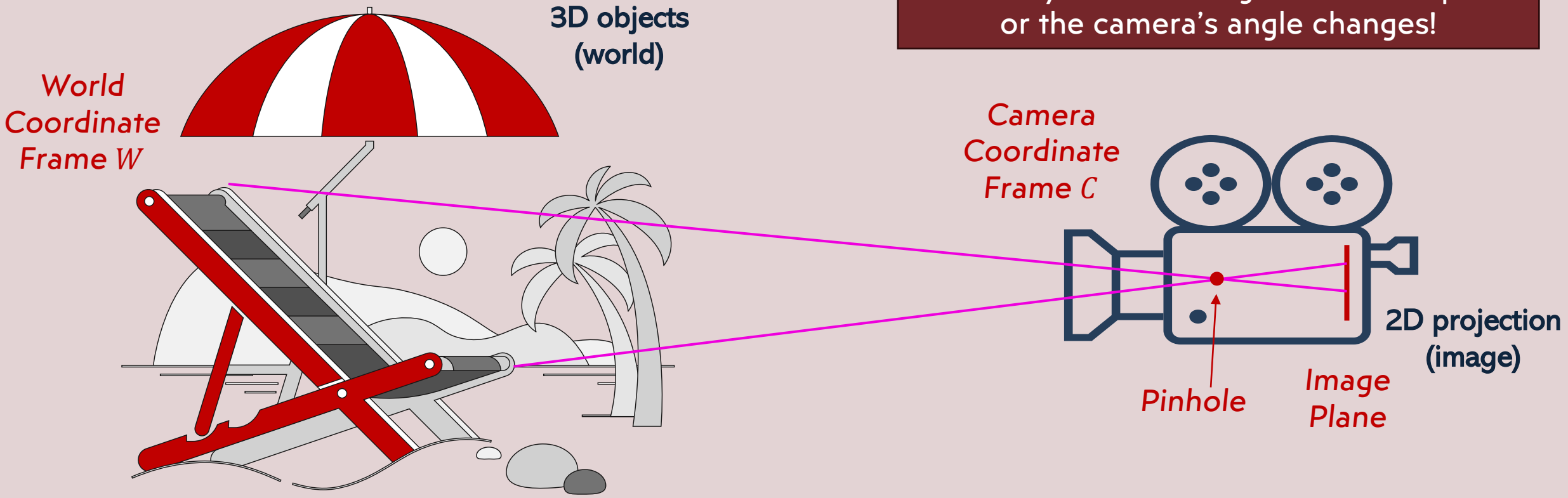
Background: The Pinhole Camera

- Camera obscura: The Latin translation of dark room.



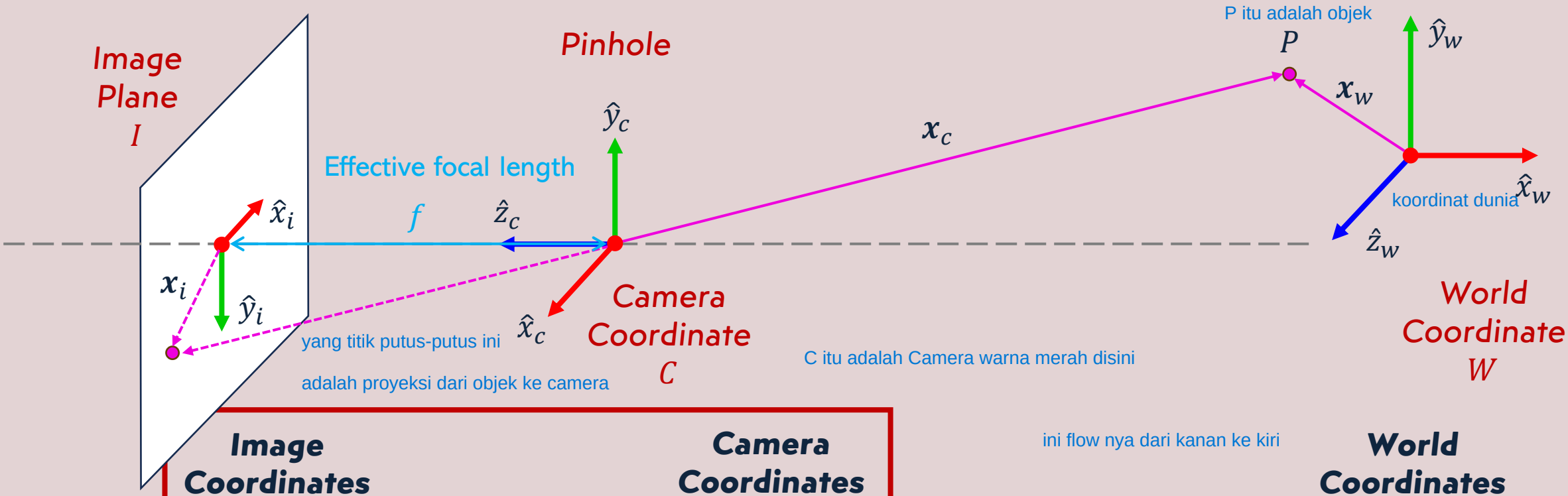
Forward Imaging Model: 3D to 2D

Notice how 3D world objects are projected differently on a 2D image when their poses or the camera's angle changes!



Adapted from: http://6.8300.csail.mit.edu/sp23/lectures/L15/2_single_view_geometry.pdf

Formulation of Forward Imaging: 3D to 2D



ini flow nya dari kanan ke kiri

Image Coordinates

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

?

Perspective Projection

Camera Coordinates

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

?

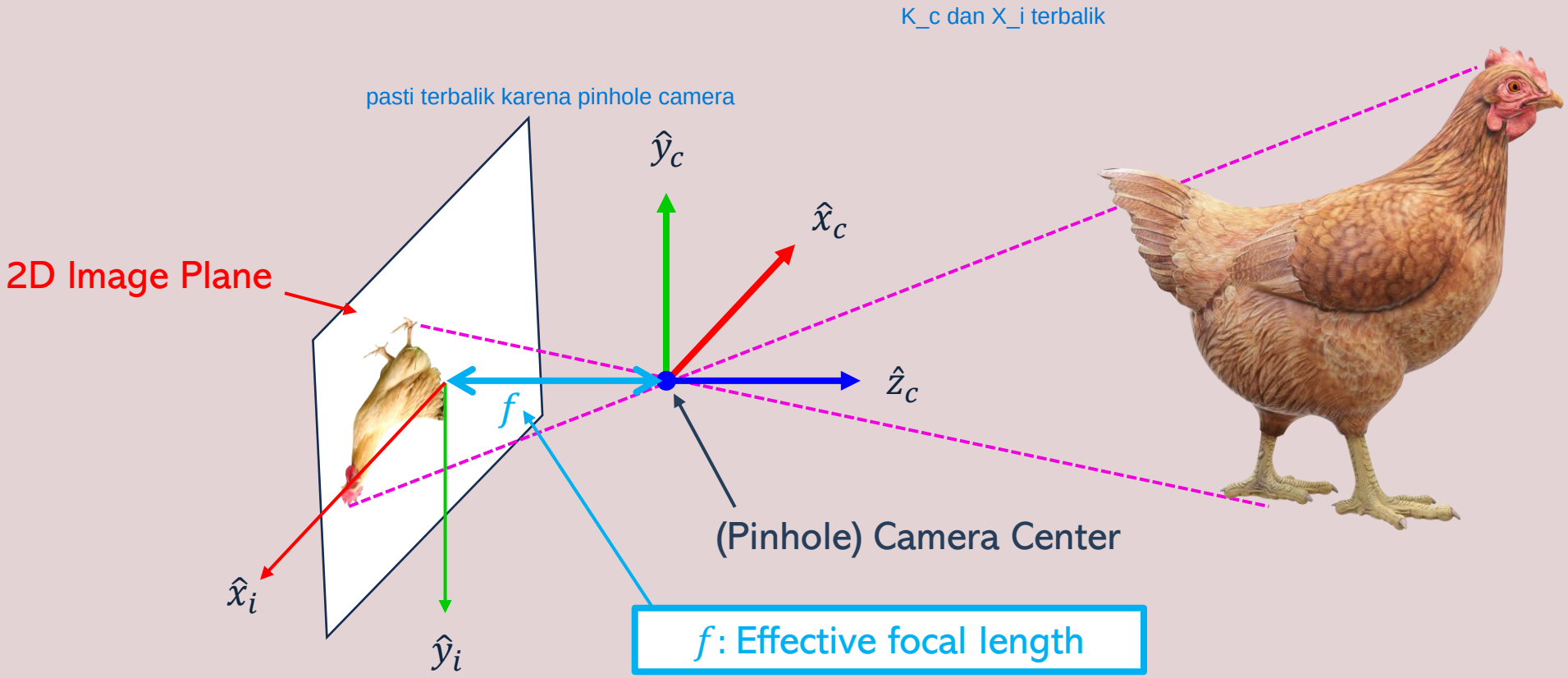
Coordinate Transformation

World Coordinates

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

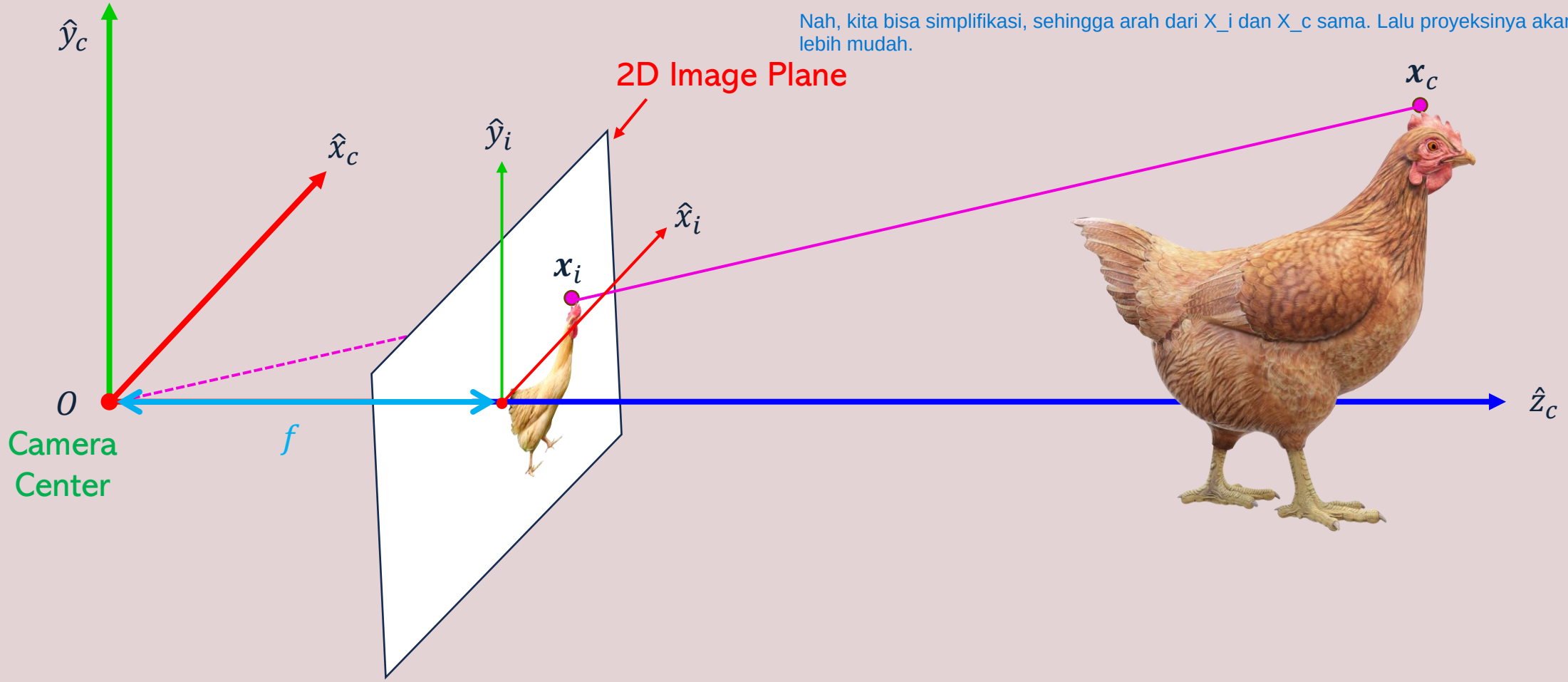
https://youtu.be/qByYk6JggQU?si=FRgEVexrzE8RSyeX

Perspective Projection



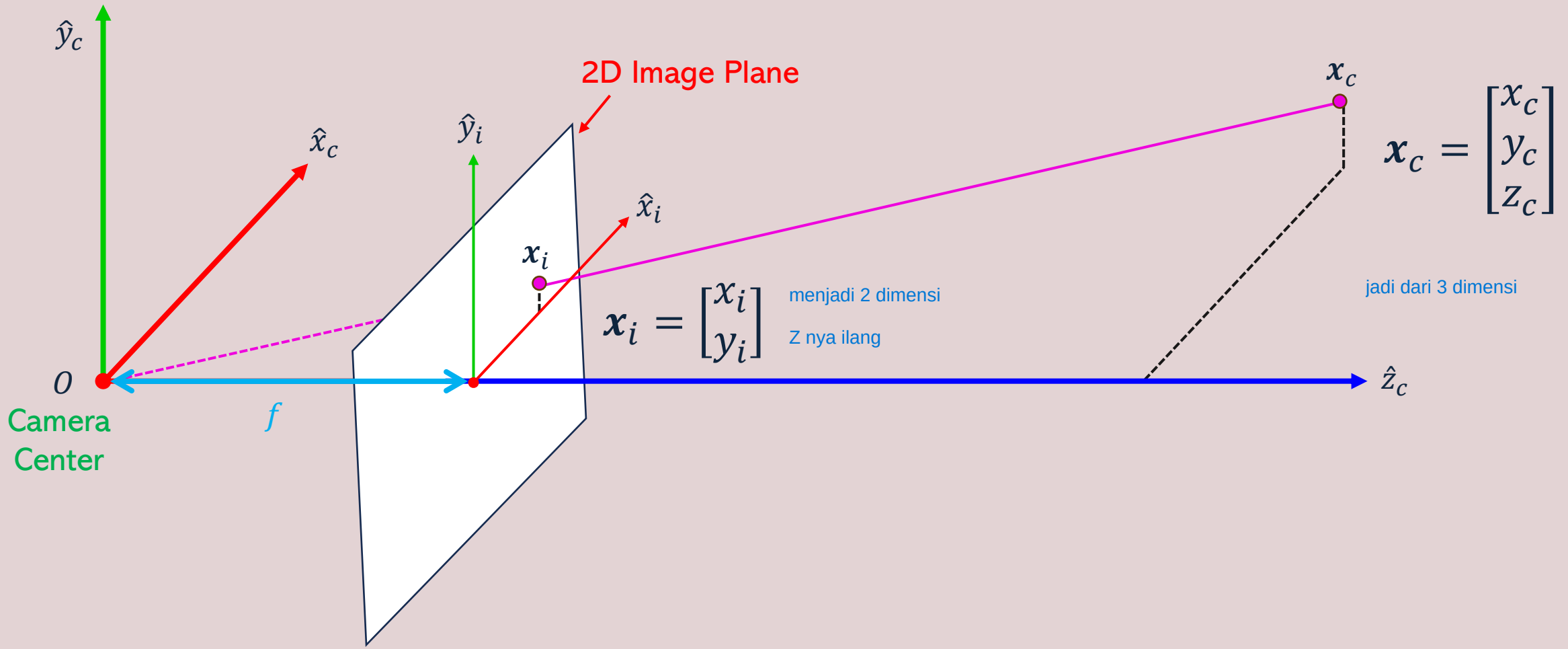
Perspective Projection (Simplified)

Nah, kita bisa simplifikasi, sehingga arah dari X_i dan X_c sama. Lalu proyeksinya akan menjadi lebih mudah.



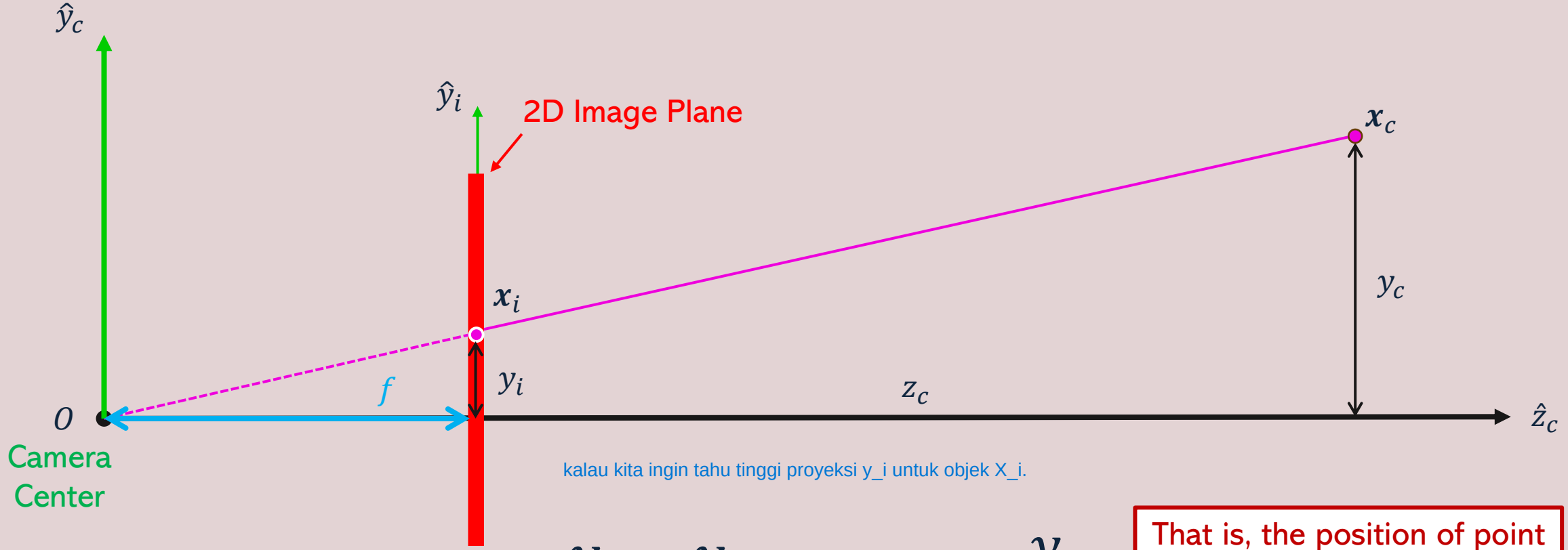
Adapted from: http://6.8300.csail.mit.edu/sp23/lectures/L15/2_single_view_geometry.pdf

Perspective Projection (Even More Simplified)



Adapted from: http://6.8300.csail.mit.edu/sp23/lectures/L15/2_single_view_geometry.pdf

Perspective Projection of y_c on a 2D Image



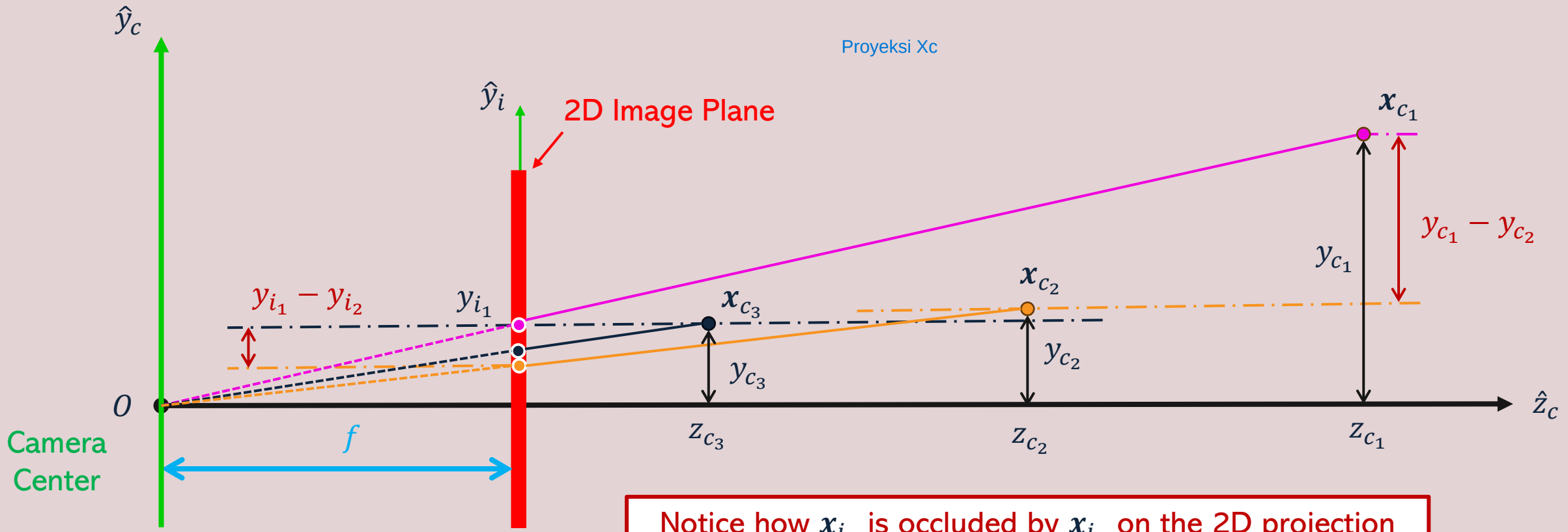
kalau kita ingin tahu tinggi proyeksi y_i untuk objek X_i.

f: Effective focal length

$$\frac{y_i}{f} = \frac{y_c}{z_c} \Rightarrow y_i = f\frac{y_c}{z_c}$$

That is, the position of point y_c of the world coordinate on the image is y_i, which depends on the f and z_c.

Perspective Projection of Multiple Objects



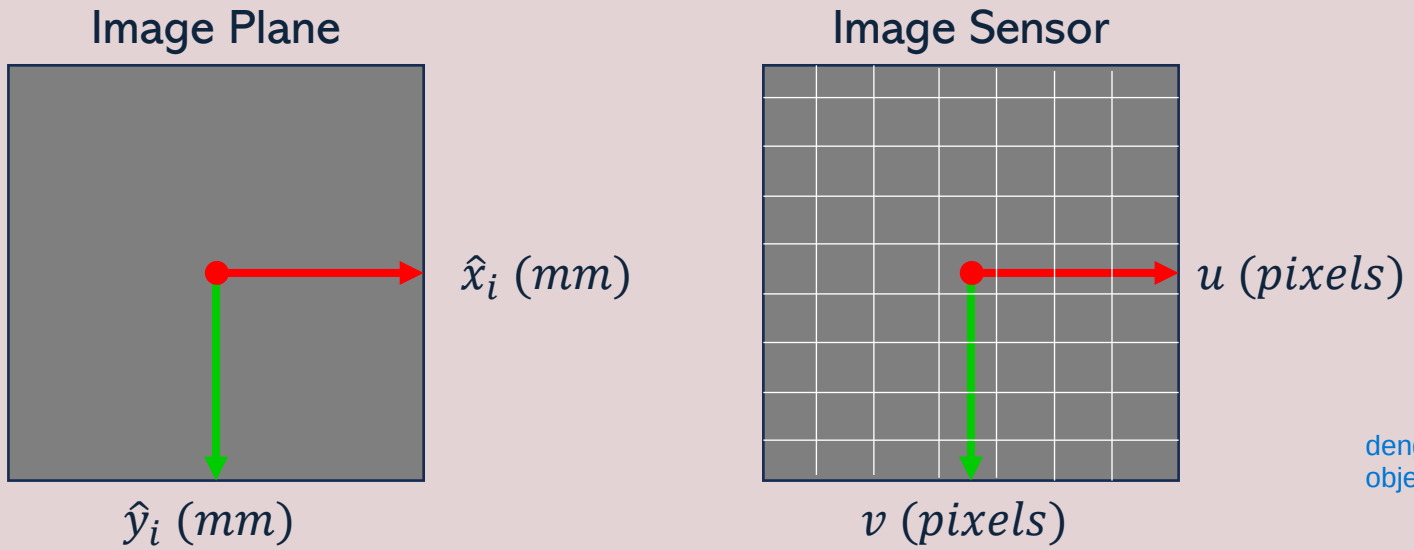
f: Effective focal length

Notice how x_{i_2} is occluded by x_{i_3} on the 2D projection (i.e., $y_{i_2} > y_{i_3}$ on the image plane) even though $y_{c_2} > y_{c_3}$ in the camera coordinate! This is called "magnification", which can be calculated using $m = {}^{f}/_{z}$.

Adapted from: http://6.8300.csail.mit.edu/sp23/lectures/L15/2_single_view_geometry.pdf

Magnification Due to Perspective Projection

Image Plane to Image Sensor Mapping

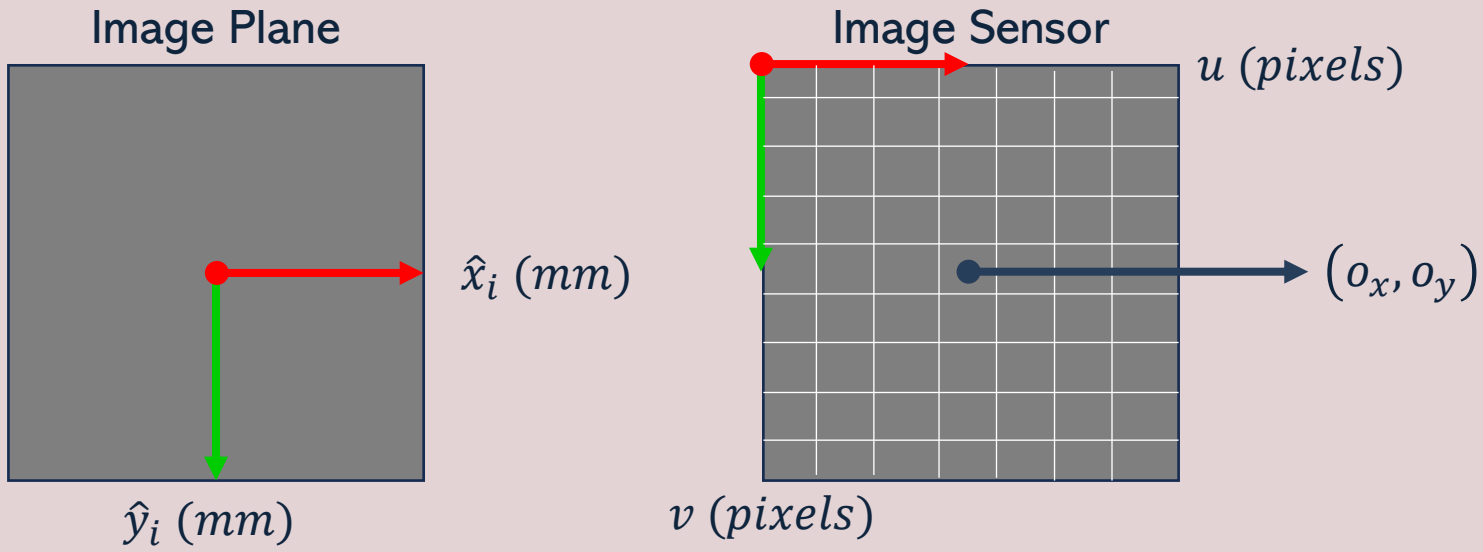


dengan mengalikan pixel density, ktia bisa menghitung letak dari pixel pada objek yang kita inginkan.

- Note that pixels might not be square (i.e., rectangular).
- If m_x and m_y are the pixel densities (pixels/mm) in x and y directions, respectively, then pixel coordinates are:

$$u = m_x x_i = m_x f \frac{x_c}{z_c} \qquad v = m_y y_i = m_y f \frac{y_c}{z_c}$$

The Origin Point of the Image Sensor



harus ada translation, karena awal-awal ditengah, mau ditaro di corner atas

- We usually treat the top-left corner of the image sensor as its origin due to easier for indexing the pixels.
- If pixel (o_x, o_y) 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: Intrinsic Parameters

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

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

- The terms of $m_x f$ and $m_y f$ can be written as f_x and f_y.

kita bisa dapetkan proyeksi pada pixel untuk objek Y. Adalah operasi penjumlahan (transformasi + translasi)

- Notice the terms $\frac{x_c}{z_c}$ and $\frac{y_c}{z_c}$ are based on the location of 3D objects in the real world, while f_x, f_y, o_x, and o_y are based on the technicality of the camera being used!

- Thus, the set of (f_x, f_y, o_x, o_y) is called intrinsic parameters of the camera, which represents the camera's internal geometry.

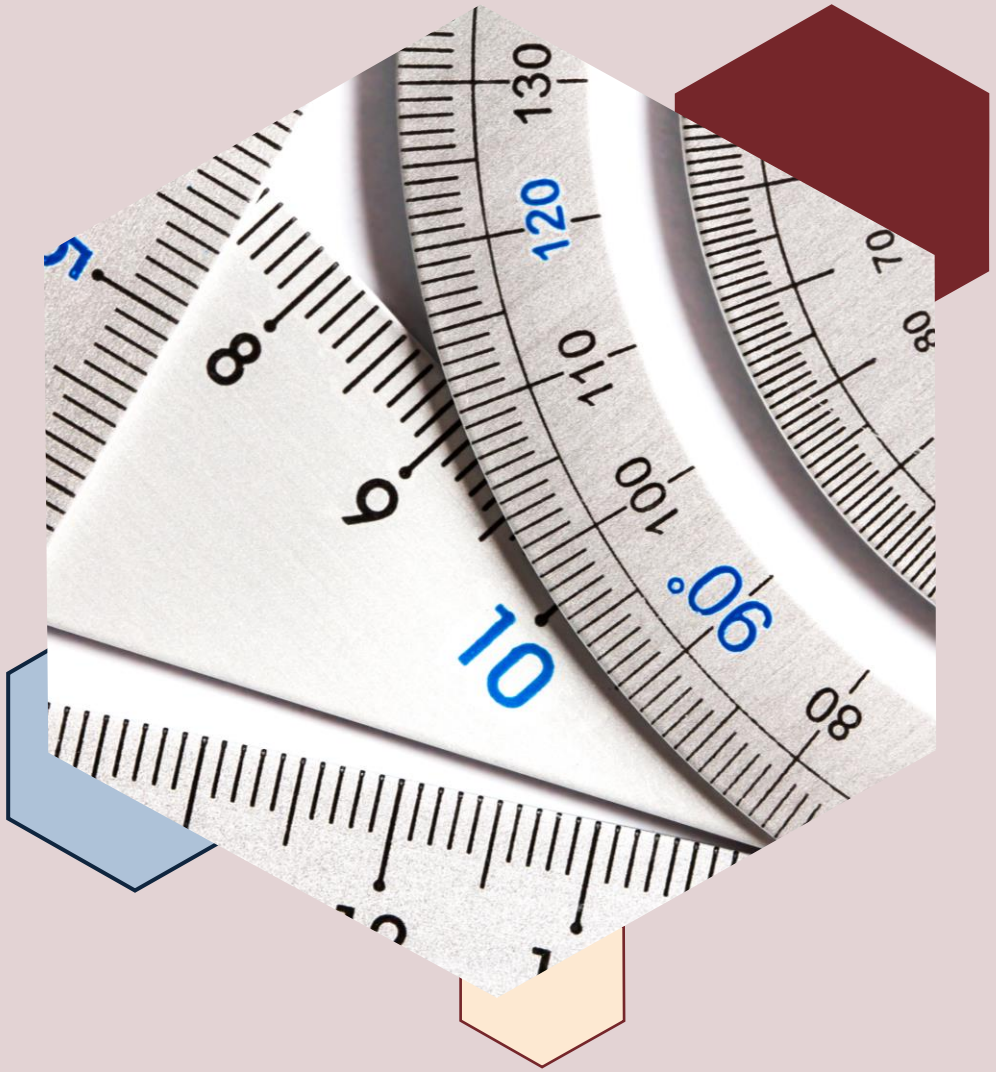
Preserved vs. Non-Preserved Information

- Preserved information?
  - Straight lines
- Non-preserved information?
  - Angles, lengths

Keyword:
Vanishing point

https://en.wikipedia.org/wiki/Vanishing_point

Adapted from: http://6.8300.csail.mit.edu/sp23/lectures/L15/2_single_view_geometry.pdf



Homogenous Coordinates

Section 2

Recall: Perspective Projection

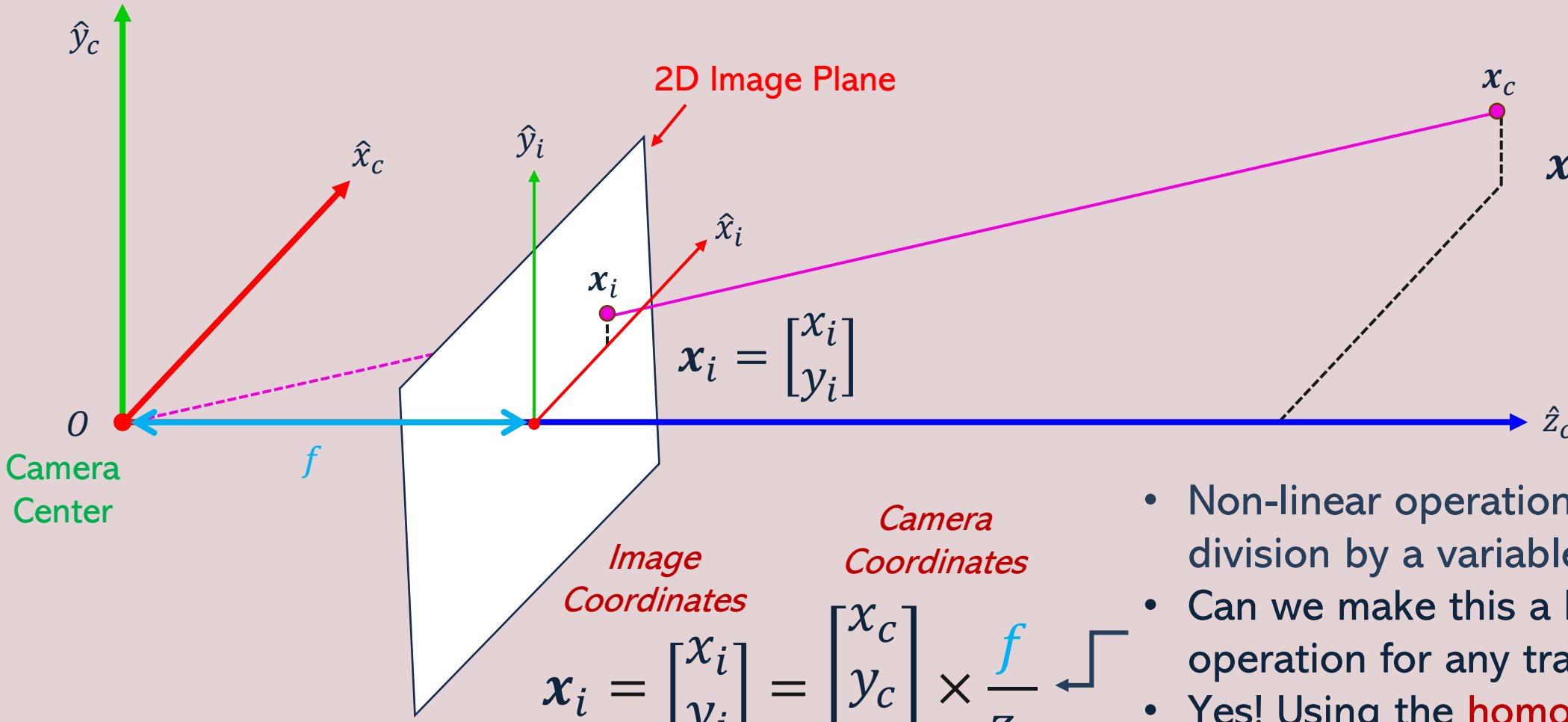


Image Coordinates $x_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \times \frac{f}{z_c}$ *Camera Coordinates*

- Non-linear operation due to division by a variable z_c.
- Can we make this a linear operation for any transformations?
- Yes! Using the homogeneous coordinate system!

Adapted from: http://6.8300.csail.mit.edu/sp23/lectures/L15/2_single_view_geometry.pdf

Homogenous Coordinates Construction

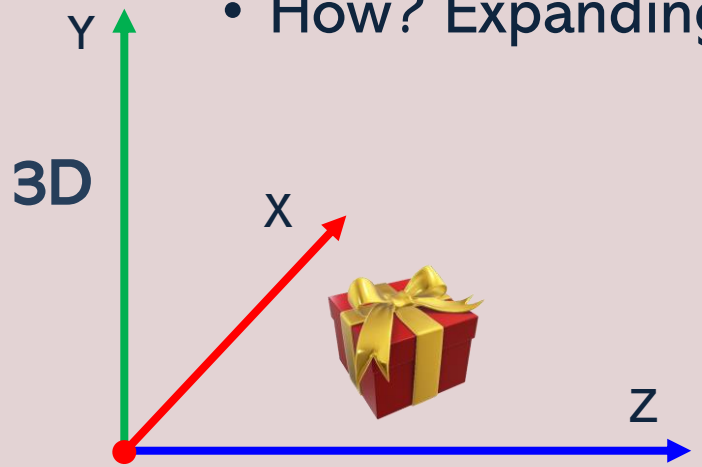
- Homogenous coordinates are required to have a uniform representation of affine transformation for 2D projection of 3D objects (e.g., rotation, translation, scaling, etc.).

$$\mathbf{x} = (x, y) \quad \Rightarrow \quad \tilde{\mathbf{x}} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$\mathbb{R}^2$ Heterogeneous coordinates

Homogenous coordinates $\mathbb{P}^2$

- How? Expanding the dimension!



$$\mathbf{X} = (X, Y, Z) \quad \Rightarrow \quad \tilde{\mathbf{X}} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

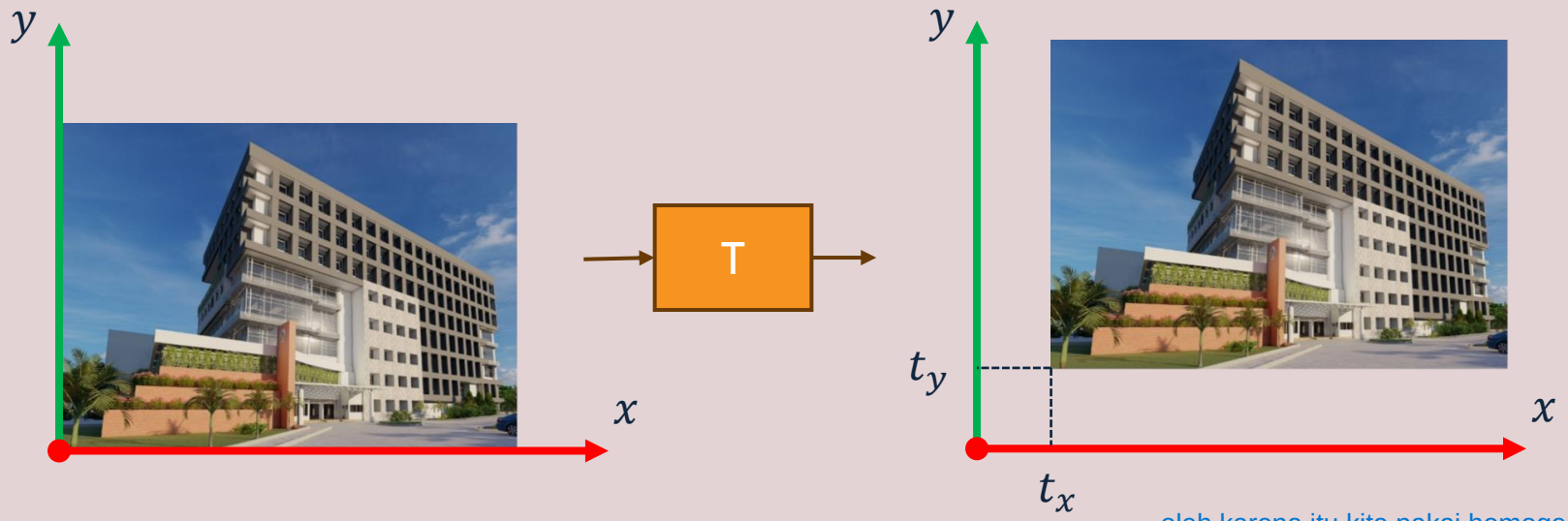
$\mathbb{R}^3$ Heterogeneous coordinates

Homogenous coordinates $\mathbb{P}^3$

Adapted from: http://6.8300.csail.mit.edu/sp23/lectures/L15/2_single_view_geometry.pdf

Why Homogenous Coordinates?

- Linearity of various image transformations.
- Example: Image translation below cannot be expressed as a (2×2) matrix operation in the heterogenous coordinates.



translasi

$$x_2 = x_1 + t_x$$

$$y_2 = y_1 + t_y$$

kita perlu cari perkalian matrix dimana bisa menghasilkan itu

$$\begin{bmatrix} ? & ? \\ ? & ? \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} x_1 + t_x \\ y_1 + t_y \end{bmatrix}$$

cuman kalo kayak gini kan jelek approachnya (boros)

oleh karena itu kita pakai homogenous coordinates

- On the other hand, we can express image translation as a (3×3) matrix operation in homogenous coordinates!

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} x_1 + t_x \\ y_1 + t_y \\ 1 \end{bmatrix} = \begin{bmatrix} \tilde{x}_2 \\ \tilde{y}_2 \\ \tilde{z}_2 \end{bmatrix} \equiv \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

Scaling, Rotation, Skew, and Translation in Homogenous Coordinates

$$\begin{bmatrix}\tilde{x}_2\\ \tilde{y}_2\\ \tilde{z}_2\end{bmatrix} = \begin{bmatrix}s_x & 0 & 0\\ 0 & s_y & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}x_1\\ y_1\\ 1\end{bmatrix}$$

Scaling

$$\begin{bmatrix}\tilde{x}_2\\ \tilde{y}_2\\ \tilde{z}_2\end{bmatrix} = \begin{bmatrix}1 & m_x & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}x_1\\ y_1\\ 1\end{bmatrix}$$

Skew

$$\begin{bmatrix}\tilde{x}_2\\ \tilde{y}_2\\ \tilde{z}_2\end{bmatrix} = \begin{bmatrix}1 & 0 & t_x\\ 0 & 1 & t_y\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}x_1\\ y_1\\ 1\end{bmatrix}$$

Translation

$$\begin{bmatrix}\tilde{x}_2\\ \tilde{y}_2\\ \tilde{z}_2\end{bmatrix} = \begin{bmatrix}\cos\theta & -\sin\theta & 0\\ \sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}x_1\\ y_1\\ 1\end{bmatrix}$$

Rotation

Composition of these transformations?

Affine Transformation

- Any transformation of the form:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{y}_1 \\ \tilde{z}_1 \end{bmatrix} = \begin{bmatrix} \tilde{x}_2 \\ \tilde{y}_2 \\ \tilde{z}_2 \end{bmatrix} \equiv \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

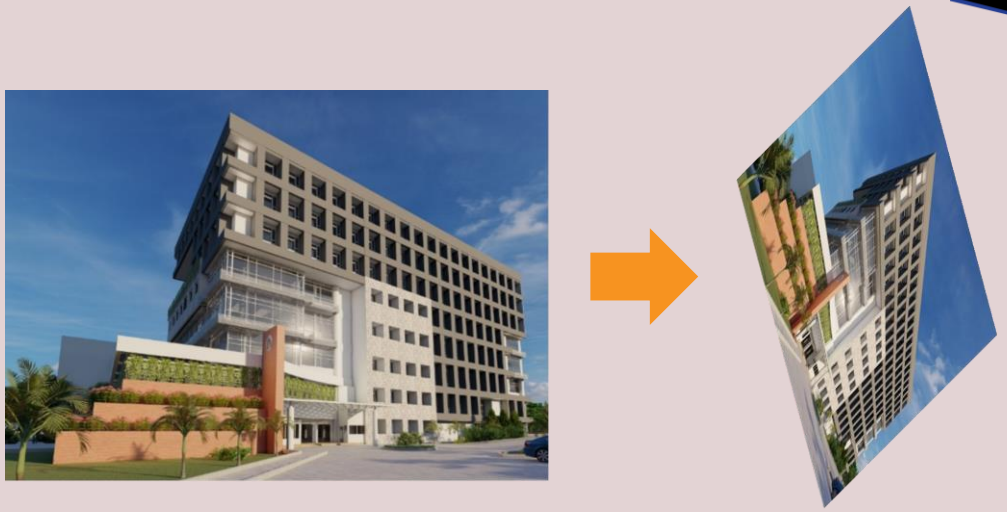


image yang udah di rotasi, terus di skew lagi

- The properties of affine transformation:
  - Origin does not necessarily map to the origin
  - Lines map to lines
  - Parallel lines remain parallel
  - Closed under composition

Heterogeneous ↔ Homogeneous (2D)

- **From heterogeneous to homogeneous:**

$$\underset{\mathbb{R}^2}{\mathbf{x} = (x, y)} \Rightarrow \mathbf{x} \equiv \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{w}.x \\ \tilde{w}.y \\ \tilde{w}.1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{bmatrix} = \underset{\mathbb{P}^2}{\tilde{\mathbf{x}}}$$

- **From homogeneous to heterogeneous:**

$$\underset{\mathbb{P}^2}{\tilde{\mathbf{x}} = \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{bmatrix}} \Rightarrow \tilde{\mathbf{x}} \equiv \left({}^{\tilde{x}}/_{\tilde{w}}, {}^{\tilde{y}}/_{\tilde{w}} \right) \equiv (x, y) = \underset{\mathbb{R}^2}{\mathbf{x}}$$

The third coordinate $\tilde{w} \neq 0$ is "fictious" to construct the homogeneous coordinate.

Adapted from: http://6.8300.csail.mit.edu/sp23/lectures/L15/2_single_view_geometry.pdf

Heterogeneous ↔ Homogeneous (3D)

- **From heterogeneous to homogeneous:**

$$\underset{\mathbb{R}^3}{\boldsymbol{X} = (X, Y, Z)} \Rightarrow \boldsymbol{X} \equiv \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{W}.X \\ \tilde{W}.Y \\ \tilde{W}.Z \\ \tilde{W}.1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{X} \\ \tilde{Y} \\ \tilde{Z} \\ \tilde{W} \end{bmatrix} = \underset{\mathbb{P}^3}{\tilde{\boldsymbol{X}}}$$

- **From homogeneous to heterogeneous:**

$$\underset{\mathbb{P}^3}{\tilde{\boldsymbol{X}} = \begin{bmatrix} \tilde{X} \\ \tilde{Y} \\ \tilde{Z} \\ \tilde{W} \end{bmatrix}} \Rightarrow \tilde{\boldsymbol{X}} \equiv \left(\tilde{X}/\tilde{W}, \tilde{Y}/\tilde{W}, \tilde{Z}/\tilde{W} \right) \equiv \underset{\mathbb{R}^3}{(X, Y, Z) = \boldsymbol{X}}$$

The fourth coordinate $\tilde{W} \neq 0$ is "fictious" to construct the homogeneous coordinate.

Adapted from: http://6.8300.csail.mit.edu/sp23/lectures/L15/2_single_view_geometry.pdf

Intrinsic Matrix: 3D Camera Coordinates to 2D Image Coordinates

Section 3

Recall: Forward Imaging from 3D to 2D

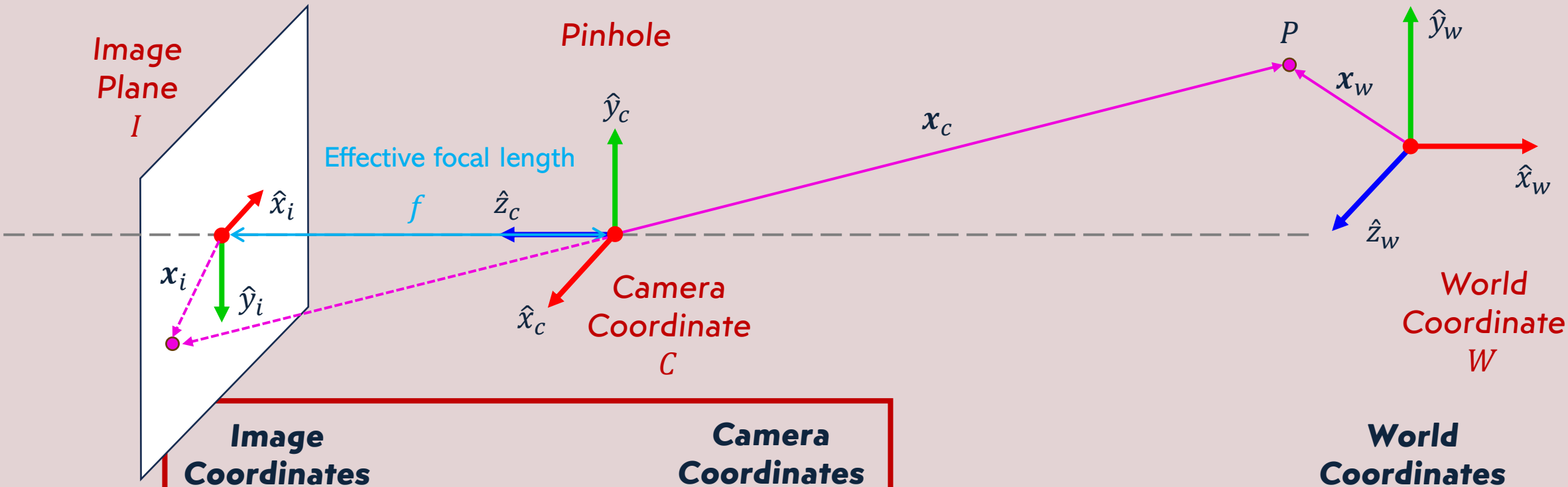


Image Coordinates

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

?

Perspective Projection

Camera Coordinates

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

Coordinate Transformation

World Coordinates

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

https://youtu.be/qByYk6JggQU?si=FRgEVexrzE8RSyeX

Recall: Homogeneous Coordinate (3D)

- The fourth coordinate $\tilde{W} \neq 0$ is "fictious" to construct the homogeneous coordinate.

Heterogeneous → Homogeneous

$$\boldsymbol{X} \equiv \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{W}.X \\ \tilde{W}.Y \\ \tilde{W}.Z \\ \tilde{W}.1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{X} \\ \tilde{Y} \\ \tilde{Z} \\ \tilde{W} \end{bmatrix} = \tilde{\boldsymbol{X}}$$

Homogeneous → Heterogeneous

$$\tilde{\boldsymbol{X}} \equiv \left({}^{\tilde{X}}/_{\tilde{W}}, {}^{\tilde{Y}}/_{\tilde{W}}, {}^{\tilde{Z}}/_{\tilde{W}} \right) \equiv (X, Y, Z) = \boldsymbol{X}$$

Adapted from: http://6.8300.csail.mit.edu/sp23/lectures/L15/2_single_view_geometry.pdf

Perspective Projection in Homogeneous Coordinate

- Perspective projection equations:

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

projection matrix bakal bantu perhitungan lebih cepat karena konversi 3d jadi 2d lebih mudah

- Homogeneous coordinates of (u, v) where $(u, v) = (\tilde{u}/\tilde{w}, \tilde{v}/\tilde{w})$:

dapetin image coordinate

Image Coordinates (2D)

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

Projection Matrix

Camera Coordinates (3D)

Linear Model for Perspective Projection

https://youtu.be/qByYk6JggQU?si=FRgEVexrzE8RSyeX

Intrinsic Matrix for Perspective Projection

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

di dapetkan karena kita berada di homogeneous matrix

Calibration Matrix K

$$K = \begin{bmatrix} f_x & 0 & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix}$$

Non-Zero Upper Right Triangular Matrix

Intrinsic Matrix M_{int}

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

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

https://youtu.be/qByYk6JggQU?si=FRgEVexrzE8RSyeX

Recall: Forward Imaging from 3D to 2D

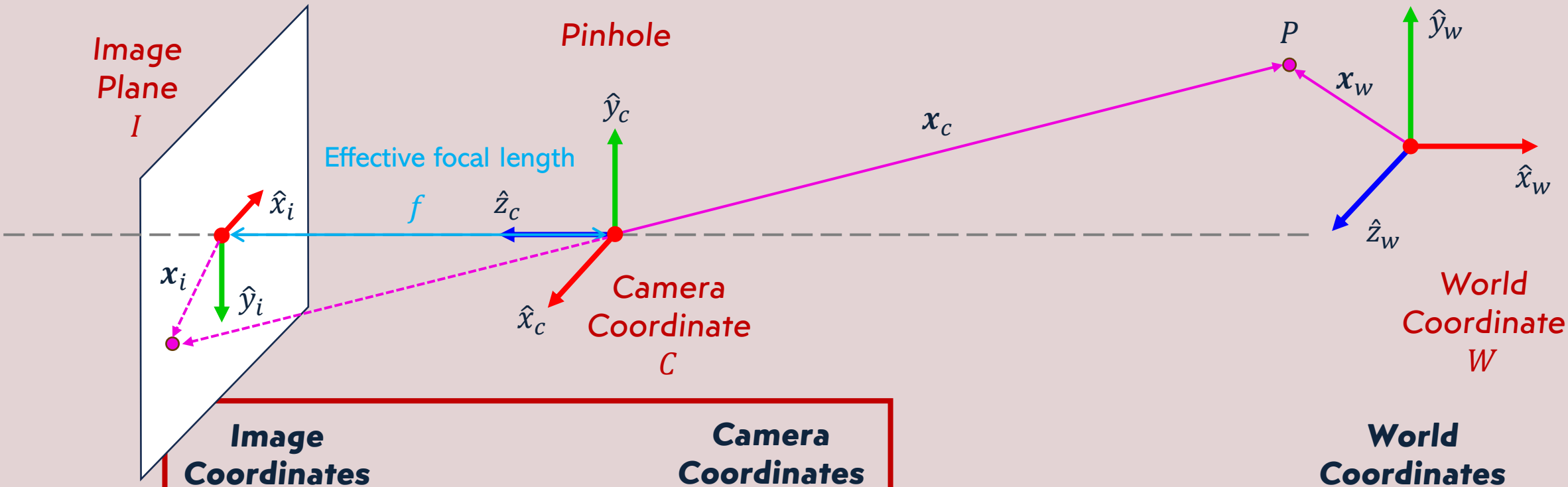


Image Coordinates

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

M_{int}

Perspective Projection

Camera Coordinates

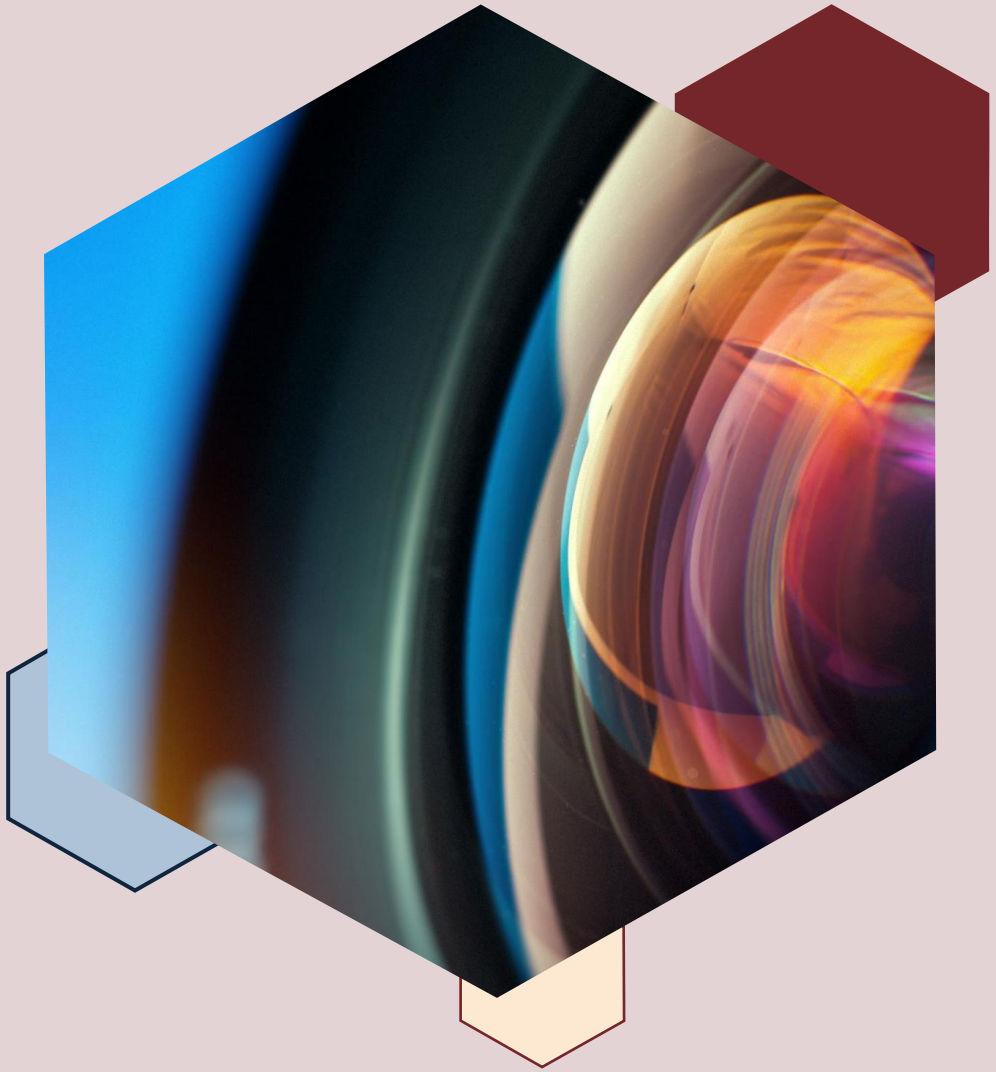
$$x_c = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$$

Coordinate Transformation

World Coordinates

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

https://youtu.be/qByYk6JggQU?si=FRgEVexrzE8RSyeX



Extrinsic Matrix: 3D World Coordinates to 3D Camera Coordinates

Section 4

Recall: Forward Imaging from 3D to 2D

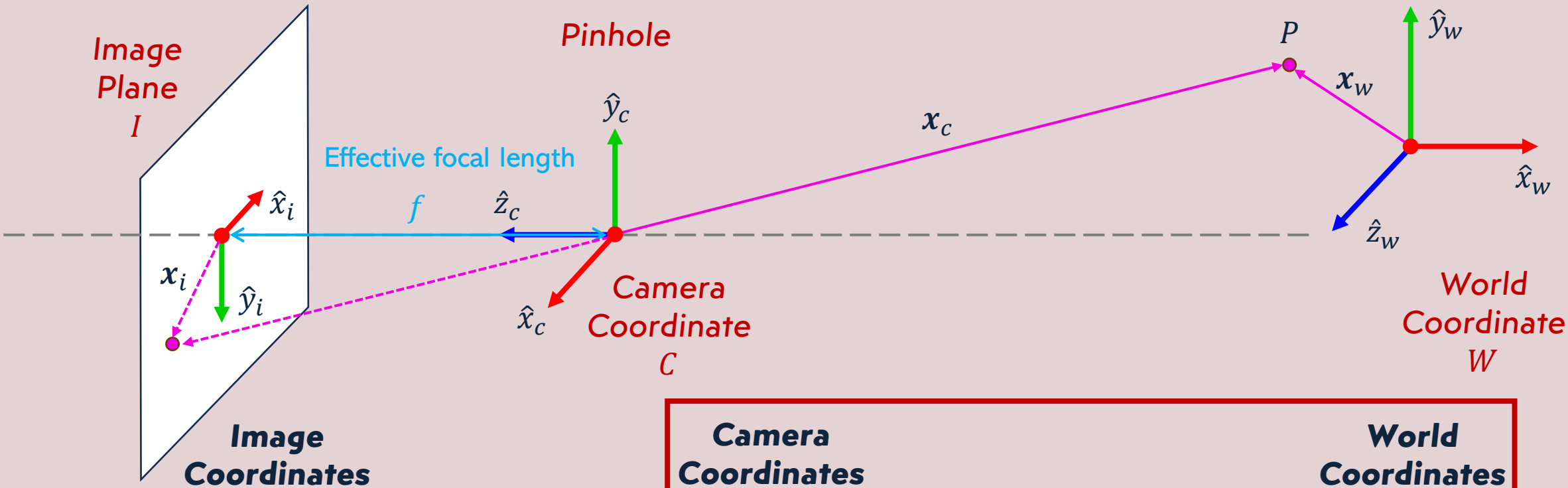


Image Coordinates

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

M_{int}

Perspective Projection

Camera Coordinates

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

?

Coordinate Transformation

World Coordinates

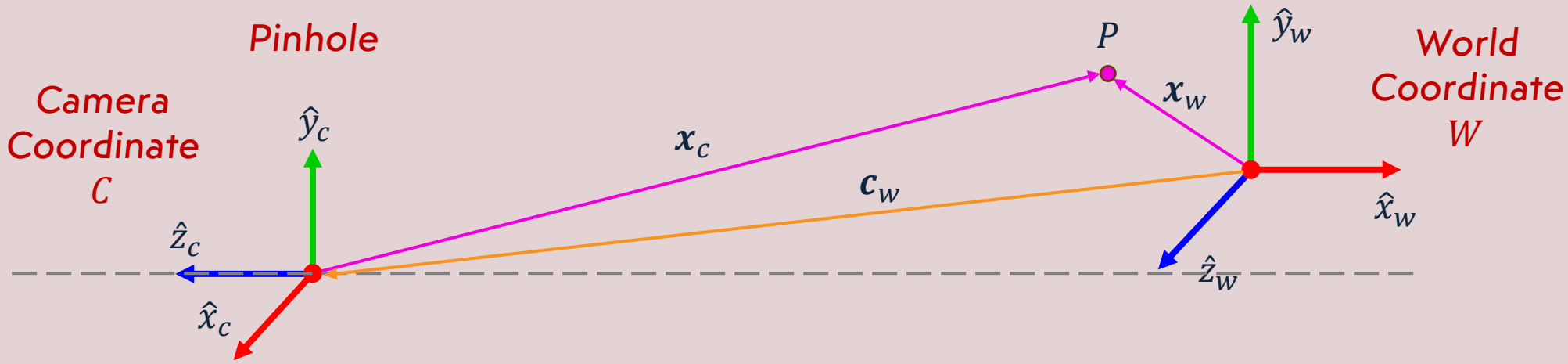
$$x_w = \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix}$$

https://youtu.be/qByYk6JggQU?si=FRgEVexrzE8RSyeX

Extrinsic Parameters

extrinsic parameter terdiri atas: position dan orientation/rotation

- Coordinate transformation from world coordinates to camera coordinates is based on the position $(\boldsymbol{c}_w)$ and orientation/rotation (R) (i.e., **the extrinsic parameters**) of the camera coordinate frame with respect to the world coordinate frame W.



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

→ Row 1: Direction of $\hat{x}_c$ in world coordinate frame

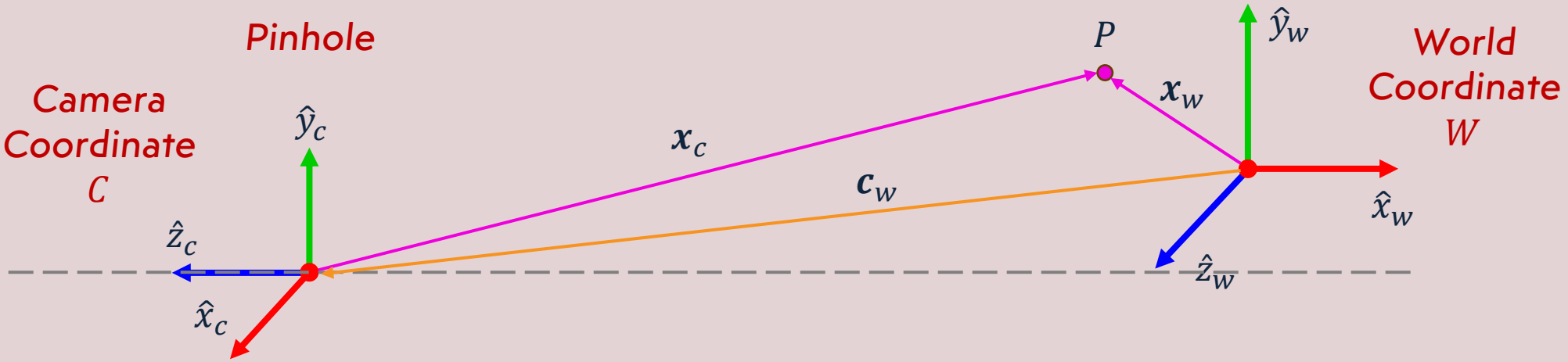
→ Row 2: Direction of $\hat{y}_c$ in world coordinate frame

→ Row 3: Direction of $\hat{z}_c$ in world coordinate frame

Note: Matrix R is orthonormal.

https://youtu.be/qByYk6JggQU?si=FRgEVexrzE8RSyeX

World-to-Camera Transformation



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

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

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

Extrinsic Matrix

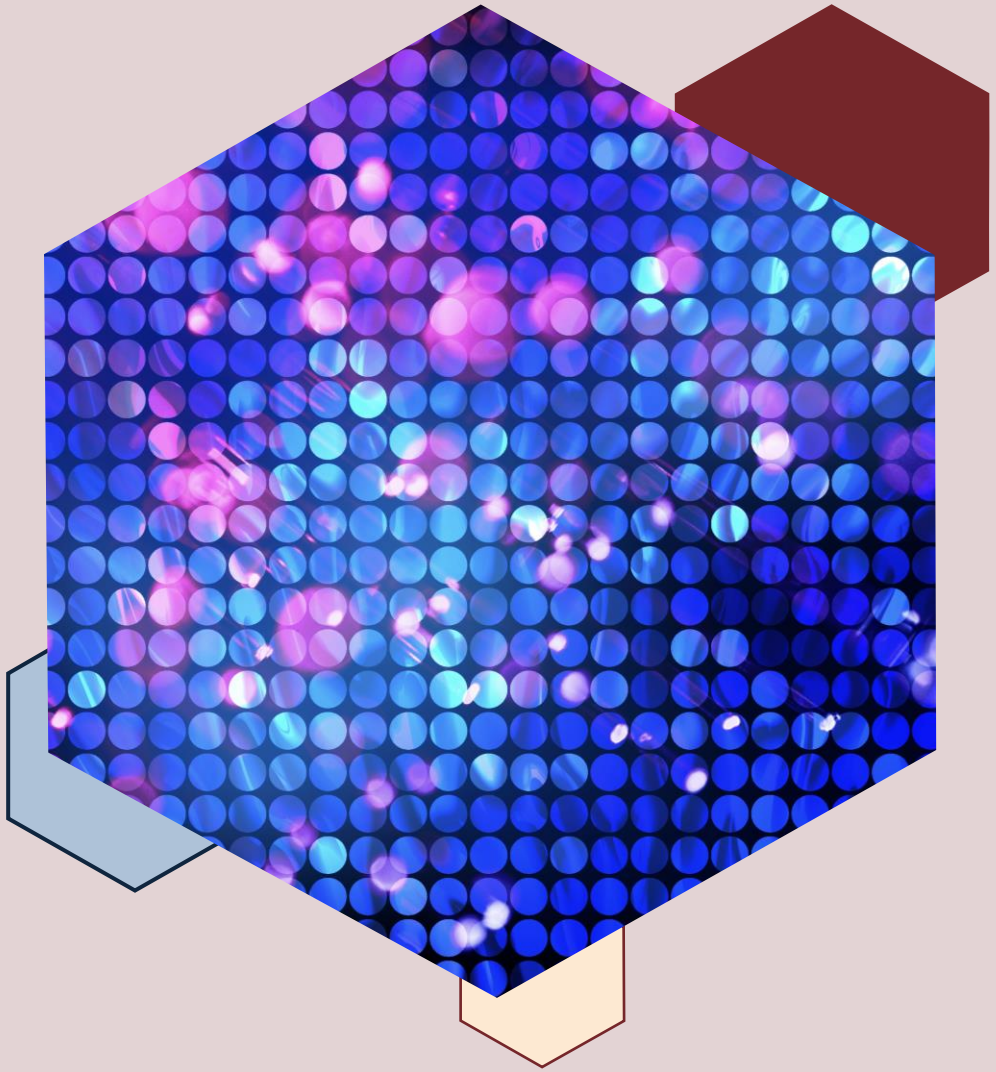
- Rewriting using homogenous coordinates:

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

- Thus, the extrinsic matrix:

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

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



Projection Matrix

Section 5

Recap: Forward Imaging from 3D to 2D

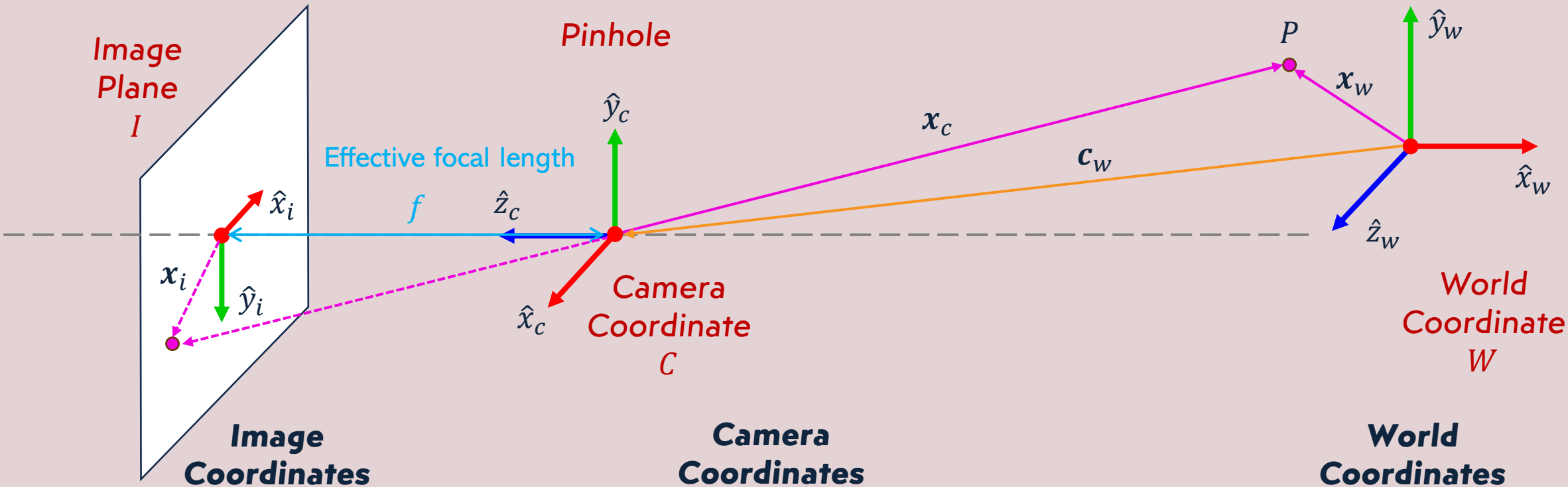


Image Coordinates $x_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}$

M_{int} ← Perspective Projection

Camera Coordinates $x_c = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$

M_{ext} ← Coordinate Transformation

World Coordinates $x_w = \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix}$

https://youtu.be/qByYk6JggQU?si=FRgEVexrzE8RSyeX

Projection Matrix P

Camera to Pixel

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

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

World to Camera

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

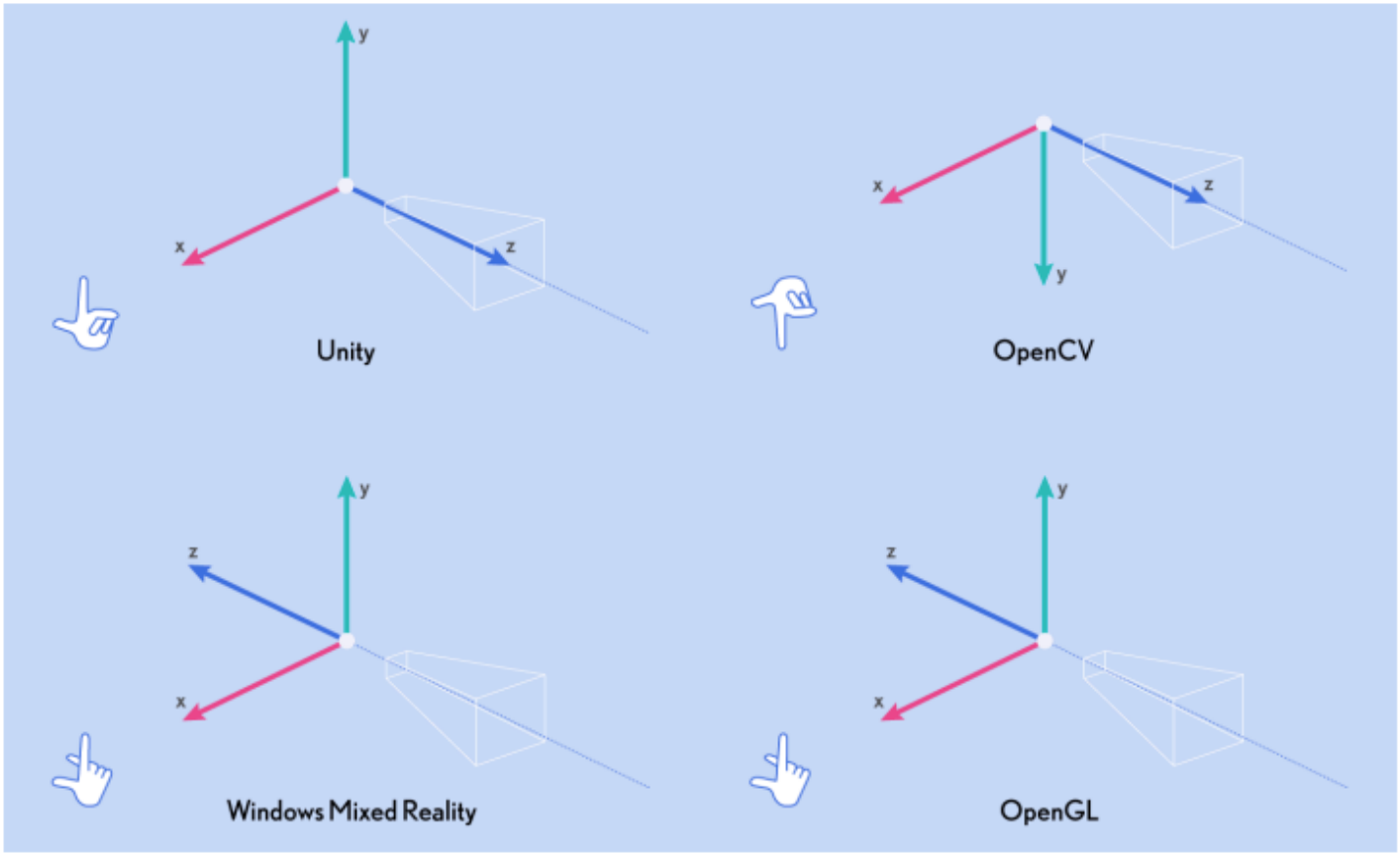
$$\tilde{\boldsymbol{x}}_c = \begin{bmatrix} R_{3\times3} & t \\ 0_{1\times3} & 1 \end{bmatrix} \cdot \tilde{\boldsymbol{x}}_w = M_{ext} \cdot \tilde{\boldsymbol{x}}_w$$

- Combining the equations of M_{int} and M_{ext}, we get the full projection matrix P:

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

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

Beware of Conventions



X=left or right? Y=up or down? etc..

If you externally obtain transformation matrices (e.g. using someone else's code), make sure of convention compatibility (one of the *most common* sources of bugs!)

read https://pytorch3d.org/docs/cameras

Adapted from: http://6.8300.csail.mit.edu/sp23/lectures/L15/2_single_view_geometry.pdf

Multi-Camera Vision: Simple Stereo Vision

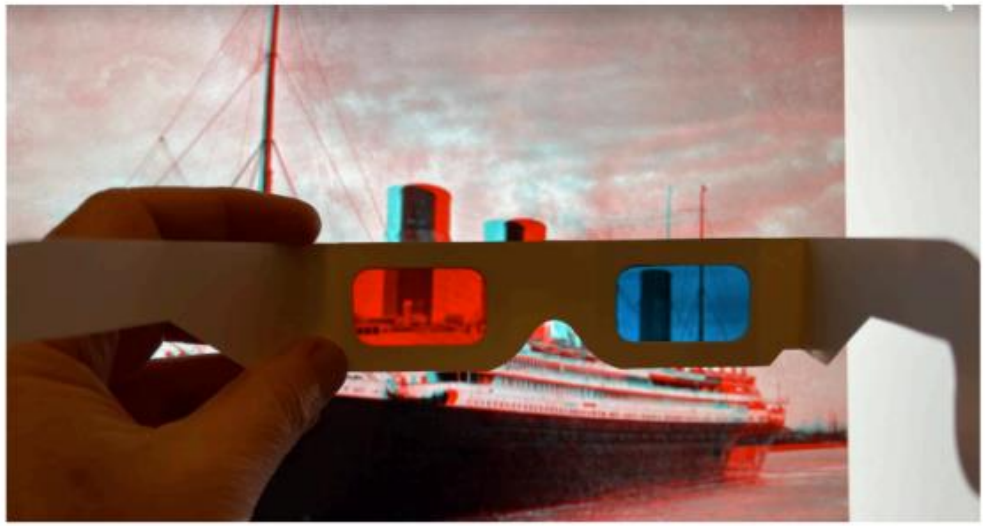
Section 6

Stereo Images

https://en.wikipedia.org/wiki/Stereo_photography_techniques

Stereo Images of Titanic

(a)

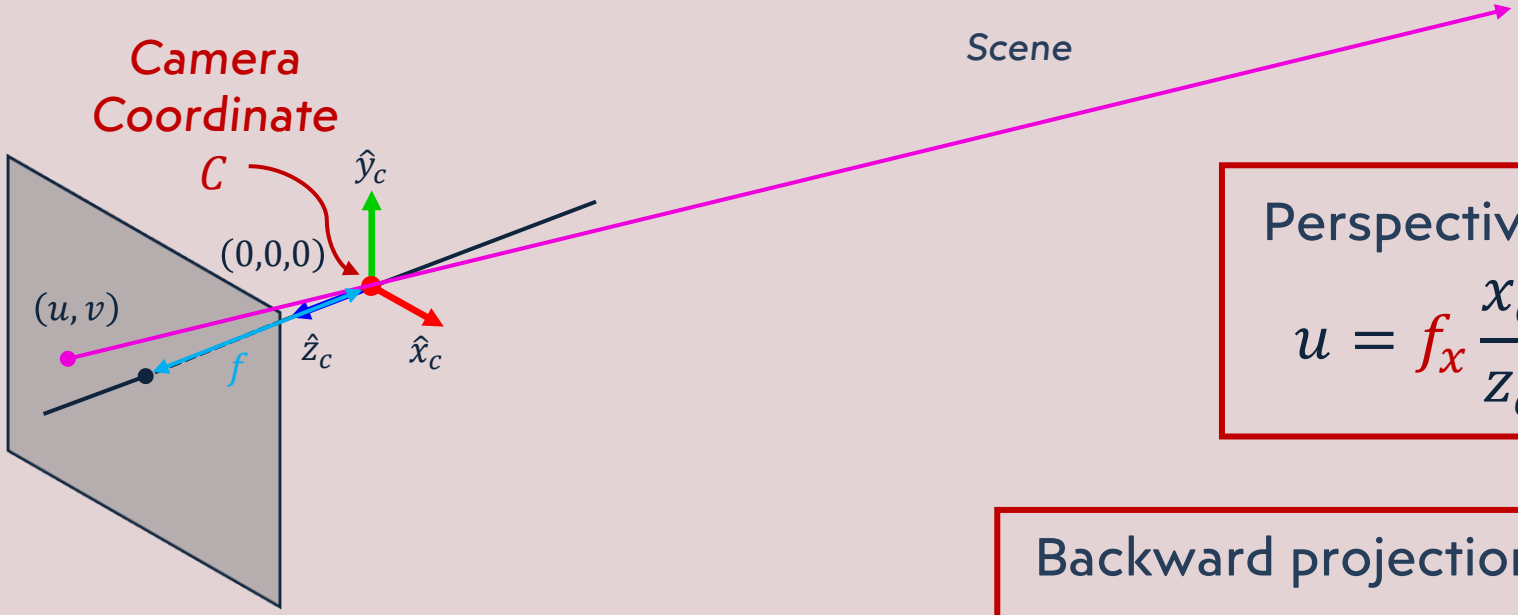


(b)

Figure 1.1: (a) Stereo anaglyph of the ocean liner, the Titanic [McManus2022]. The red image shows the right eye's view, and cyan the left eye's view. When viewed through stereo red/cyan stereo glasses, as in (b), the cyan contrast appears in the left eye image and the red variations appear to the right eye, creating a the perception of 3d.

Computing 2D-to-3D Outgoing Ray

- Given a calibrated camera (i.e., intrinsic parameters of the camera are known), can we find the 3D scene point from a single 2D image? No, but we can trace back the ray/light!



kita bisa pakai intrinsic parameter untuk mendapatkan 3d Scene point.

Perspective projection point (3D-to-2D):

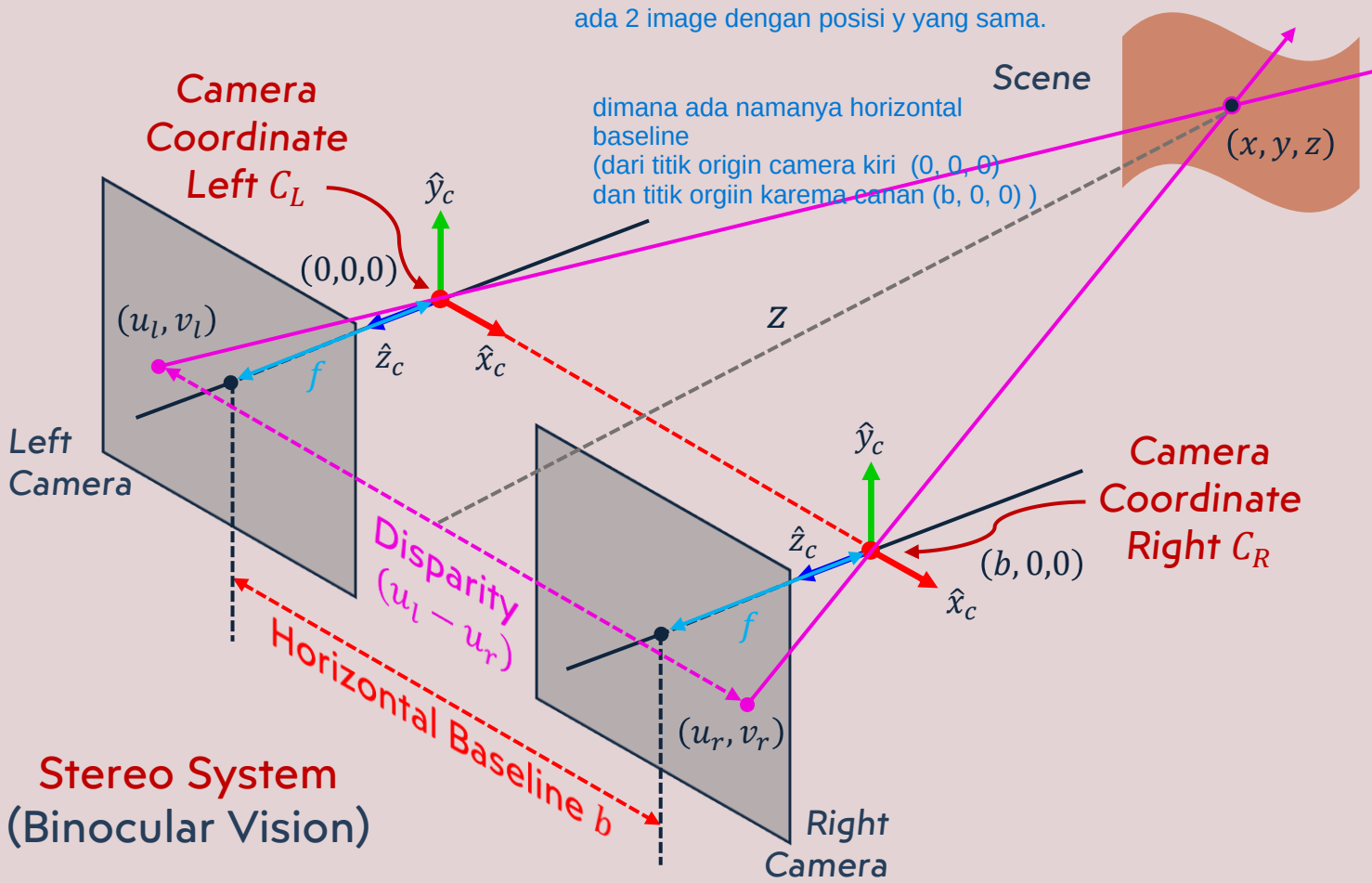
$$u = f_x \frac{x_c}{z_c} + o_x \qquad v = f_y \frac{y_c}{z_c} + o_y$$

Backward projection of ray/light (2D-to-3D):

$$x = \frac{z}{f_x}(u - o_x) \qquad y = \frac{z}{f_y}(v - o_y) \qquad z > 0$$

Triangulation of Two Cameras

- Given a calibrated camera (i.e., intrinsic parameters of the camera are known), can we find the 3D scene point from a single 2D image? No, but we can trace back the ray/light!



Perspective projection point of the left camera (3D-to-2D):

$$u_l = f_x \frac{x}{z} + o_x \qquad v_l = f_y \frac{y}{z} + o_y$$

Perspective projection point of the right camera (3D-to-2D):

karena objek dikanan itu ada b

$$u_r = f_x \frac{x - b}{z} + o_x \qquad v_r = f_y \frac{y}{z} + o_y$$

Notes: Intrinsic parameters f_x, f_y, o_x, o_y, and b are known (i.e., calibrated camera).

https://youtu.be/qByYk6JggQU?si=FRgEVexrzE8RSyeX

Simple Stereo: Depth and Disparity

- From perspective projection:

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

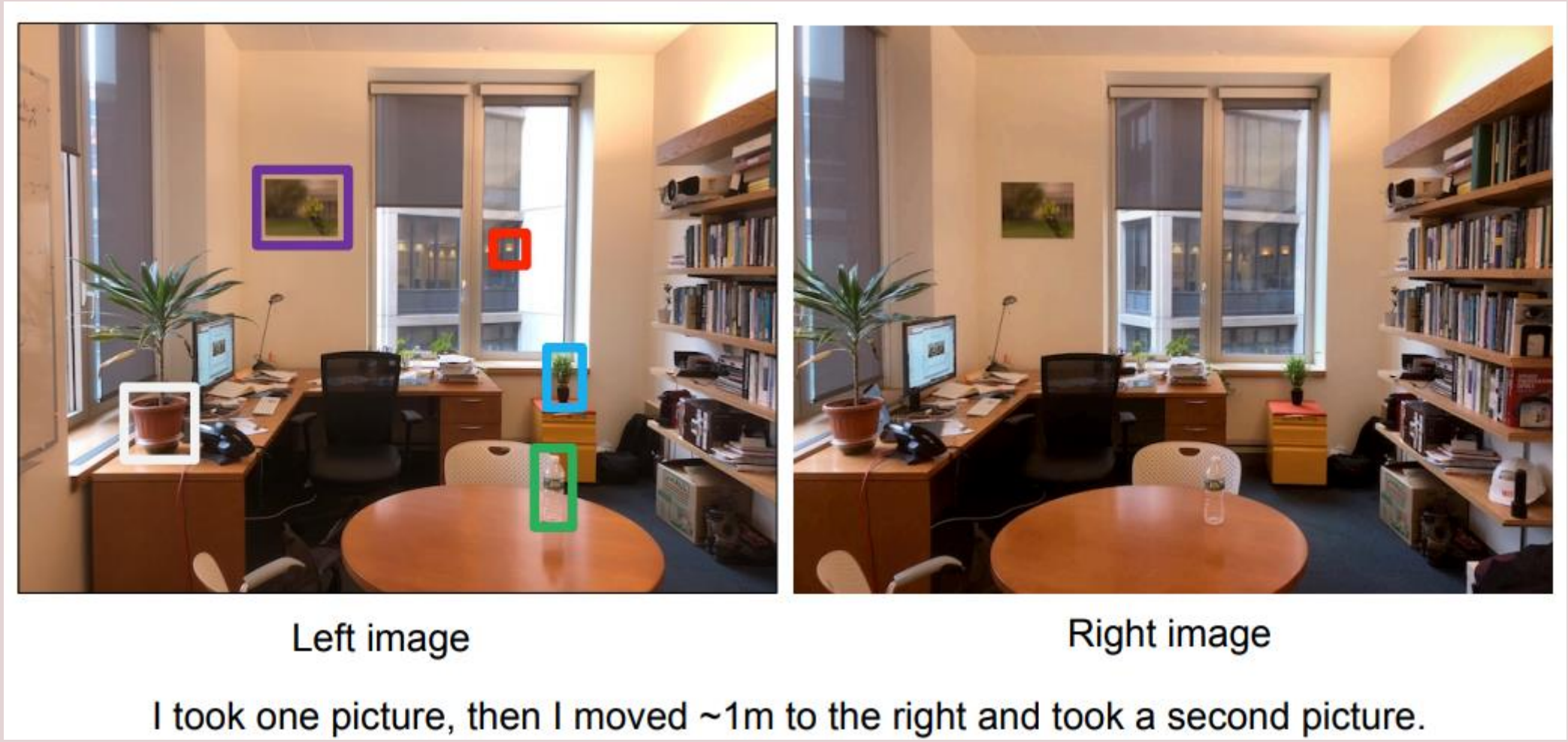
- Solving for (x, y, z):

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

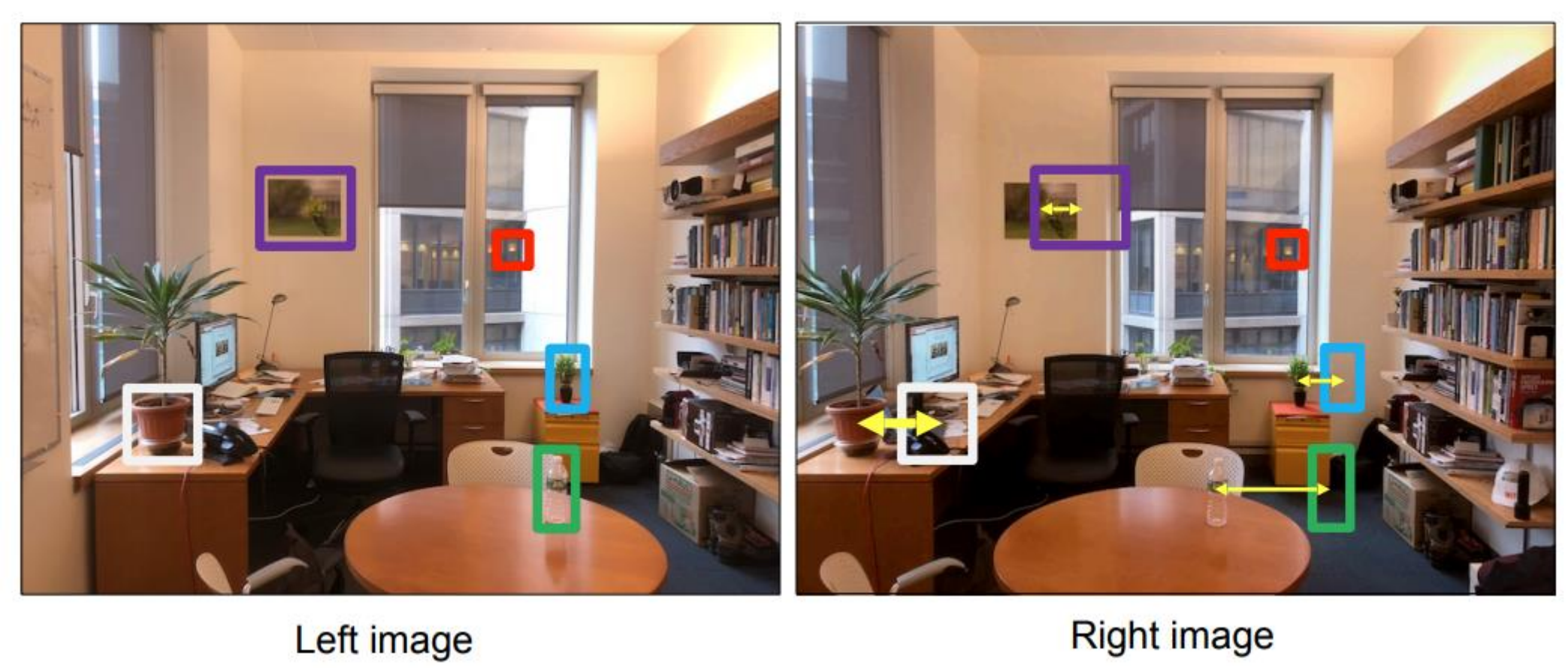
- The z is the "depth" while $(u_l - u_r)$ is called "disparity".
  - Depth z is inversely proportional to disparity $(u_l - u_r)$.
  - Disparity $(u_l - u_r)$ is proportional to baseline b.

nanti biar dapet semuanya itu hasil-hasilnya di akomodasi buat mendapatkan yang real

From Disparity to Depth (1)

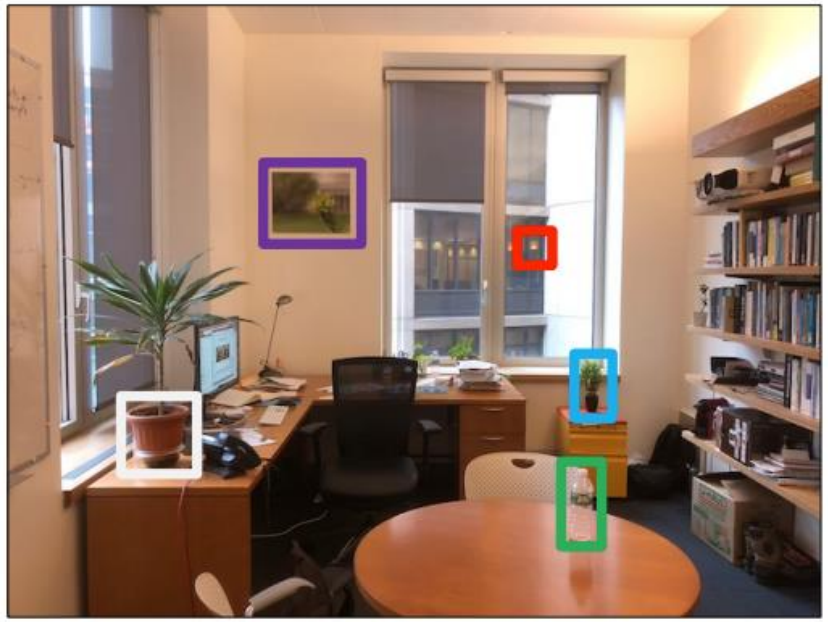


From Disparity to Depth (2)

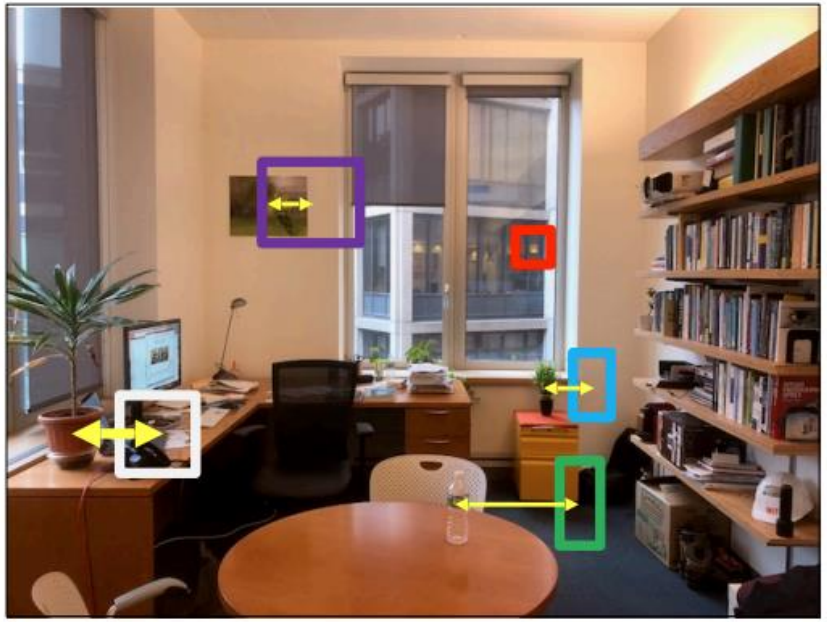


Adapted from: http://6.8300.csail.mit.edu/sp23/lectures/L15/2_single_view_geometry.pdf

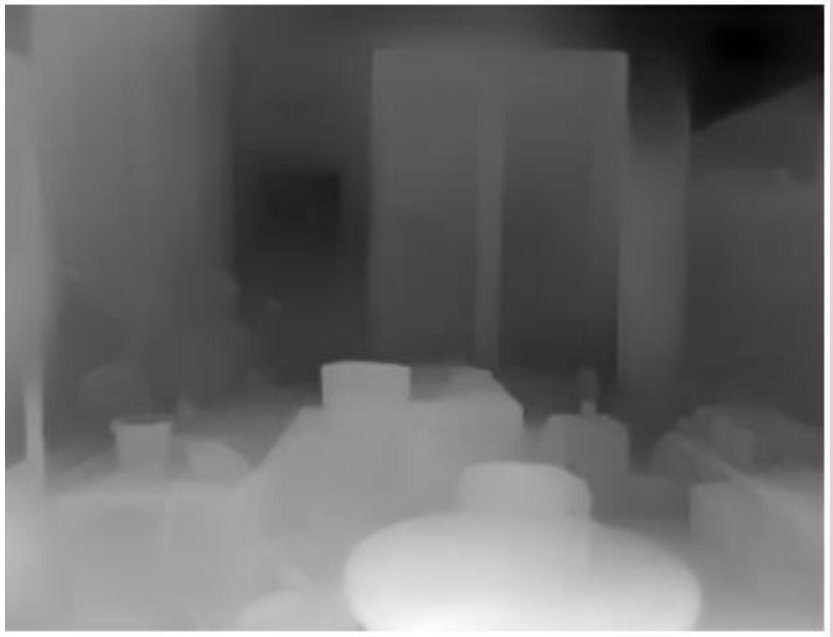
From Disparity to Depth (3)



Left image



Right image

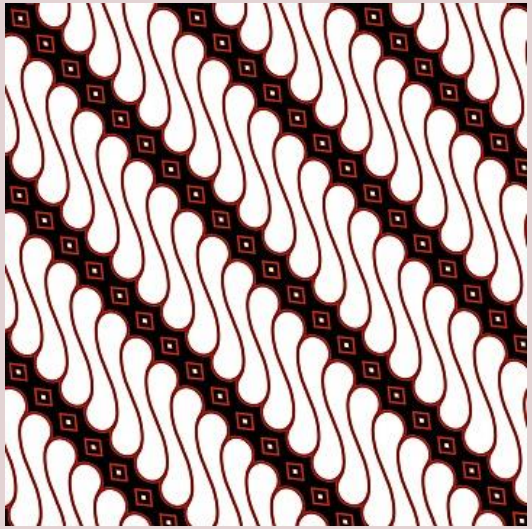


$$z = \frac{b f_x}{(u_l - u_r)}$$

- Template matching!
- Based on the triangulation, disparity only happen on the horizontal axis!

Challenges of Stereo Matching

- Surface must have (non-repetitive) texture



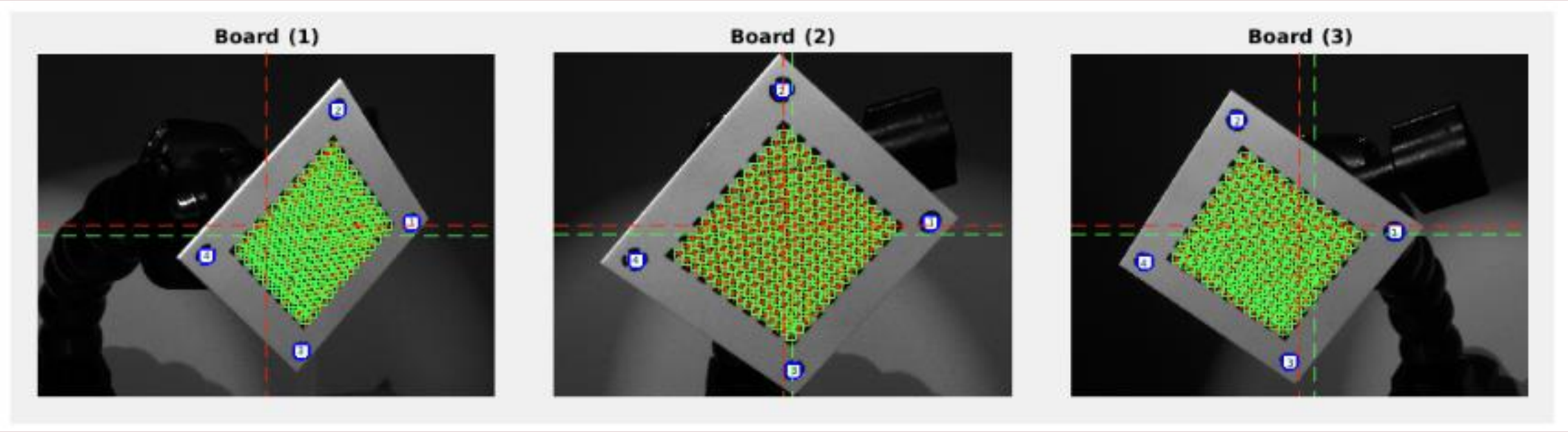
- The window size for stereo matching.
- The objects are too close/far away from stereo vision system.
- Etc.

Advanced Topics of Multi-Camera Vision Systems

Section 7

Uncalibrated Cameras

- Stereo vision with uncalibrated cameras, e.g.,
  - the intrinsic parameters f_x, f_y, o_x, o_y, and b are unknown
  - the orientations of cameras are not simple (i.e., not horizontally placed)
- We have to calculate more geometry equations for calibrating the cameras.



https://justinblaber.org/camera-calibration-application/

Multi-View Geometry

- Suppose we have a landmark building of a country. Can we reconstruct its 3D model from multiple photos from the different people?




Thank you!

CSCE604133 Computer Vision
Faculty of Computer Science
Universitas Indonesia

Dr. Eng. Laksmita Rahadianti
Muhammad Febrian Rachmadi, Ph.D.
Dr. Dina Chahyati, Prof. Dr. Aniati M. Arymurthy