ITE4052 Computer Vision

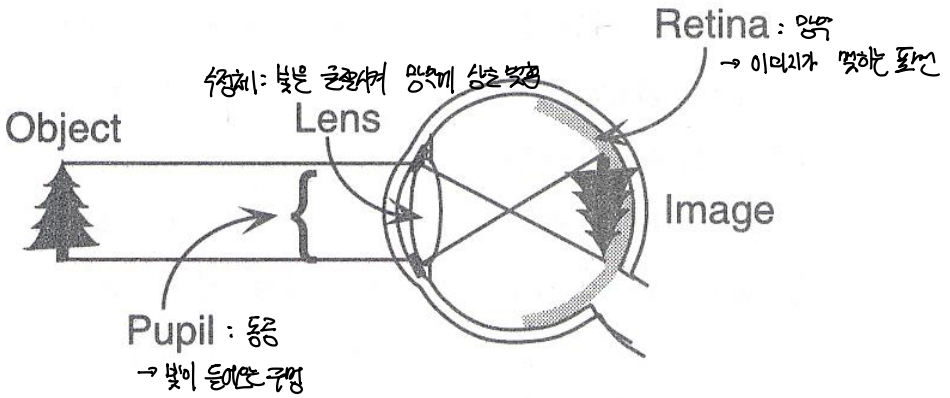
Camera

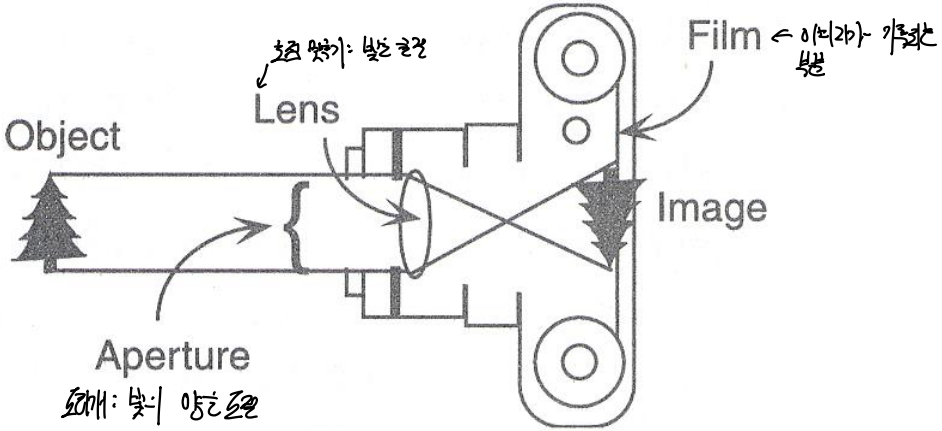
Dong-Jin Kim

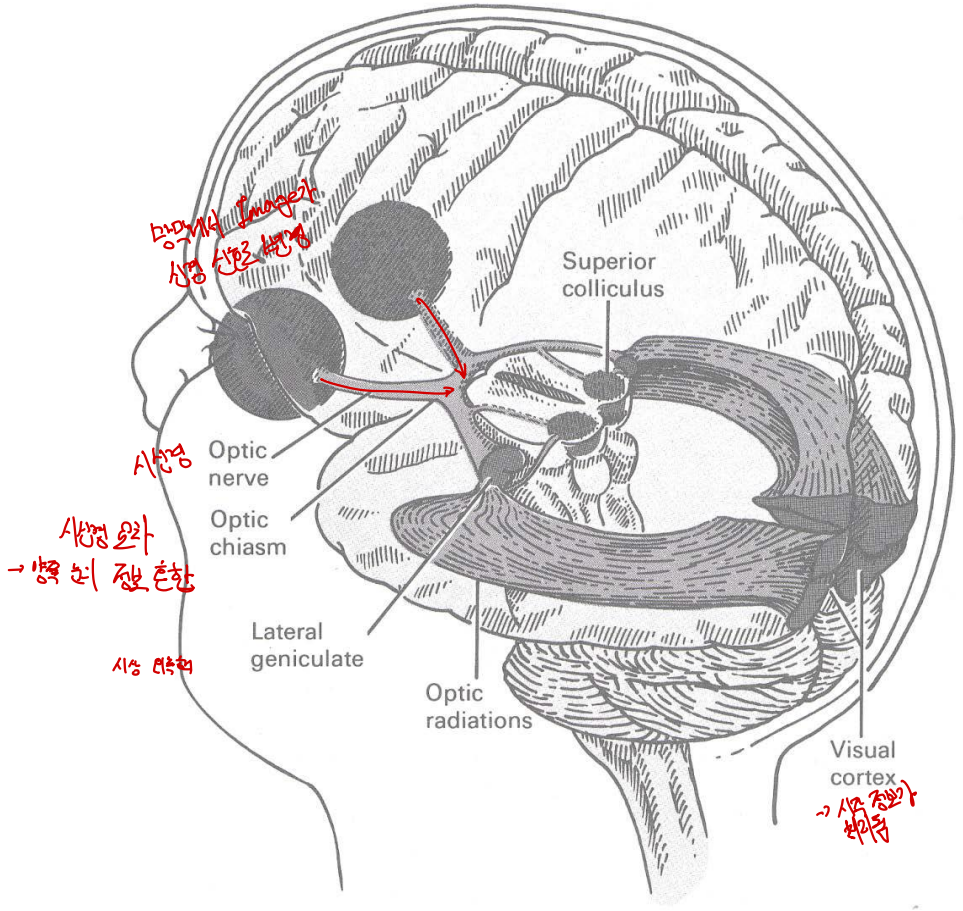
Spring 2025

Eye-Camera Analogy

Eye :

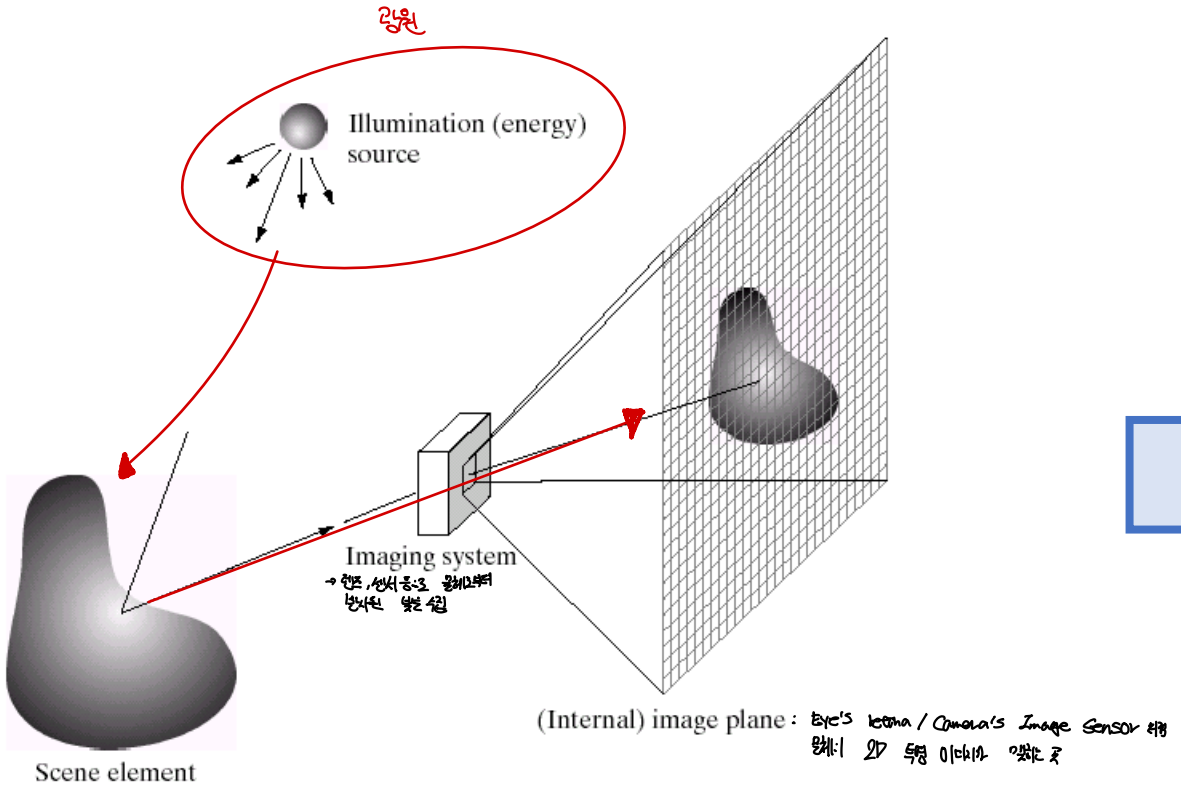






What is an image?

What is an image?



Digital Image
: A matrix

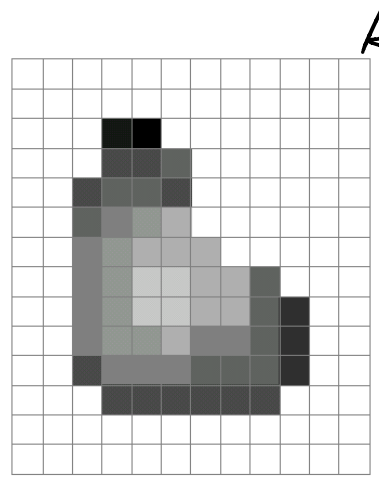


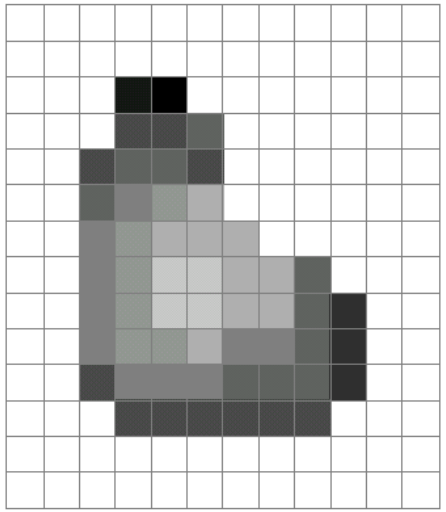
Image System에 의해 맺힌 이미지를 pixel이라는 단위로 샘플링한것

Digital Camera

What is an image?

Pixel 하나하나 색상 정보 저장 필요

A **matrix** of intensity values (0 ~255)

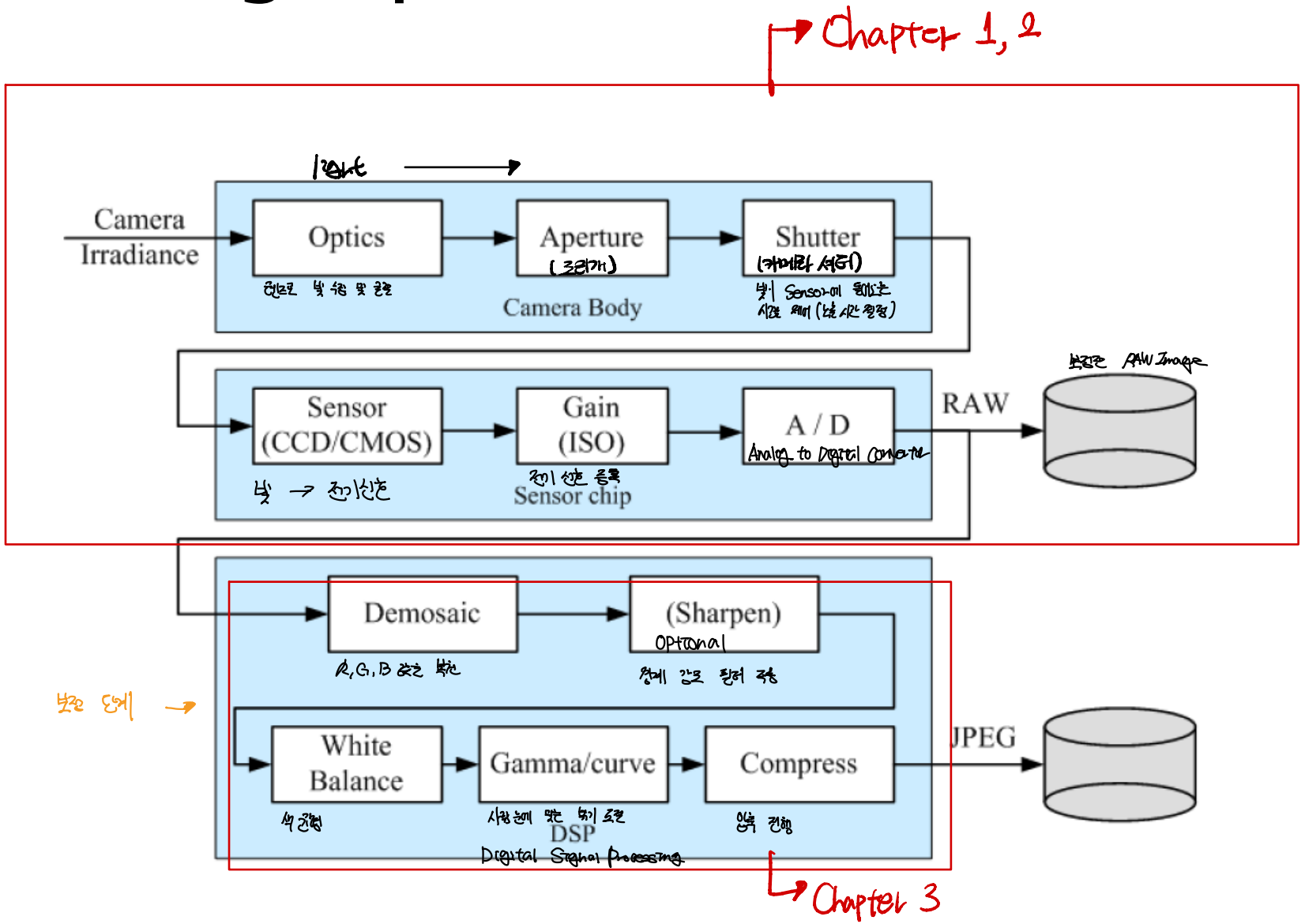


=

255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	20	0	255	255	255	255	255	255	255
255	255	255	75	75	75	255	255	255	255	255	255
255	255	75	95	95	75	255	255	255	255	255	255
255	255	96	127	145	175	255	255	255	255	255	255
255	255	127	145	175	175	175	255	255	255	255	255
255	255	127	145	200	200	175	175	95	255	255	255
255	255	127	145	200	200	175	175	95	47	255	255
255	255	127	145	145	175	127	127	95	47	255	255
255	255	74	127	127	127	95	95	95	47	255	255
255	255	255	74	74	74	74	74	74	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255

(0 = black, 255 = white)

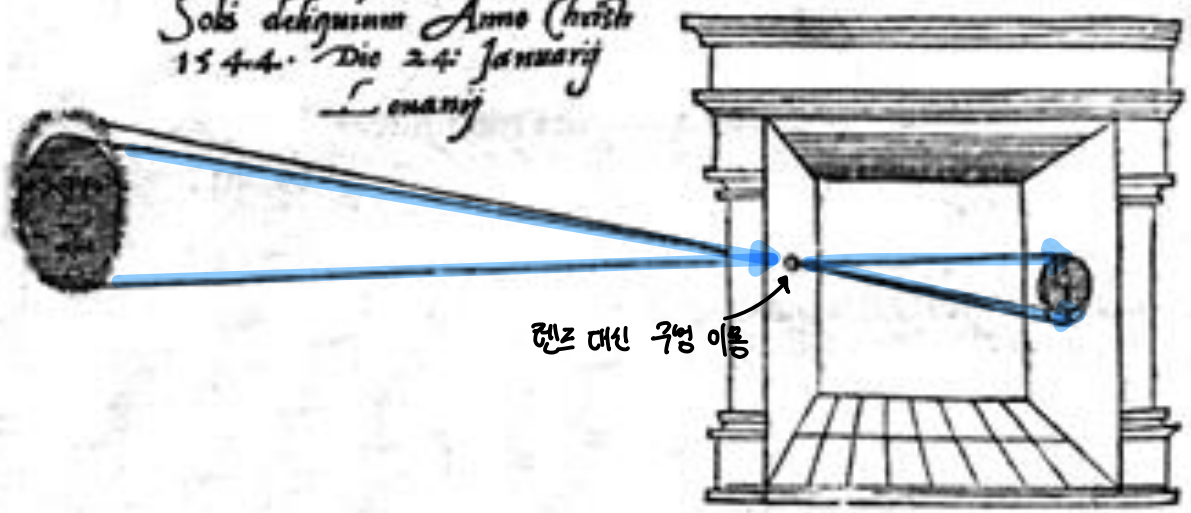
Camera Sensing Pipeline



Pinhole Camera (Camera Obscura)

illum in tabula per radios Solis, quàm in cœlo contingit: hoc eſt, ſi in cœlo ſuperior pars deliquiũ patiatur, in radiis apparebit inferior deficere, vt ratio exigit optica.

Solis deliquium Anno Christi 1544. Die 24. Januarij Louanij



Sic nos exactè Anno .1544. Louanii eclipſim Solis obſeruauimus, inuenimusq; deficere paulò plus q̃ dex-

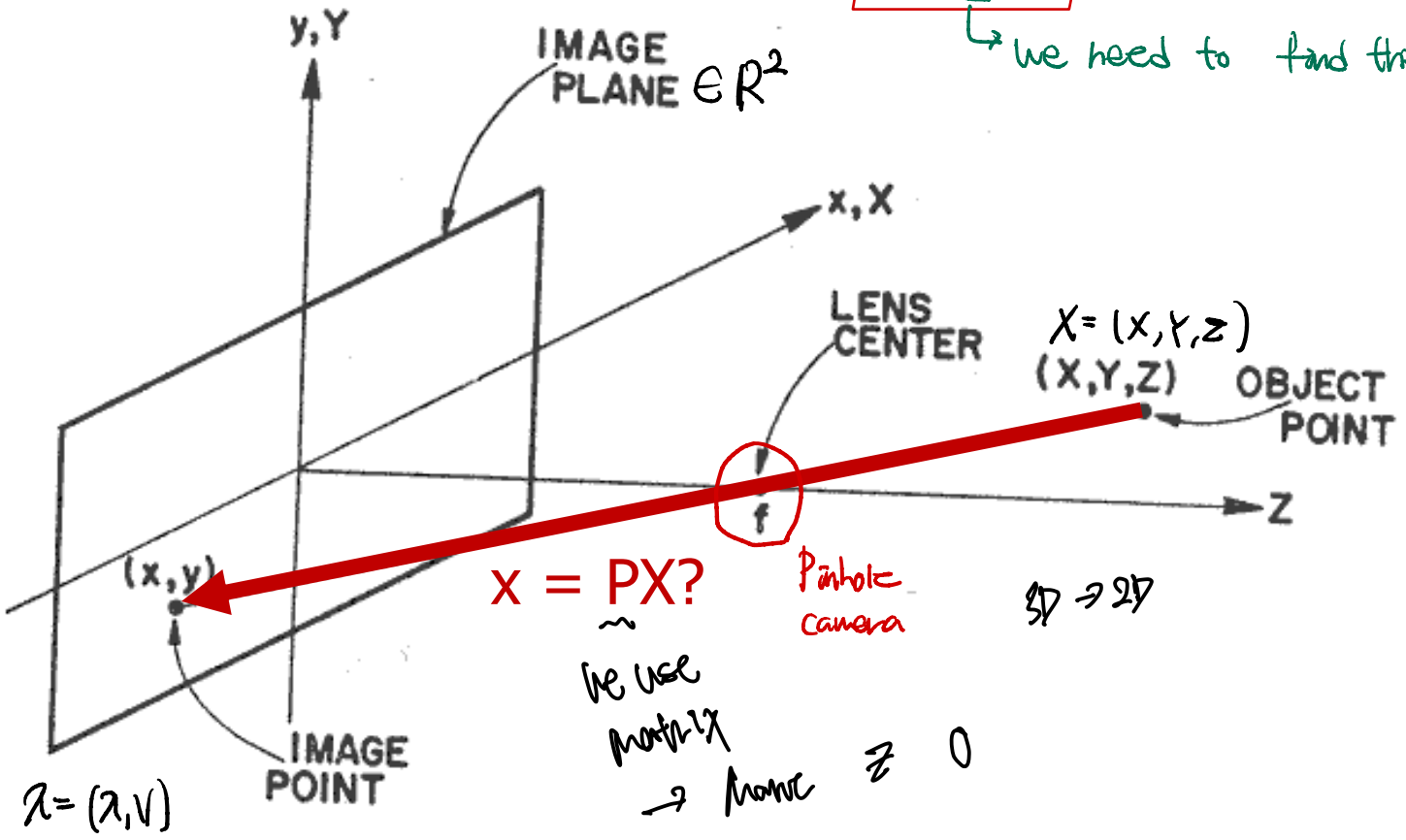
Camera Anatomy

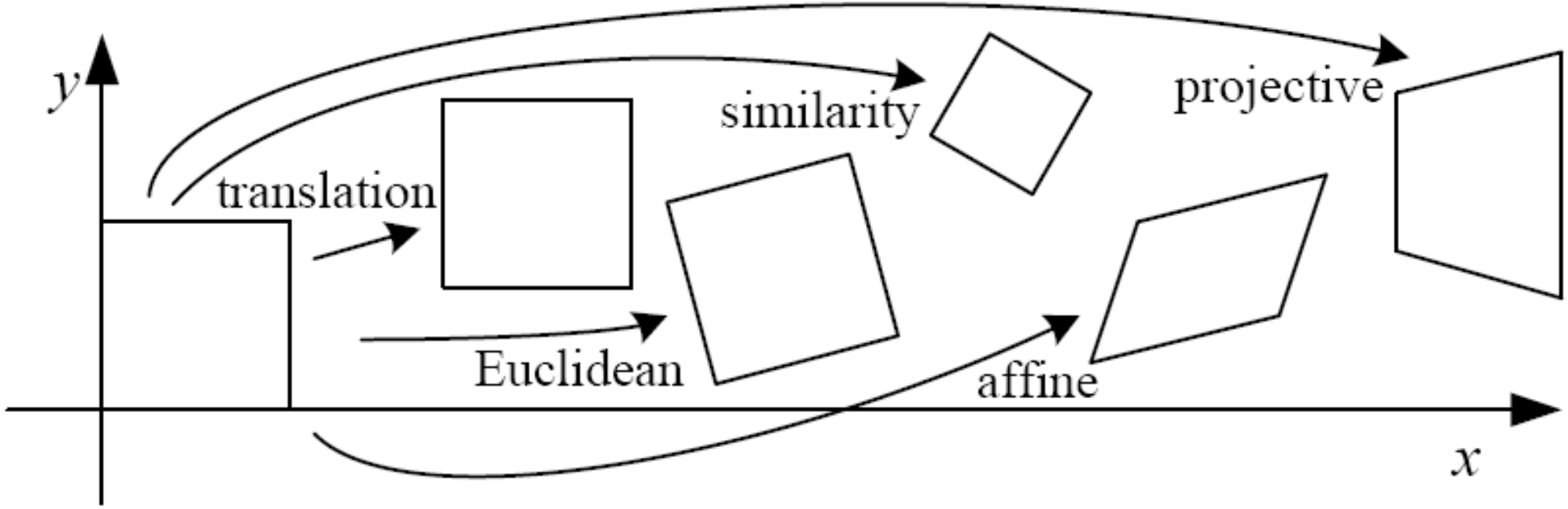
We have to know

Relation between object point(X, Y, Z) and (x, y) Image point

$x = PX$ 3D(X) to 2D(x)

↳ we need to find this matrix P



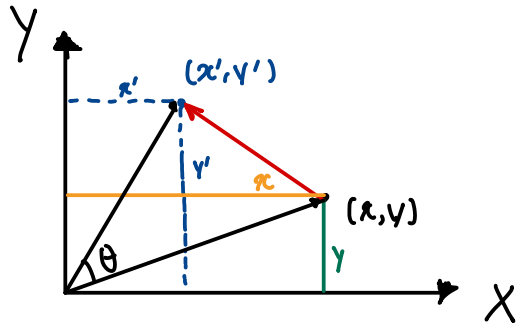


Before 3D, Let’s start from 2D Transformation

2D Image Transformation (2D to 2D)

Affine Transformation

- **Translation**, **Scaling**, and **Rotation**



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Translation

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Scaling

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

Rotation

$$x' = x\cos\theta - y\sin\theta$$

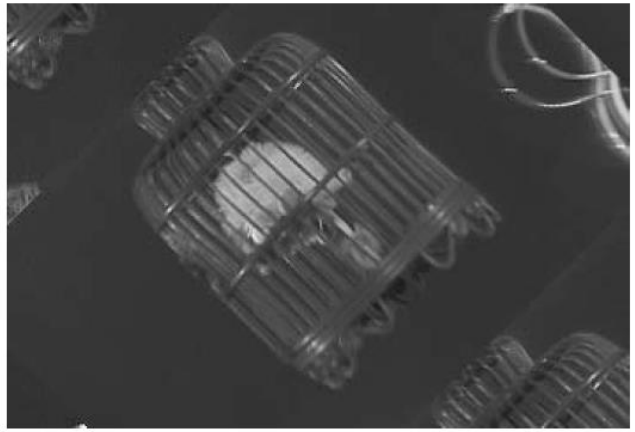
$$y' = x\sin\theta + y\cos\theta$$

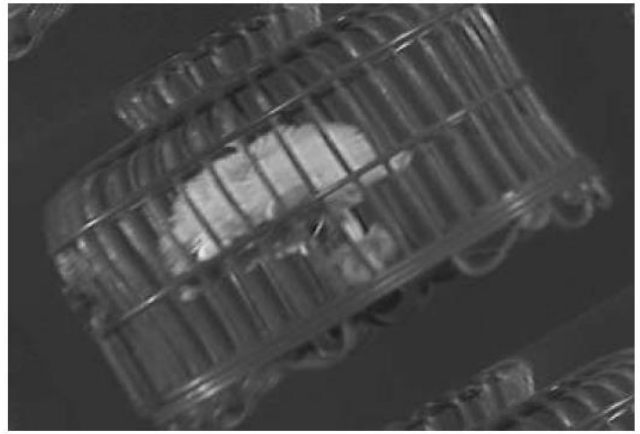
2D Image Transformation

Original

S_x=2,
S_y=2

$\theta = -45°$





S_x=2,
S_y=1,
$\theta = -30°$

Linear Transformations

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

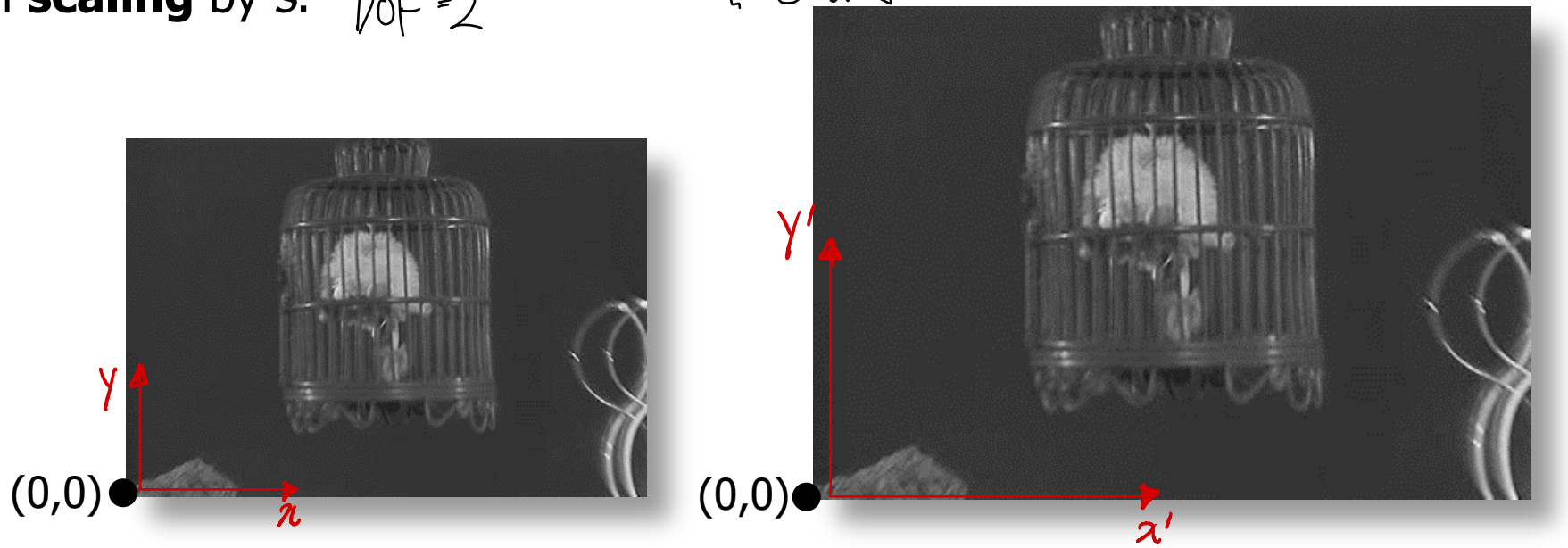
$x' = s_x \cdot x$

$y' = s_y \cdot y$

$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$

$A^{-1} = \frac{1}{ad-bc} \begin{bmatrix} d & b \\ -c & a \end{bmatrix}$

- Uniform **scaling** by *s*: DOF = 2

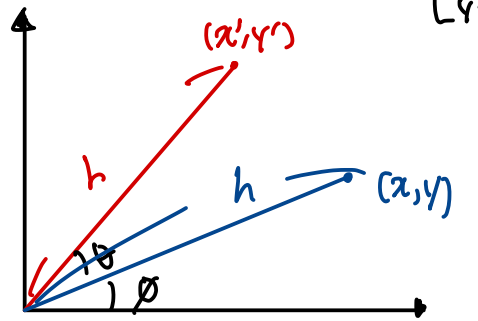


$$\mathbf{S} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$

What is the inverse?

$S^{-1} = \begin{bmatrix} \frac{1}{s_x} & 0 \\ 0 & \frac{1}{s_y} \end{bmatrix}$

Linear Transformations



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} A \end{bmatrix} \begin{bmatrix} x = h\cos\phi \\ y = h\sin\phi \end{bmatrix}$$

$$= \begin{bmatrix} h\cos(\theta+\phi) \\ h\sin(\theta+\phi) \end{bmatrix}$$

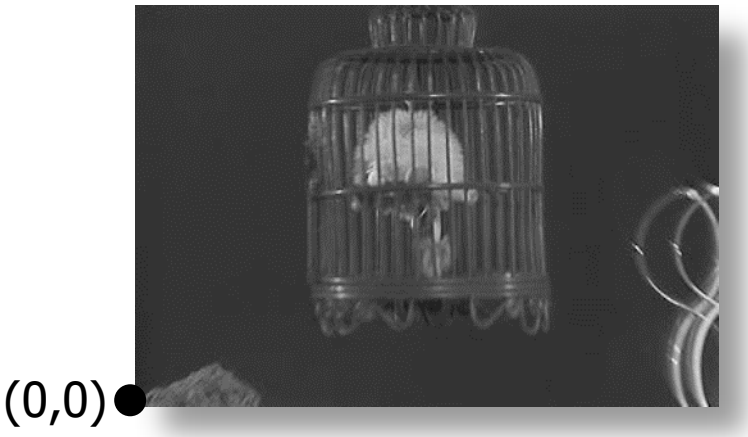
$$= \begin{bmatrix} h\cos(\theta)\cos(\phi) - h\sin(\phi)\sin(\theta) \\ h\sin\theta\cos(\phi) - h\cos(\theta)\sin(\phi) \end{bmatrix}$$

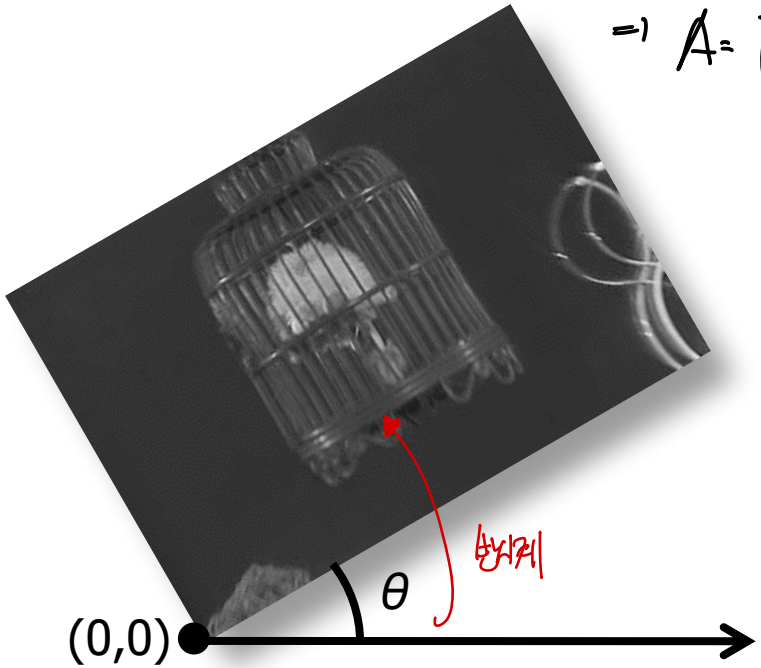
$$= \begin{bmatrix} x\cos\theta - y\sin\theta \\ x\sin\theta + y\cos\theta \end{bmatrix}$$

$$\Rightarrow A = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

- **Rotation** by angle θ (about the origin)

→ Rotation matrix를 구하는 방법이 됩니다.





$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

What is the inverse? $= R(-\theta)$

For rotations:

$$\mathbf{R}^{-1} = \mathbf{R}^T \quad R^{-1} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

Linear Transformations

- What else with a 2x2 matrix?

2D mirror across Y axis?

$$\begin{aligned} x' &= -x \\ y' &= y \end{aligned}$$

$$\mathbf{T} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

2D mirror across line y = x?

$$\begin{aligned} x' &= y \\ y' &= x \end{aligned}$$

$$\mathbf{T} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Linear Transformations?

↳ 입력 벡터에 대해 행렬이 곱셈의 연산을

Definition of Linear Transformation

$$\begin{cases} ① f(cx) = c \cdot f(x) \\ ② f(x_1 + x_2) = f(x_1) + f(x_2) \end{cases}$$

T(x)는 모든 Translation matrix에 대한 결과 X'

- What else with a 2x2 matrix?

2D Translation?

$$x' = x + t_x$$
$$y' = y + t_y$$

NO!

특이점 < $T(u)+T(v) = (u+v)+2t$
$T(u+v) = (u+v)+t$

$T(x) = x + t_x$ 인데,
$T(c \cdot x) = c \cdot T(x)$ 면 Linear하다!
하지만 $T(c \cdot x) = c \cdot x + t_x$
$\neq c(x + t_x)$
따라서 Non-linear

Translation is not a **linear operation** on 2D coordinates

2D에서 Translation이 Linear 하려면

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = [M] \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \end{bmatrix}$$

t_x, t_y

$M = \begin{bmatrix} 1 & \frac{t_x}{y} \\ \frac{t_y}{x} & 1 \end{bmatrix}$ 만 가능

곱셈에 있어 not linear

→ 3D에서 가능하다.

2D Image Transformation

- Translation: $P' = P + T$
- Scaling: $P' = SP$
- Rotation: $P' = RP$

$$T_h \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix}$$

$$S_h \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{bmatrix} ax + t \\ by + t \\ 1 \end{bmatrix}$$

$$R_h \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}$$

1. 실제적 3D 좌표
   → 3D to 3D transformation 진행
   → Extrinsic 이라고 한다.
2. 3D → 2D projection (카메라 촬영할 때)
   → $\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$ ← Intrinsic 이라고 한다.

   이동 등은 위해 Camera parameter 사용

   → $\left[\frac{x'}{w} \quad \frac{y'}{w}\right]$

Real 3D Point → Camera 3D Point → Image Point
(X, Y, Z) (X_c, Y_c, Z_c) (x, y)

We want to treat all transformations as **linear** operations (matrix multiplication)

➔ **"Homogeneous Coordinates"**

$$(x, y) \rightarrow (x, y, 1)$$

Translation (이동)은 Matrix multiplication으로 표현불가능
→ 한개의 차원(값은 1)을 추가해 행렬로 표현

Then $P_h' = T_h P_h$, $P_h' = S_h P_h$, $P_h' = R_h P_h$

$(x, y) \rightarrow (wx, wy, w)$: w 값으로 조절해 2D → 3D 확장

$$\mathbf{T_h} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{S_h} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{R_h} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Homogeneous Coordinates

차원의 표현을 한 차원 증가시켜 표현하는 방법
→ 3x3 행렬로 Linear translation까지 표현하라 (모든 2D 변환은 3x3 행렬로 표현)

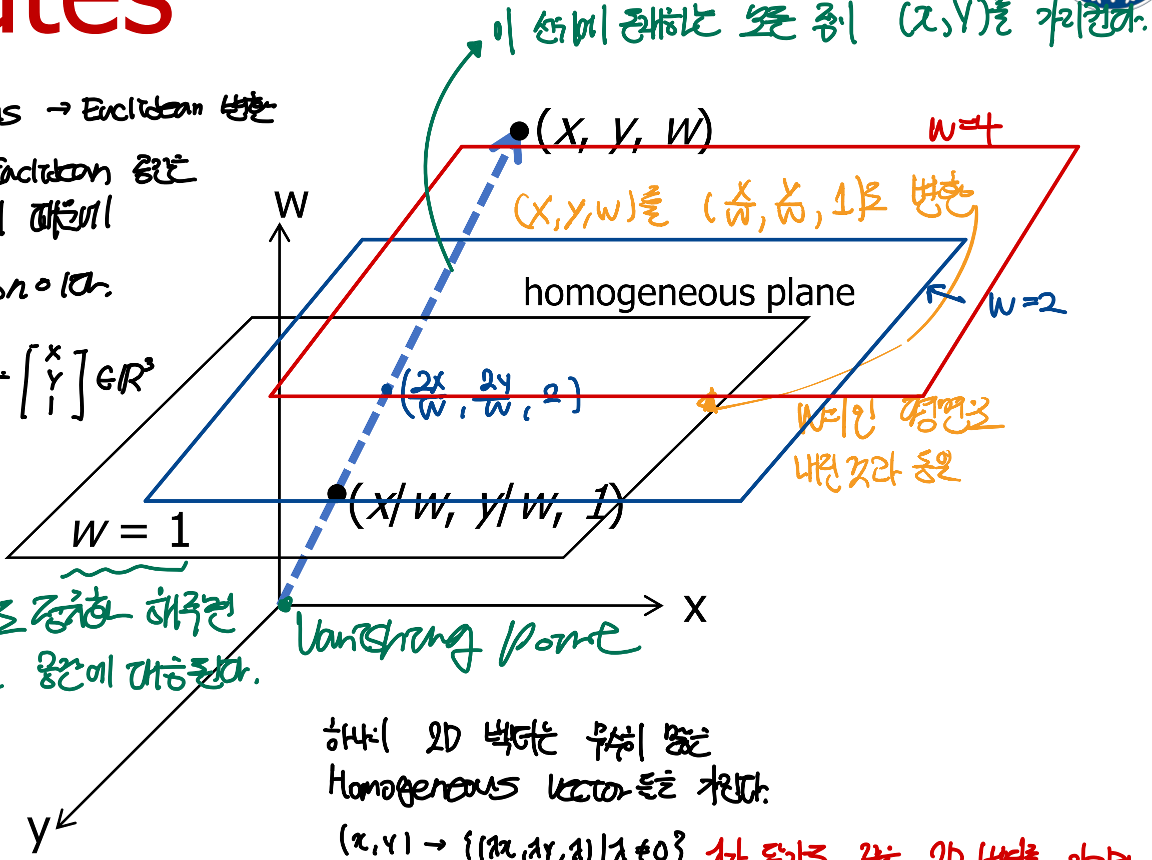
Idea: add one more coordinate:

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

homogeneous image coordinates

Projection : Homogenous → Euclidean 변환
- Homogenous에서 Euclidean 공간은 W=1 평면으로 정의되기 때문에 W=1로의 projection이다.

$$p = \begin{bmatrix} x \\ y \end{bmatrix} \in \mathbb{R}^2 = p = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \in \mathbb{R}^3$$



하나의 2D 벡터는 무수히 많은 Homogeneous vector들을 가진다.

$(x, y) \rightarrow \{(\lambda x, \lambda y, \lambda) \mid \lambda \neq 0\}$ λ가 달라도 같은 2D 벡터를 가리킴

복원(3D → 2D)에서는 현재 대응되는 Homogenous vector가 (x,y,w)라면 원래의 2D는 $\left(\frac{x}{w}, \frac{y}{w}\right)$이다.

↑ Scaling 이라고도 한다.

Converting from homogeneous coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$

$\begin{bmatrix} x/w \\ y/w \\ 1 \end{bmatrix}$

Should be always 1

왜 2D → W=1 일 때로 생각?
W=0 일 때라면 복원할 때 ∞, ∞처럼 됨 → Point of Infinity

Translation

Solution: **homogeneous** coordinates!

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix}$$

Affine transformations : Linear transform + Translation (Affine → not linear)

동형성, 비례유지

2D든 3D든 translation은 선형이 아니다.

→ 우리는 그저 2D→2D에서 translation까지 하나의 행렬로 표현하고픈 뿐이다.

w가 변화할 때:

w↑ : $\frac{x'}{w}$↓, $\frac{y'}{w}$↓ → 작아짐

w↓ : $\frac{x'}{w}$↑, $\frac{y'}{w}$↑ → 커짐

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

↳ Affine transformation 에서도 w가 1로 유지되도록 하기 위해서

affine transformation:

3x3 matrix with last row [0 0 1]

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$$

Linear Transform ← a, b, d, e

c, f → Translation

Basic **Affine** Transformations

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translate

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{s}_x & 0 & 0 \\ 0 & \boldsymbol{s}_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ 1 \end{bmatrix}$$

Scale

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

2D rotation

Where do we go from here?

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$$

affine transformation

what happens when we change this row?

→ 여기가 달라진다.

Projective Transformations (Homography)

3D → 2D 에서 두 시점에 따라 바뀌는 변환

→ Last row ≠ 0 0 1

$$\mathbf{H} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix}$$

≠ Affine transformation

Transformation 결과 distort 발생 가능

8 degree of freedom

Homography : 두 평면 사이의 투영 변환을 시키는 행렬

(2D ⇔ 2D)

Projective matrix는
2D → 2D에서 3차원

↳ 하나의 matrix로
연산하려면 1차원 늘려야 함

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

왜 동차좌표계? projective matrix 때문

→ 유클리드 좌표에서 $x' = \frac{x}{z}$, $y' = \frac{y}{z}$ 를 만들어보자

$$\begin{bmatrix} x \\ y \end{bmatrix} = A \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

↳ 분모에 z가 들어가도록 하기 위해선 필연적으로
A에 z가 포함됨 ← 비선형적

→ Homography는 선형 행렬 하나로 표현 가능

Homography

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} ax+by+c \\ dx+ey+f \\ gx+hy+1 \end{bmatrix}$$

$\neq 1$

w'로 나눠줌

$$\sim \begin{bmatrix} \frac{ax+by+c}{gx+hy+1} \\ \frac{dx+ey+f}{gx+hy+1} \\ 1 \end{bmatrix}$$

What happens when the denominator is **0**?

If $w' = gx+hy+1 = 0$

$\rightarrow x' = y' \approx \infty$

Points at Infinity

$(x_1, x_2, 0)^T$: (x_1, x_2) : a point of infinity in 2D

"This is why we need **homogeneous coordinates**"

Projective → 사영 변환

원근 Vanishing Point를 이미지에 만든 수도 ~

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

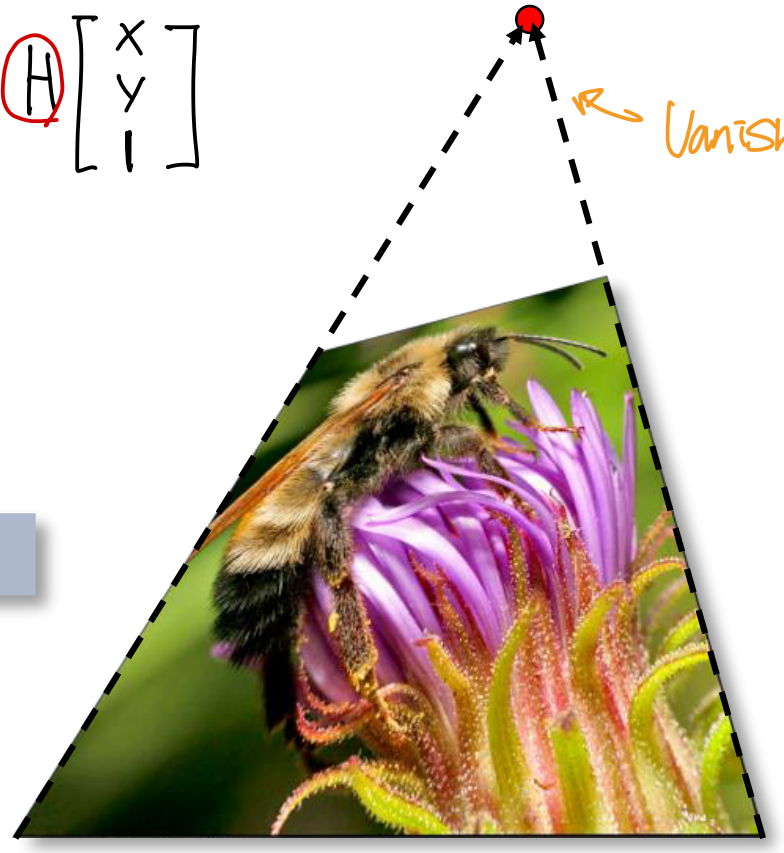
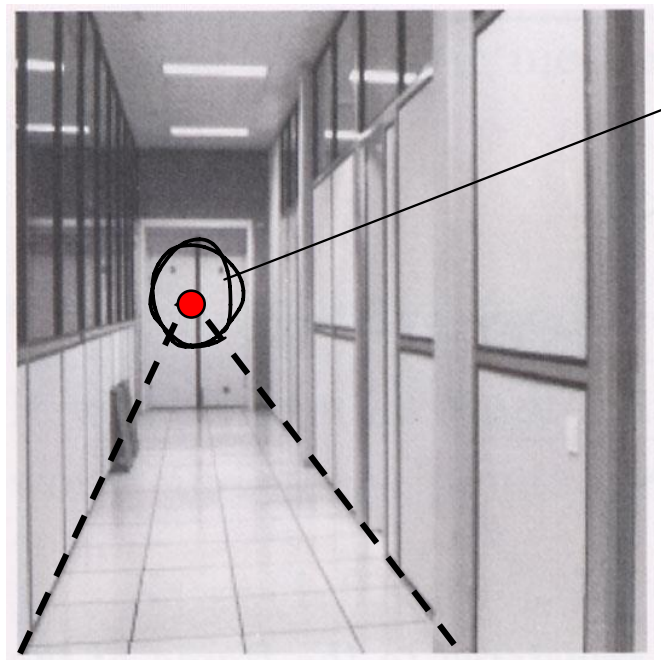
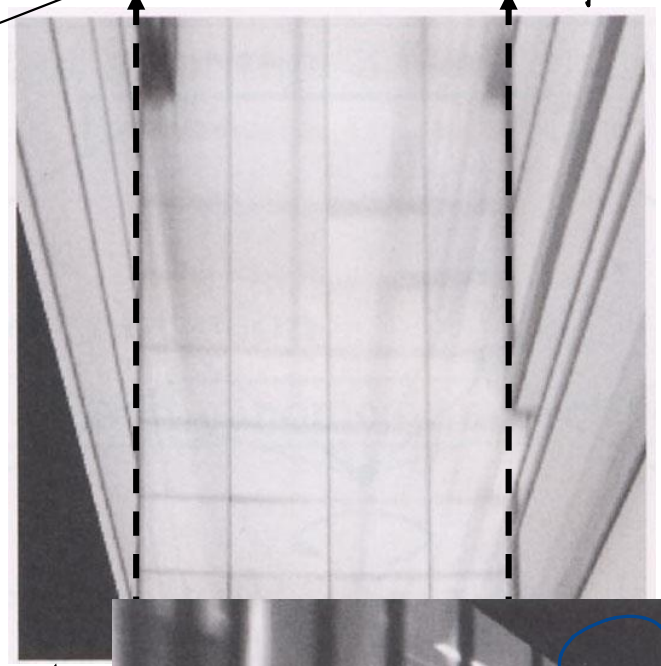


Image **Warping** with Homographies

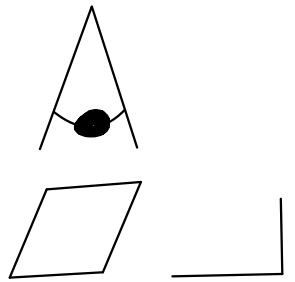
Example



H_1



Point of Infinity



위에서 본 시점으로 변환

H_2

Rectification

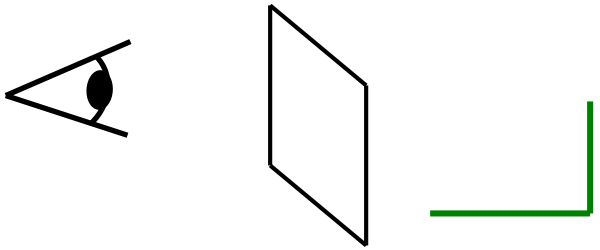


image plane in front

Vanishing Point
: Point of Infinity가 카메라로 투영된 것

black area where no pixel maps to

흰색 경계 영역 → 검은색 영역으로 채워짐

Removing Projective Distortion

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

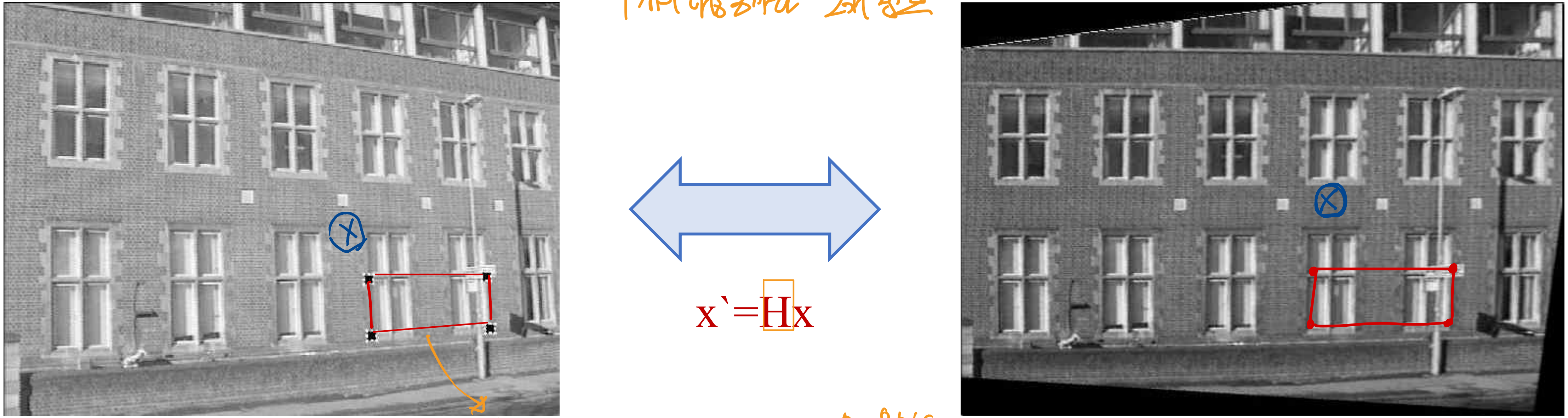
$$x'' = \frac{x'}{w'}, \quad y'' = \frac{y'}{w'}$$

$$x'' = \frac{ax+by+c}{gx+hy+1}, \quad y'' = \frac{dx+ey+f}{gx+hy+1}$$

$$H = \begin{pmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & 1 \end{pmatrix}$$

→ 8 unknowns

- The original image with **perspective distortion**:
- Mapping the 4 window corners to an ideal rectangle (linear equation):
  - 2 equations for each correspondence
  - 4 correspondences and eight equations for 8 unknowns of H



4pts × 2 (2×2) → 8pts (3×3 −1)

2D Image Transformations

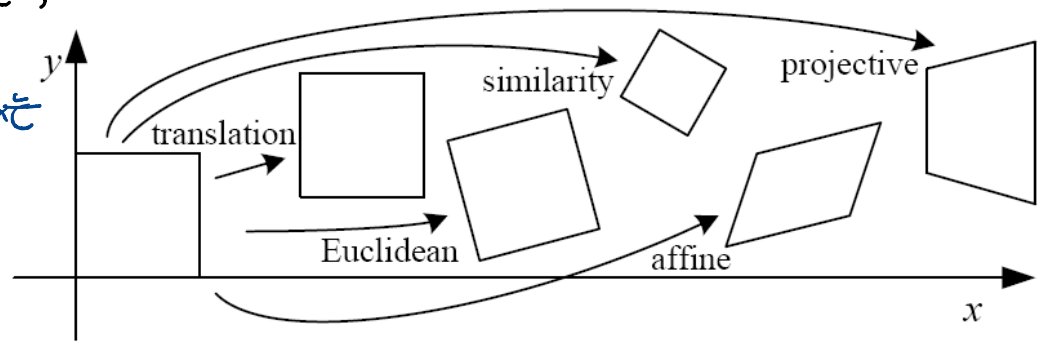
Shearing $\begin{bmatrix} 1 & sh_x \\ sh_y & 1 \end{bmatrix}$

Affine $= \begin{bmatrix} s_x\cos\theta & -s_y\sin\theta + sh_x & t_x \\ s_x\sin\theta + sh_y & s_y\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}$

$R(\theta) = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}$

2D-rotation matrix

DoF = 1



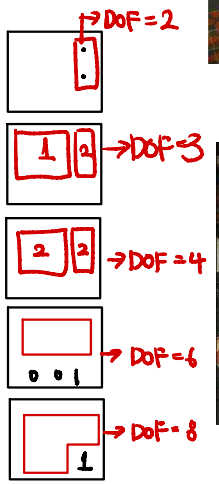
Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	2	orientation + ⋯	
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	3	lengths + ⋯	
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	4	angles + ⋯	
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times3}$	6	parallelism + ⋯	
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times3}$	8	straight lines	

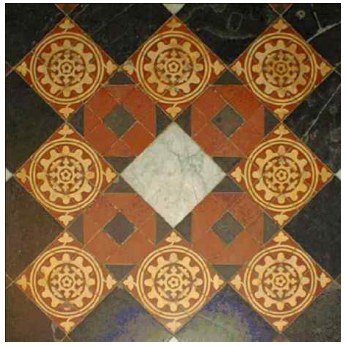
Rotation + translation

Scaling + Rotation + Translation

DoF = 1 DoF = 1 DoF = 2

affine ← similarity + shearing





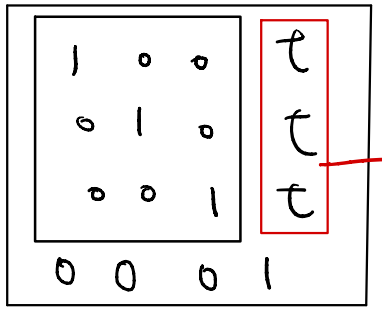
Euclidean

Affine

Projective

We are ready to learn 3D Transformation!

3D Transformation



DOF=3

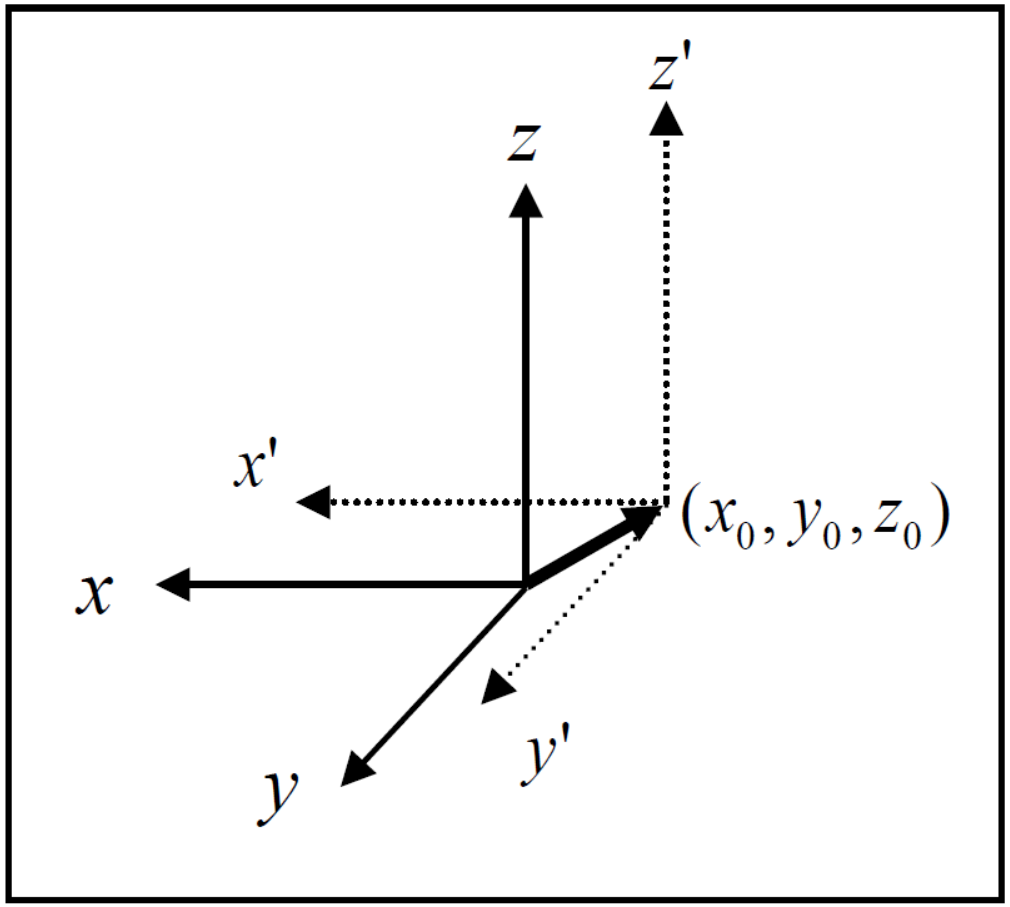
- Translation (3D to 3D

$$(x, y, z) \rightarrow (x_p, y_p, z_p)$$

$$\begin{pmatrix} x_p \\ y_p \\ z_p \end{pmatrix} = \begin{pmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{pmatrix} \Rightarrow \quad \tilde{\mathbf{v}}_\mathbf{p} = \mathbf{T}\tilde{\mathbf{v}}$$

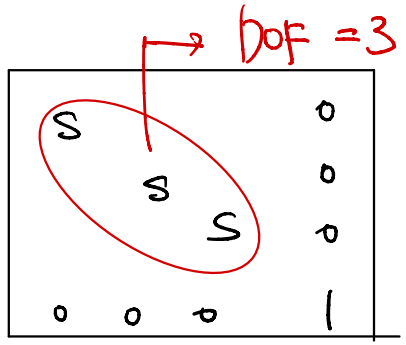
넣은 변화

$$\text{where } \mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{T}^{-1} = \begin{bmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



3D Transformation

- Scaling $(x, y, z) \rightarrow (x_p, y_p, z_p)$



$$\begin{pmatrix} x_p \\ y_p \\ z_p \end{pmatrix} = \begin{pmatrix} k_x x \\ k_y y \\ k_z z \end{pmatrix} \Rightarrow \quad \tilde{\mathbf{v}}_\mathbf{p} = \mathbf{S}\tilde{\mathbf{v}}$$

$$\text{where} \quad \mathbf{S} = \begin{bmatrix} k_x & 0 & 0 & 0 \\ 0 & k_y & 0 & 0 \\ 0 & 0 & k_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{S}^{-1} = \begin{bmatrix} 1/k_x & 0 & 0 & 0 \\ 0 & 1/k_y & 0 & 0 \\ 0 & 0 & 1/k_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3D Transformation

- Rotation (x-axis)

$$(x, y, z) \rightarrow (x', y', z')$$

$$x' = x$$

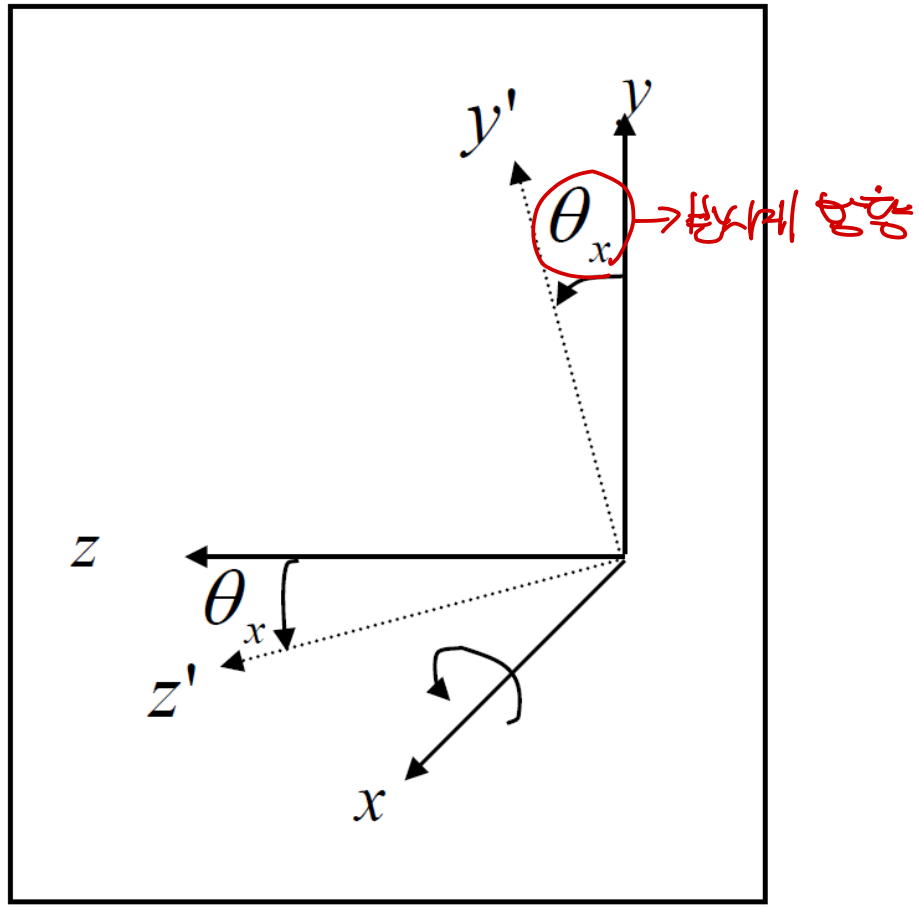
$$\begin{pmatrix} y' \\ z' \end{pmatrix} = \begin{pmatrix} \cos\theta_x & \sin\theta_x \\ -\sin\theta_x & \cos\theta_x \end{pmatrix} \begin{pmatrix} y \\ z \end{pmatrix}$$

$$\Rightarrow \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & \sin\theta_x \\ 0 & -\sin\theta_x & \cos\theta_x \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$\Rightarrow$ Using homogeneous coordinates

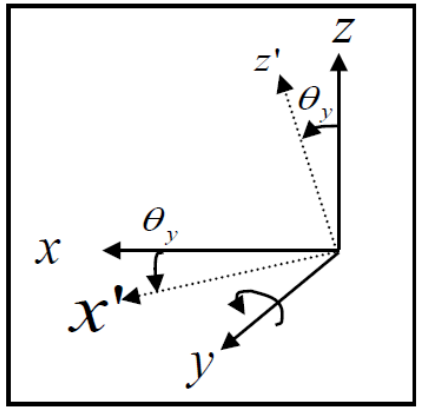
$$\tilde{\mathbf{v}}_{\mathbf{p}} = \mathbf{R}_{\mathbf{x}} \tilde{\mathbf{v}}$$

where $$\mathbf{R}_{\mathbf{x}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_x & \sin\theta_x & 0 \\ 0 & -\sin\theta_x & \cos\theta_x & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

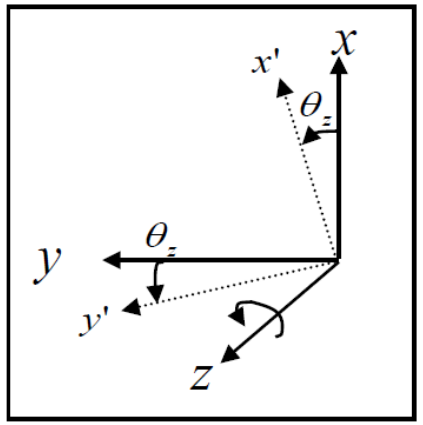


3D Transformation

- Rotation
(y-axis & z-axis)

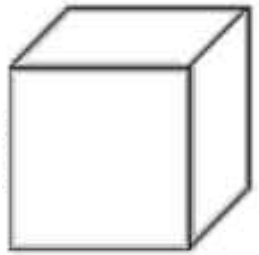


$$\tilde{\mathbf{v}}_p = \mathbf{R}_y \tilde{\mathbf{v}} \quad \text{where} \quad \mathbf{R}_y = \begin{pmatrix} \cos\theta_y & 0 & -\sin\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta_y & 0 & \cos\theta_y & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

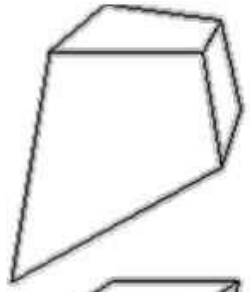


$$\tilde{\mathbf{v}}_p = \mathbf{R}_z \tilde{\mathbf{v}} \quad \text{where} \quad \mathbf{R}_z = \begin{pmatrix} \cos\theta_z & \sin\theta_z & 0 & 0 \\ -\sin\theta_z & \cos\theta_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

3D Transformations



Rotation Matrix's Dof = 3
→ x축, y축, z축에 대한 각각 회전 변수 생김
2D에서는 only z축



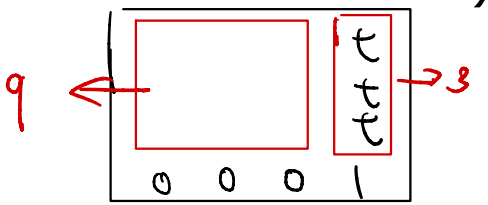
- **Projective** (15 dof)

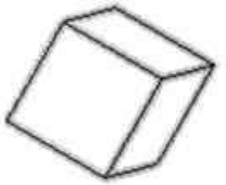
$$\begin{bmatrix} A & \boldsymbol{t} \\ \boldsymbol{v}^T & 1 \end{bmatrix}$$

$A \in \mathbb{R}^{3\times3}$, $t \in \mathbb{R}^{3\times1}$, $v \in \mathbb{R}^{3\times1}$

- **Affine** (12 dof = 9 rotation + 3 translation)

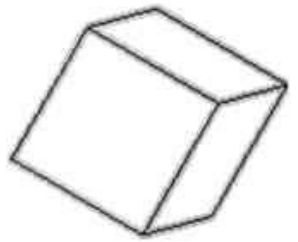
$$\begin{bmatrix} A & \boldsymbol{t} \\ \mathbf{0}^T & 1 \end{bmatrix}$$





- **Similarity** (7 dof = 1 scaling + 3 rotation + 3 translation)

$$\begin{bmatrix} sR & \boldsymbol{t} \\ \mathbf{0}^T & 1 \end{bmatrix}$$



- **Euclidean** (6 dof = 3 rotation + 3 translation)

$$\begin{bmatrix} R & \boldsymbol{t} \\ \mathbf{0}^T & 1 \end{bmatrix}$$

$R \in \mathbb{R}^{3\times3}$, $t \in \mathbb{R}^{3\times1}$

Perspective Projection Model

Camera: 3D (World coordinates: X,Y,Z) → 2D mapping (Image coordinates : x,y)

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} \xrightarrow{\text{정리}} \begin{pmatrix} \frac{fX}{Z} \\ \frac{fY}{Z} \\ 1 \end{pmatrix}$$

$$(X_c, Y_c, Z_c)^T \rightarrow (x, y) = \left(\frac{fX_c}{Z_c}, \frac{fY_c}{Z_c}\right)$$

$$w\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}$$

초점거리

$$\vec{x} = P\vec{X_c}$$

원래는 너비가 1이다. $P = diag(f, f, 1)[I|0]$

$$\frac{X}{Z} = \frac{x}{f}$$

$$\Rightarrow x = \frac{fX}{Z}$$

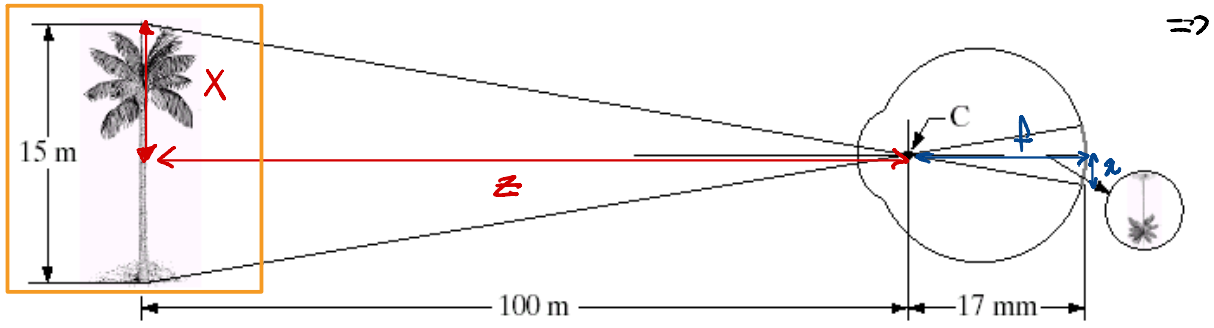
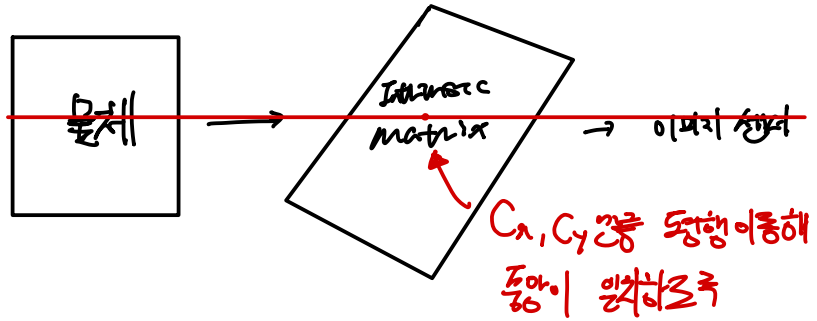


FIGURE 2.3 Graphical representation of the eye looking at a palm tree. Point C is the optical center of the lens.

Camera Intrinsic Parameters

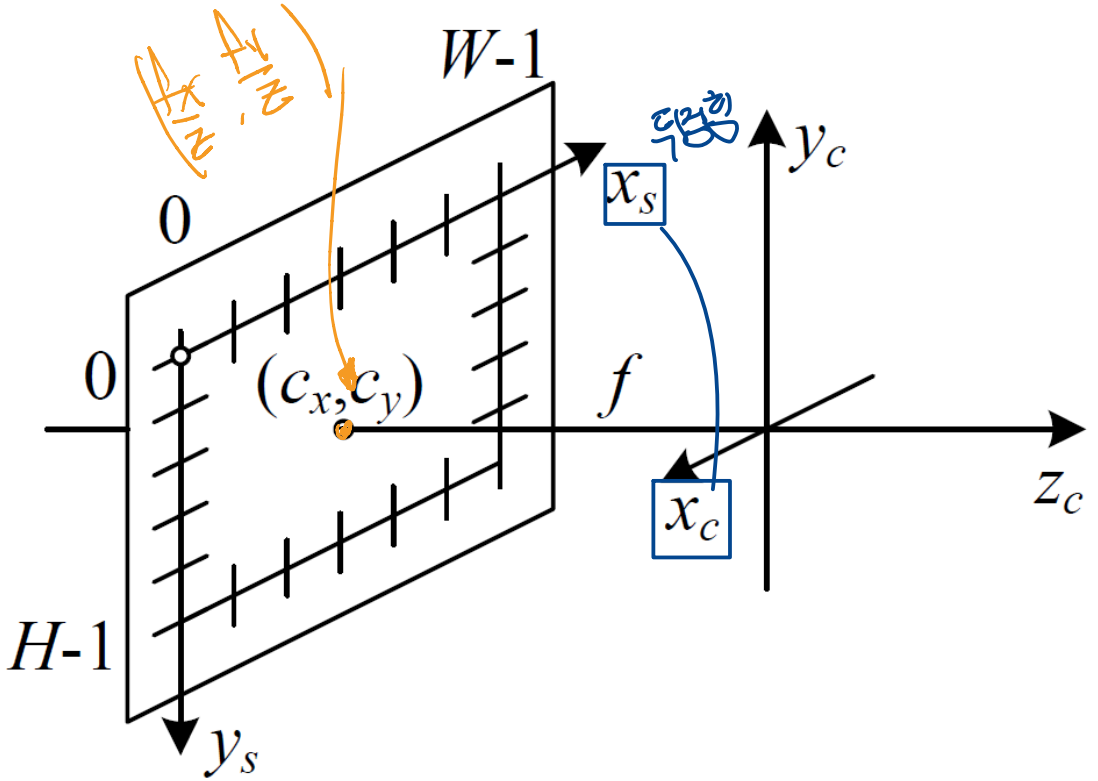
→ 카메라 내부 변환 보정



Principal point offset:

$$(X_c, Y_c, Z_c)^T \rightarrow (x, y) = \left(\frac{fX_c}{Z_c} + c_x, \frac{fY_c}{Z_c} + c_y\right)$$

$$w\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} fX_c + Z_c c_x \\ fY_c + Z_c c_y \\ Z_c \end{bmatrix} = \begin{bmatrix} f & & c_x & 0 \\ & f & c_y & 0 \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}$$



➔ $\vec{x} = K[I\ \mathbf{0}]\overrightarrow{X_c}$ where $K = \begin{bmatrix} f & & c_x \\ & f & c_y \\ & & 1 \end{bmatrix}$

↳ $\begin{bmatrix} f & & c & 0 \\ & f & c & 0 \\ & & 1 & 1 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{pmatrix} fx + cz \\ fy + cz \\ z \end{pmatrix} = \begin{pmatrix} \frac{fx}{z} + c \\ \frac{fy}{z} + c \\ 1 \end{pmatrix}$ "Camera **Intrinsic** Parameters"

Camera Rotation and Translation

A rigid transformation between the camera coordinate frame and the world coordinate

World to Camera

$$X_c = \begin{bmatrix} R & T \\ 0^T & 1 \end{bmatrix} X_w$$

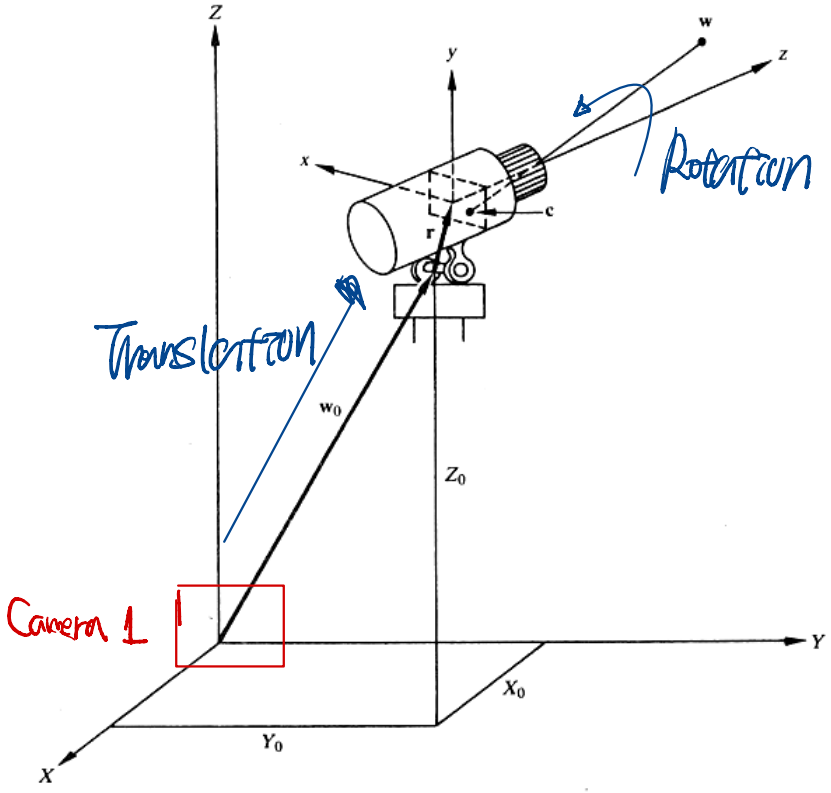
$$A^{-1} = \begin{bmatrix} R^T & -R^T T \\ 0 & 1 \end{bmatrix}$$

A^{-1} → Camera to World

➔ $\widetilde{X_c} = R\widetilde{X_w} + T$

Rotation

$\Rightarrow R^T = R^{-1}$



Camera Coordinate

- 카메라가 가지는 좌표계
  → 실제 좌표계와 다르다.

Extrinsic Parameters : Real World와 Camera의 대응

실제 좌표계를 카메라 좌표계로 변환해야한다.

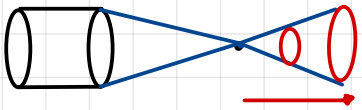
Rotation : 실세계 좌표를 카메라 좌표축에 맞게 돌린 (카메라가 실세계를 보는 각도)

Translation : 실세계 좌표의 원점이 카메라 중심의 좌표로 이동 (카메라 위치 좌표)

Intrinsic Parameters : 카메라 내부에서 3D → 2D로 어떻게?

f : 렌즈 – 이미지 센서 사이의 거리 f_x↑ : x축 확대, f_y↑ : y축 확대

C : 이미지의 중심과 렌즈 중심이 일치하도록 한다.



f↑ → 이미지↑

Real World to 2D Camera Image

X_w : real world $\in \mathbb{R}^3$, X_c : Camera view $\in \mathbb{R}^3$ $\quad K = \begin{bmatrix} f & & C_x \\ & f & C_y \\ & & 1 \end{bmatrix}$

$X_c = RX_w + T$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = K[I\ 0] \cdot \begin{bmatrix} X \\ 1 \end{bmatrix}$$

↳ Translation을 하나의 matrix로 → Homogenous

$x'' = \frac{x'}{z'}$, $y'' = \frac{y'}{z'}$ ← 3D를 2D로 projection

$\underbrace{K[R|T]}_{P} = K \cdot [R|T]$ ← Calibration "P"를 찾는 과정

(K : 3×3, [R|T] : 3×4)

→ 카메라 상수로 변한다

Perspective Projection Model

$$\widetilde{X_c} = R\widetilde{X_w} + T$$

$$\vec{x} = K[I\ \mathbf{0}]\overrightarrow{X_c} \rightarrow \vec{x} = K[R \mid \mathbf{t}]\overrightarrow{X_w}$$

Camera Matrix: $P = K[R \mid \mathbf{t}]$

Camera calibration:

Computational procedure used to obtain the **camera parameters**. (Linear Equation)

Inverse Perspective Transformation?

$$(X_c, Y_c, Z_c)^T \rightarrow (x, y) = \left(\frac{fX_c}{Z_c}, \frac{fY_c}{Z_c}\right)$$

$$w\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}$$

2D → 3D

Inverse Perspective Transformation

- Not possible to recover (X, Y, Z) from (x, y) → 왜냐면 그림에서 처음 어느 위치에 있더라도 같은 V(z)가 생성될 수 있다.
- For free variable z, ↔ 종속변수

$$(X_c, Y_c, Z_c) \rightarrow v(z) = \left(\frac{zx}{f}, \frac{zy}{f}, z\right)$$ ← 근데 확산할 수 있다.

→ 2D to 3D 변환 with one Camera 불가능

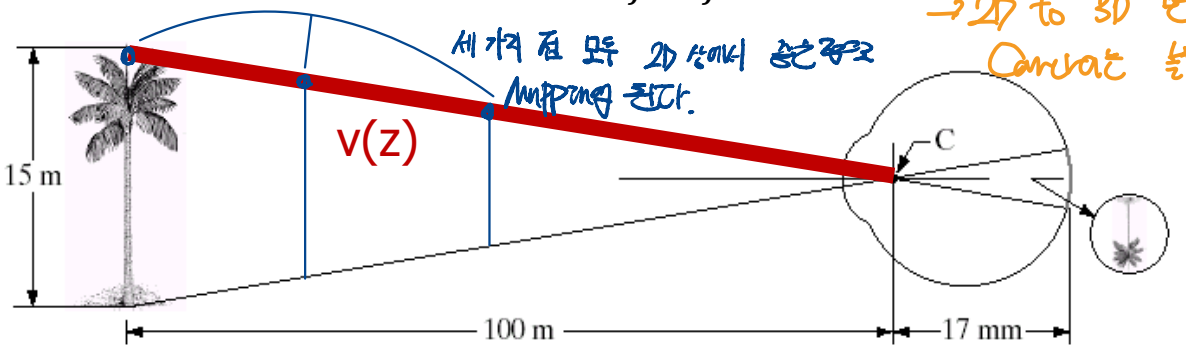


FIGURE 2.3 Graphical representation of the eye looking at a palm tree. Point *C* is the optical center of the lens.

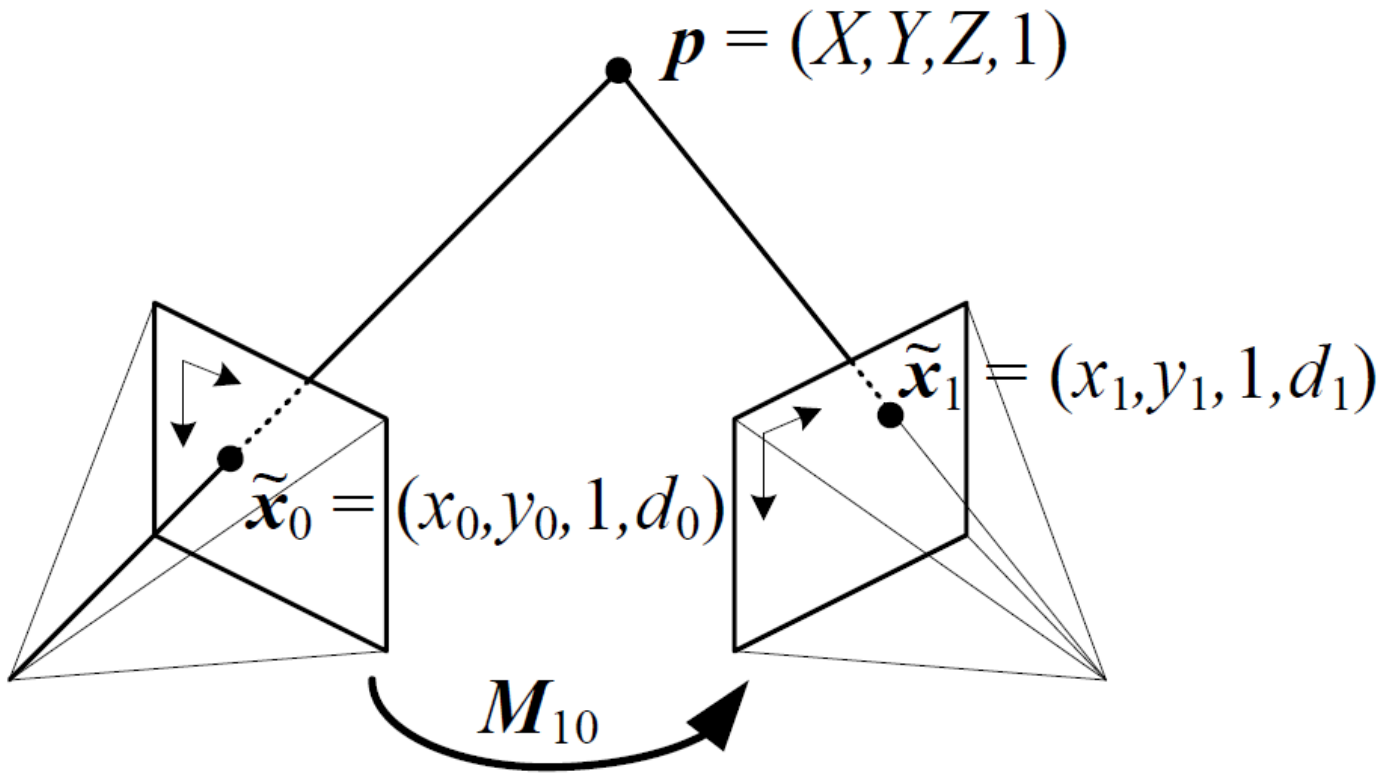
Stereo Vision

→ for perfect recover (x,y,z) from (x,y)

- Require multiple views to reconstruct 3D (at least 2 views).

How to do it?

Coming soon!



Vanishing Points

- Projection of a **point at infinity**
- Any two parallel lines (in 3D) have the same **vanishing point.**

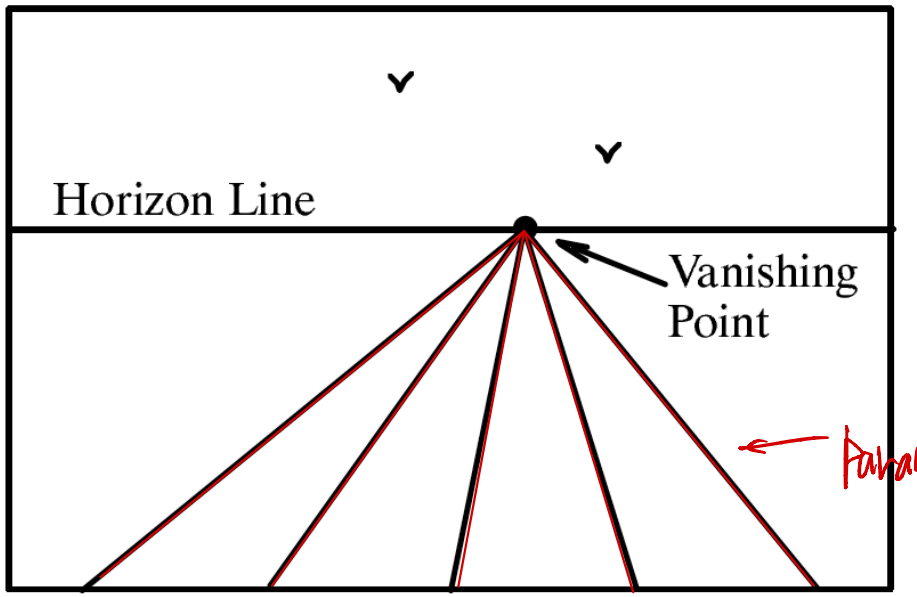
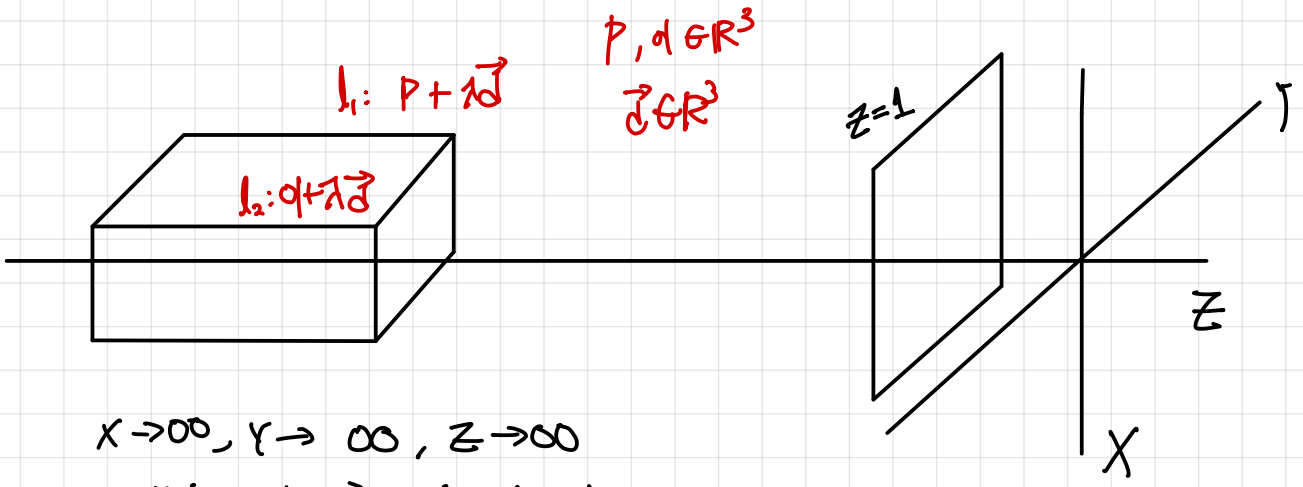


Figure 23.4
A perspective view of a set of parallel lines in the plane. All of the lines converge to a single vanishing point.



$X \to \infty$, $Y \to \infty$, $Z \to \infty$

$\Rightarrow$ 방향 벡터 $\vec{d} = (d_x, d_y, d_z)$ 이용

$l_1 = (x_1 + \lambda d_x,\ y_1 + \lambda d_y,\ z_1 + \lambda d_z)$

$l_2 = (x_2 + \lambda d_x,\ y_2 + \lambda d_y,\ z_2 + \lambda d_z)$

$$P = \begin{bmatrix} a & 0 & t_1 & 0 \\ 0 & b & t_2 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$l_1 : \begin{bmatrix} x_1' \\ y_1' \\ w_1 \end{bmatrix} = P \cdot \begin{bmatrix} x_1 + \lambda d_x \\ y_1 + \lambda d_y \\ z_1 + \lambda d_z \\ 1 \end{bmatrix}$$

$$= \begin{cases} x_1' = a x_1 + \lambda a d_x + t_1 z_1 + \lambda t_1 d_z \\ y_1' = b y_1 + \lambda b d_y + t_2 z_1 + \lambda t_2 d_z \\ w_1 = z_1 + \lambda d_z \end{cases}$$

$$\begin{bmatrix} x + \lambda d_x \\ y + \lambda d_y \\ z + \lambda d_z \\ 1 \end{bmatrix}$$: 유한한 점에 대응되지만 $\lambda \to \infty$ 으로 무한대로 나타냄

$$l_2 : \begin{bmatrix} x_2' \\ y_2' \\ w_2 \end{bmatrix} = P \cdot \begin{bmatrix} x_2 + \lambda d_x \\ y_2 + \lambda d_y \\ z_2 + \lambda d_z \\ 1 \end{bmatrix}$$

$$= \begin{cases} x_2' = a x_2 + \lambda a d_x + t_1 z_2 + t_1 d_z \\ y_2' = b y_2 + \lambda b d_y + t_2 z_2 + \lambda t_2 d_z \\ w_2 = z_2 + \lambda d_z \end{cases}$$

$$\begin{bmatrix} d_x \\ d_y \\ d_z \\ 0 \end{bmatrix}$$: 가상의 점 ($w=0$)으로 d_x, d_y, d_z의 방향으로 무한대로 갈 것이다.

Vanishing point $(x_1', y_1') = (x_2', y_2')$

$$x_1 \Rightarrow \lim_{\lambda \to \infty} \frac{a x_1 + t_1 z_1 + \lambda(a d_x + t_1 d_z)}{\lambda d_z + z_2} = \frac{a d_x}{d_z} + t_1$$

$$x_2 \Rightarrow \lim_{\lambda \to \infty} \frac{a x_2 + t_1 z_2 + \lambda(a d_x + t_1 d_z)}{\lambda d_z + z_2} = \frac{a d_x}{d_z} + t_1$$

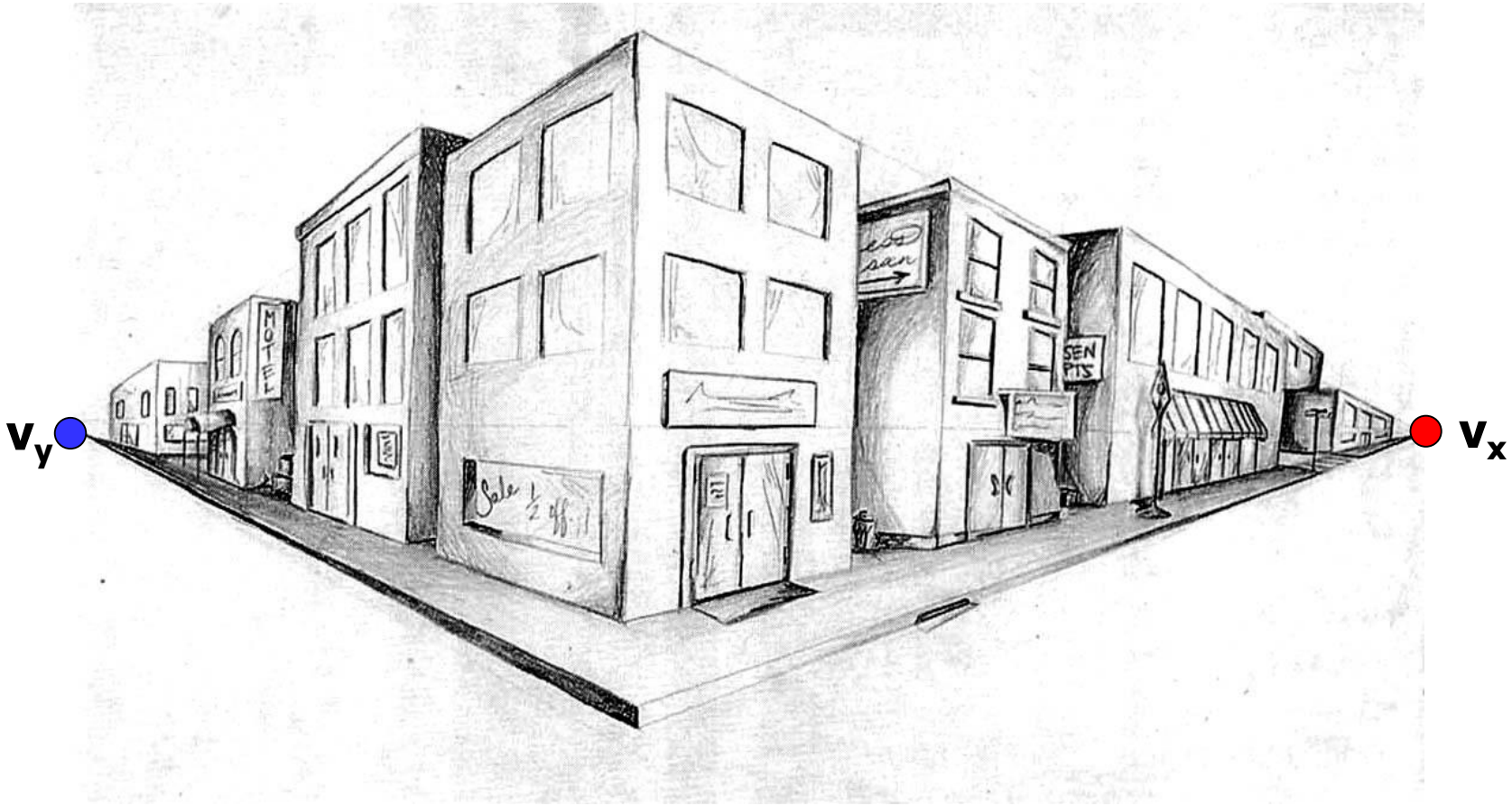
동일

One-point Perspective

→ 정의

Two-point Perspective

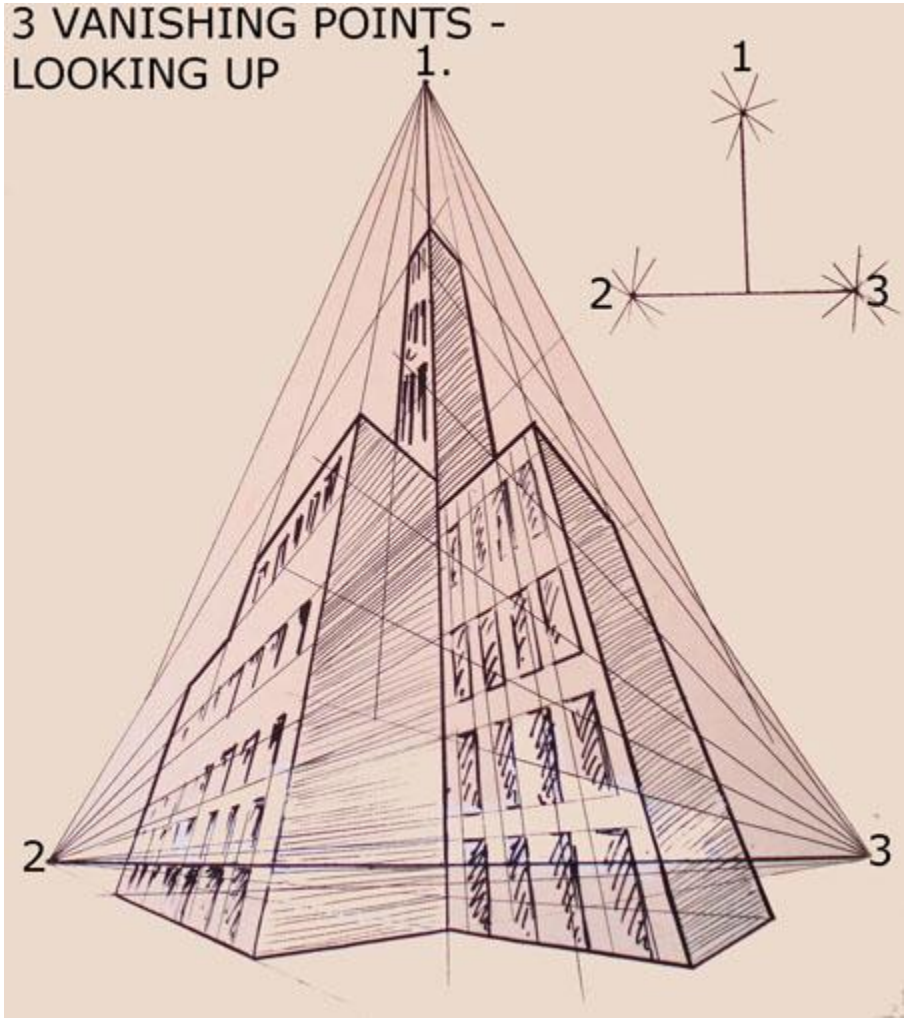
→ 건물이 비스듬히



Three-point Perspective

→ 고층건물 등, 위/아래에서 찍은 사진

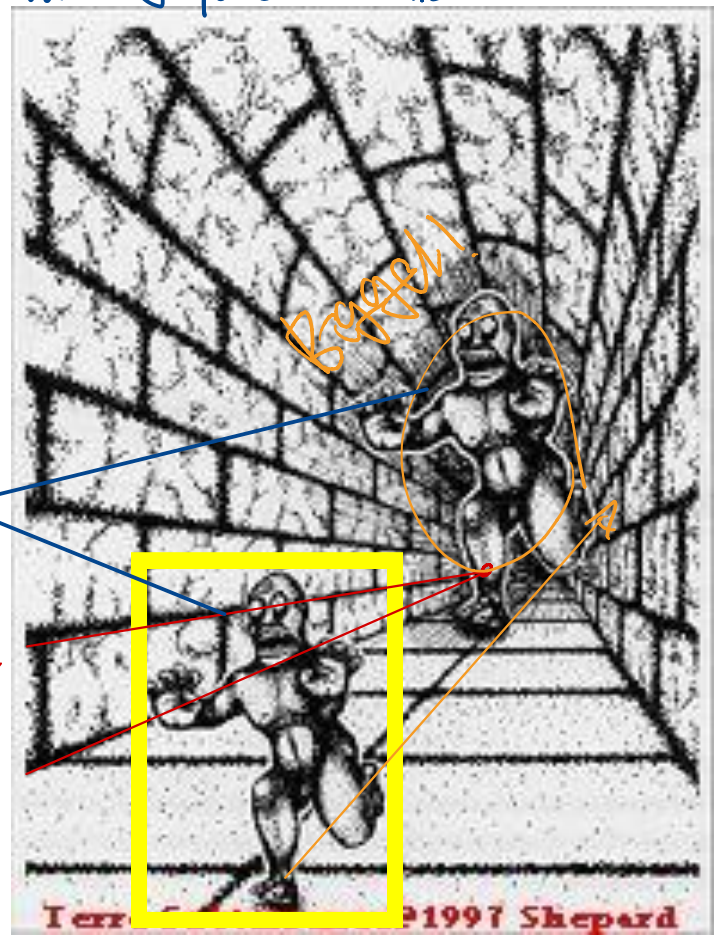
→ X, Y, Z 축이 모두 Vanishing Point를 가짐



Characteristics of Vanishing Points

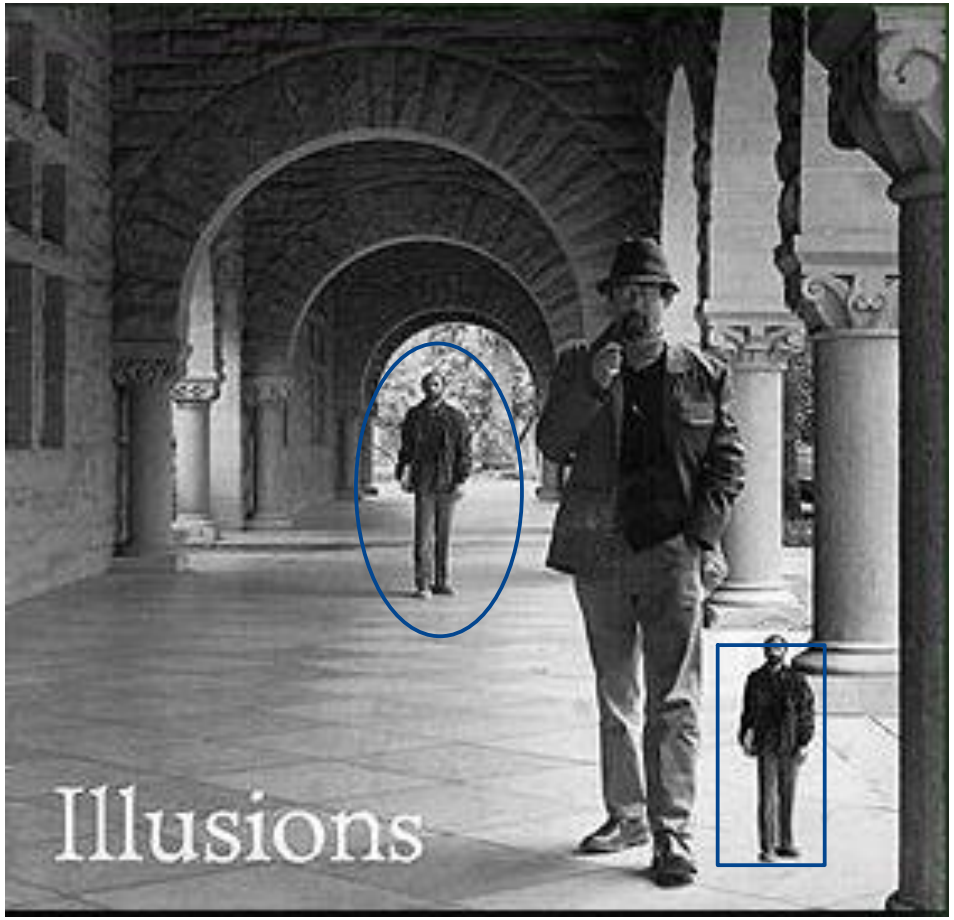
→ Vanishing point를 통해 Object의 size를 예측할 수 있다.

Vanishing point에 가까울수록 크게 보임 → 원근감 때문에 멀리있는 것이 가까이보다 작다.

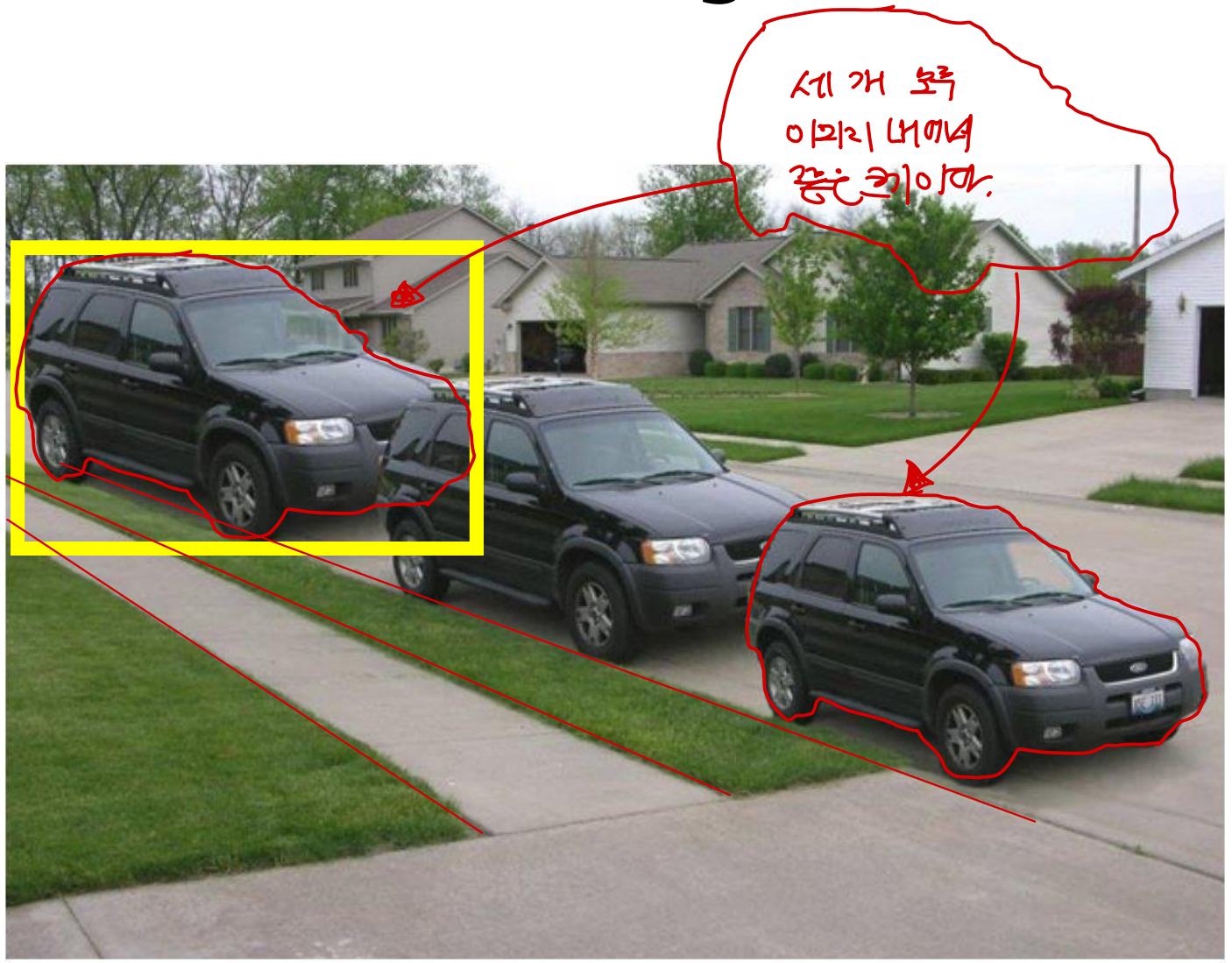


같은 size인데 소실점에 가까울수록 크게 그려짐

Real space에서는 Parallel

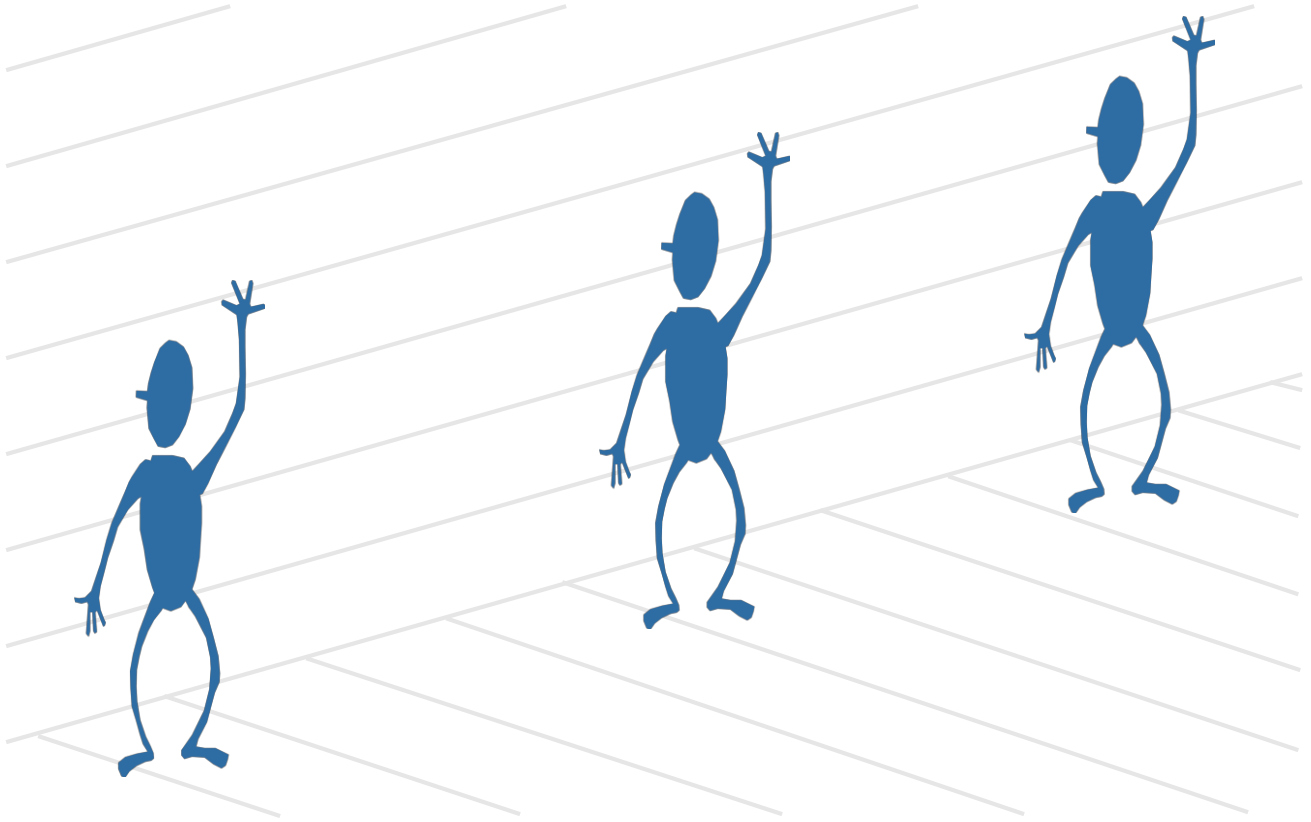


Characteristics of Vanishing Points



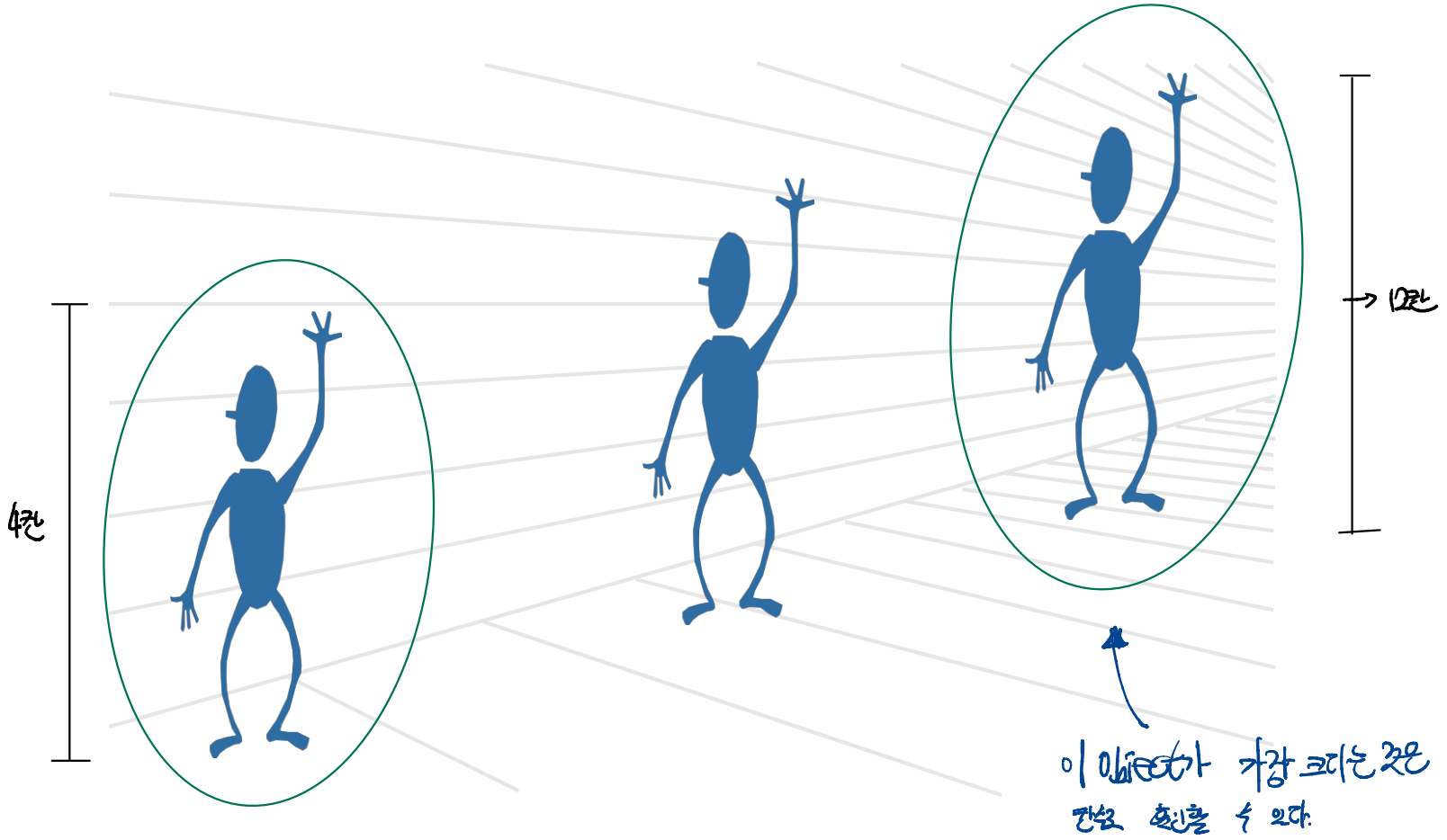
원래는 같은 크기
→ Vanishing point 때문에
크기 차이가 나는 것처럼 보인다.

Perspective Cues

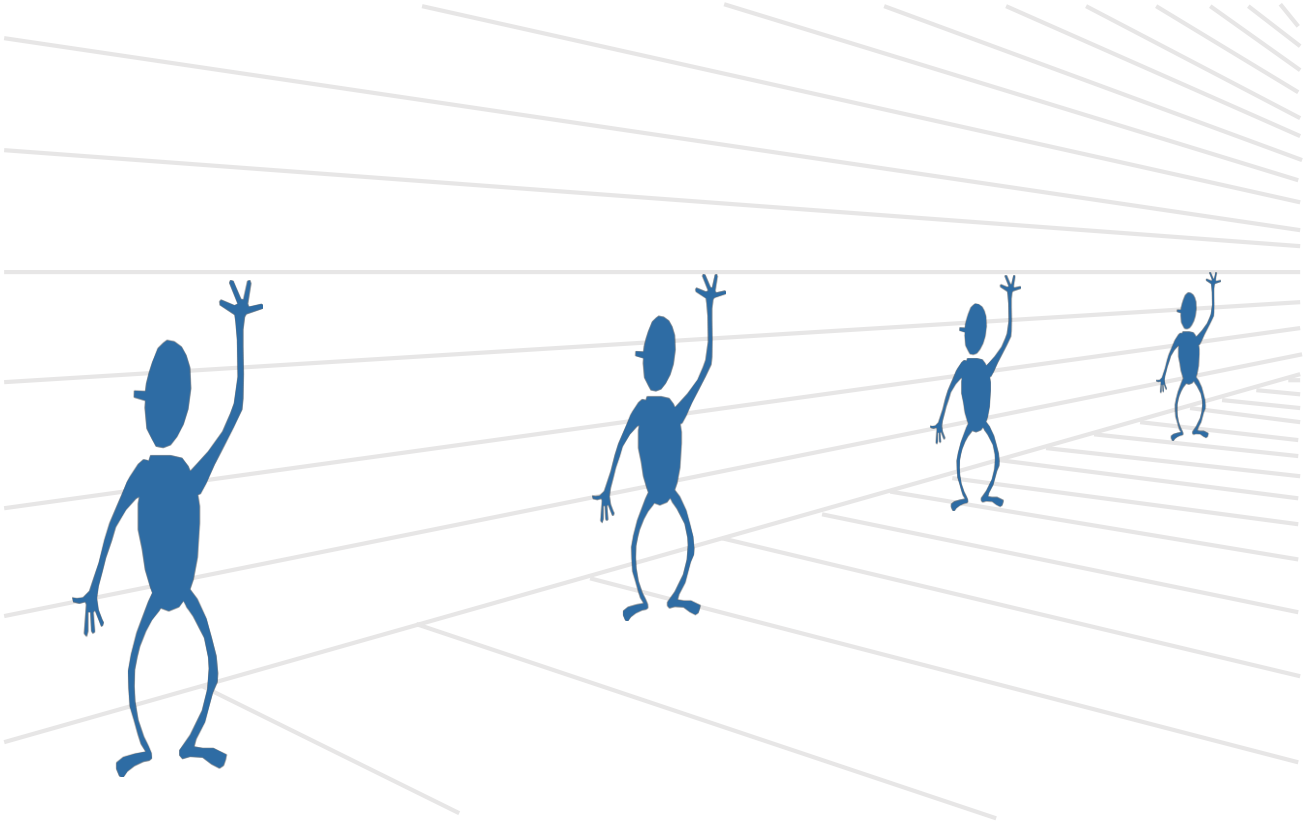


Perspective Cues

→ 2D Image에서 3D 상황을 파악할 수

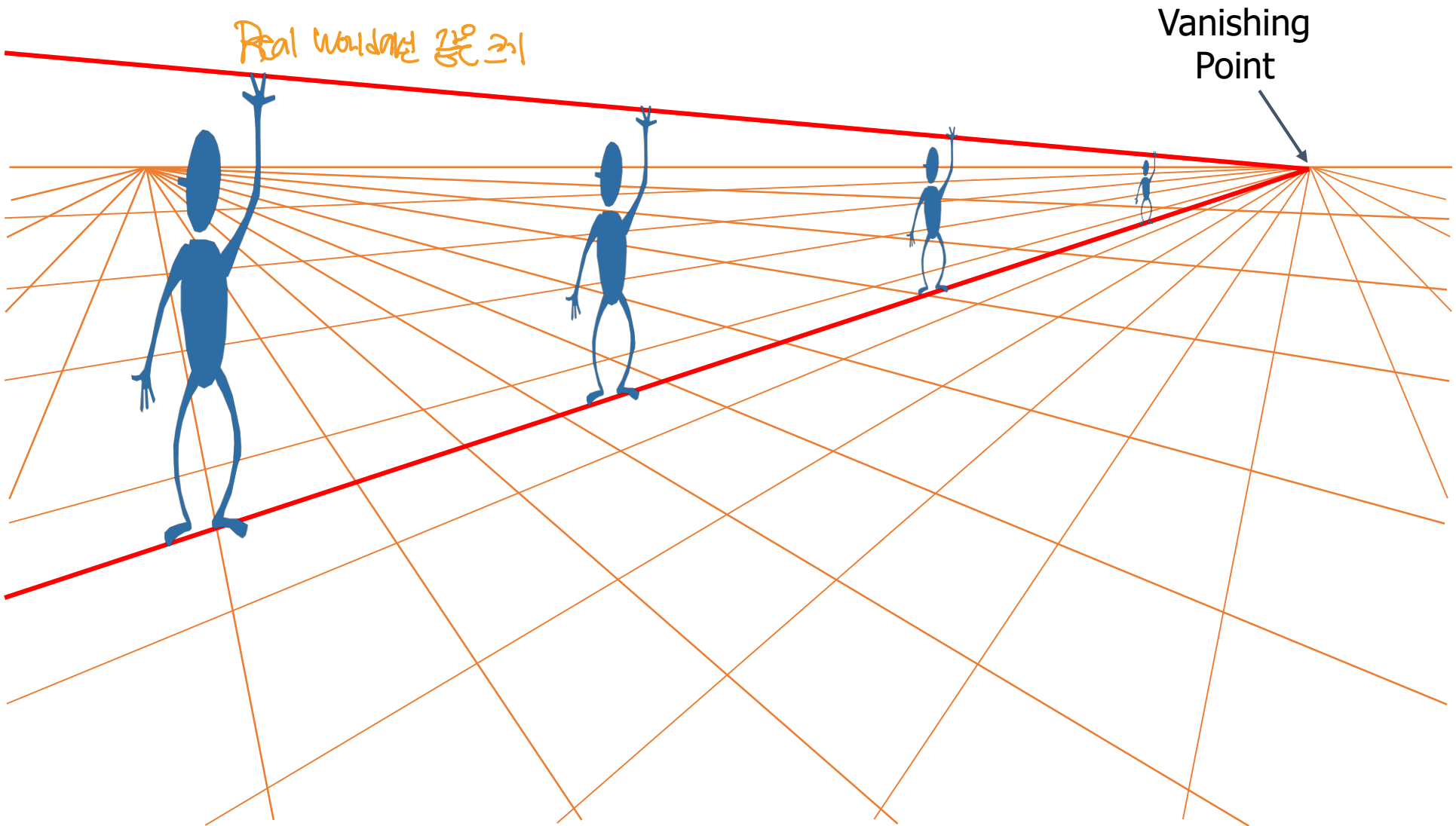


Perspective Cues

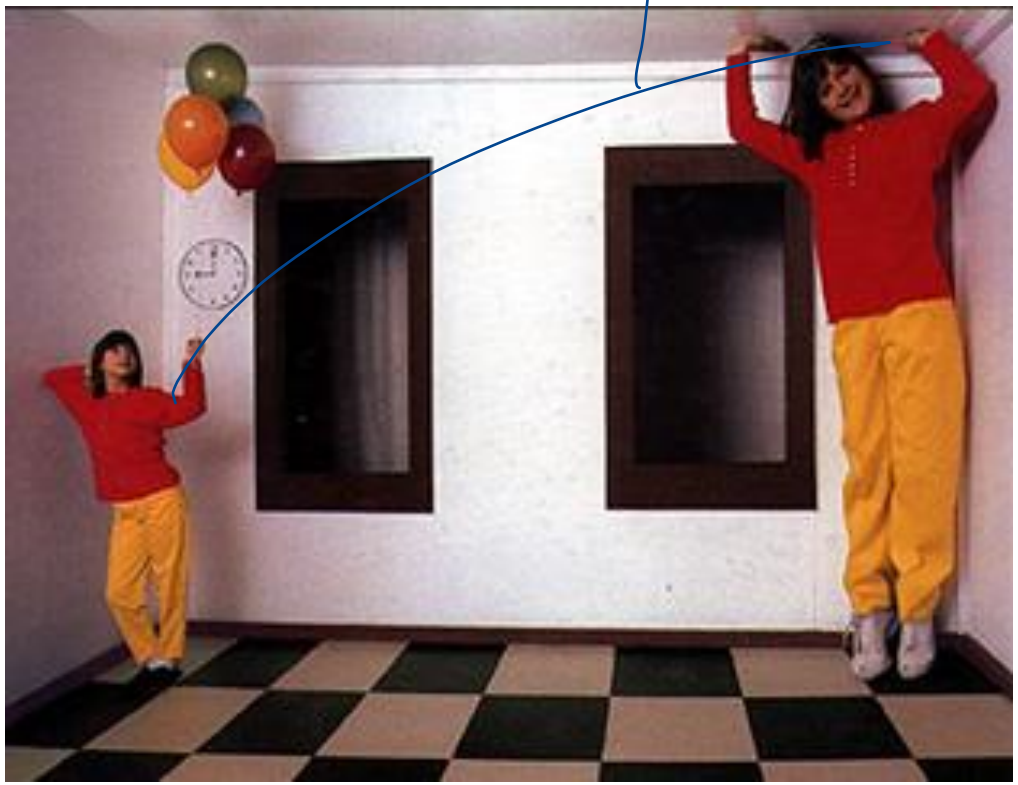


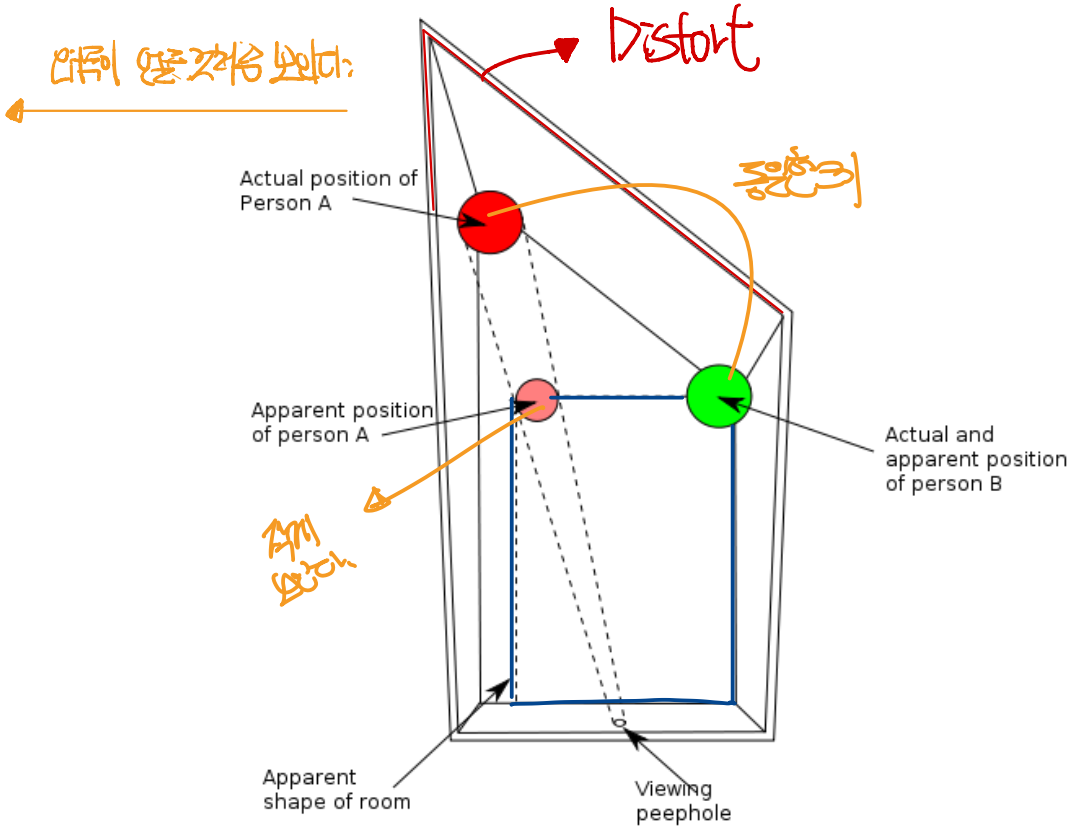
Human Visual System

Comparing Heights



Ames Room

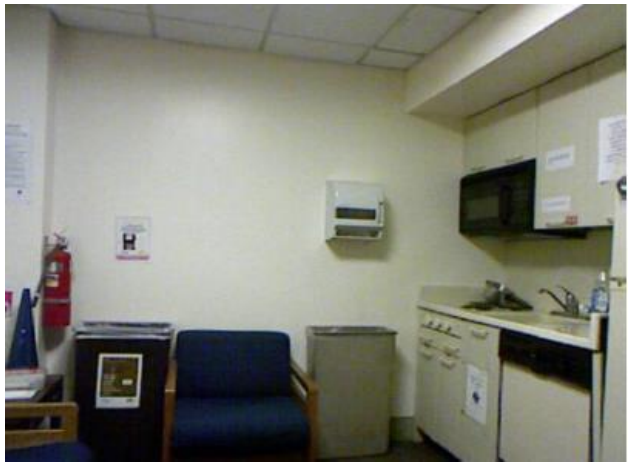


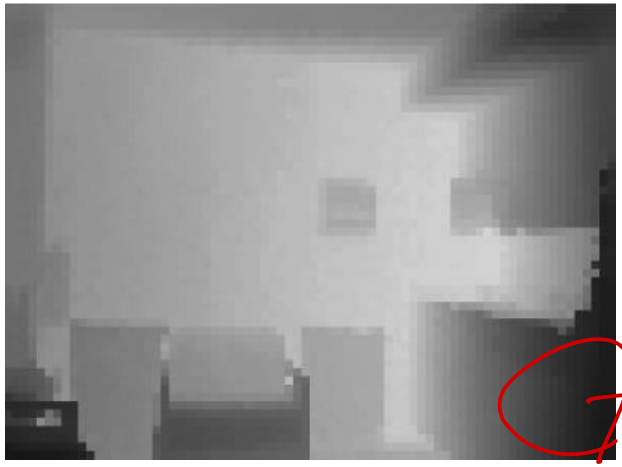

Actual position of Person A
Apparent position of person A
Actual and apparent position of person B
Apparent shape of room
Viewing peephole

Perspective Tricks in Films

Single Image Depth Estimation

- Estimates depth information given a single RGB image
- Ill-posed problem because of lack of geometric information
- **Deep Learning** Based Approaches: Make the model guess the depth based on the **data**





An example of RGB-Depth pair [NYUDepthV2[1)] Dataset]

1) https://cs.nyu.edu/~silberman/datasets/nyu_depth_v2.html

Single Image Depth Estimation

Recent Performance [1]

[1] Boosting Monocular Depth Estimation Models to High-Resolution via Content-Adaptive Multi-Resolution Merging. CVPR 2021.

From Pinhole to Real Cameras

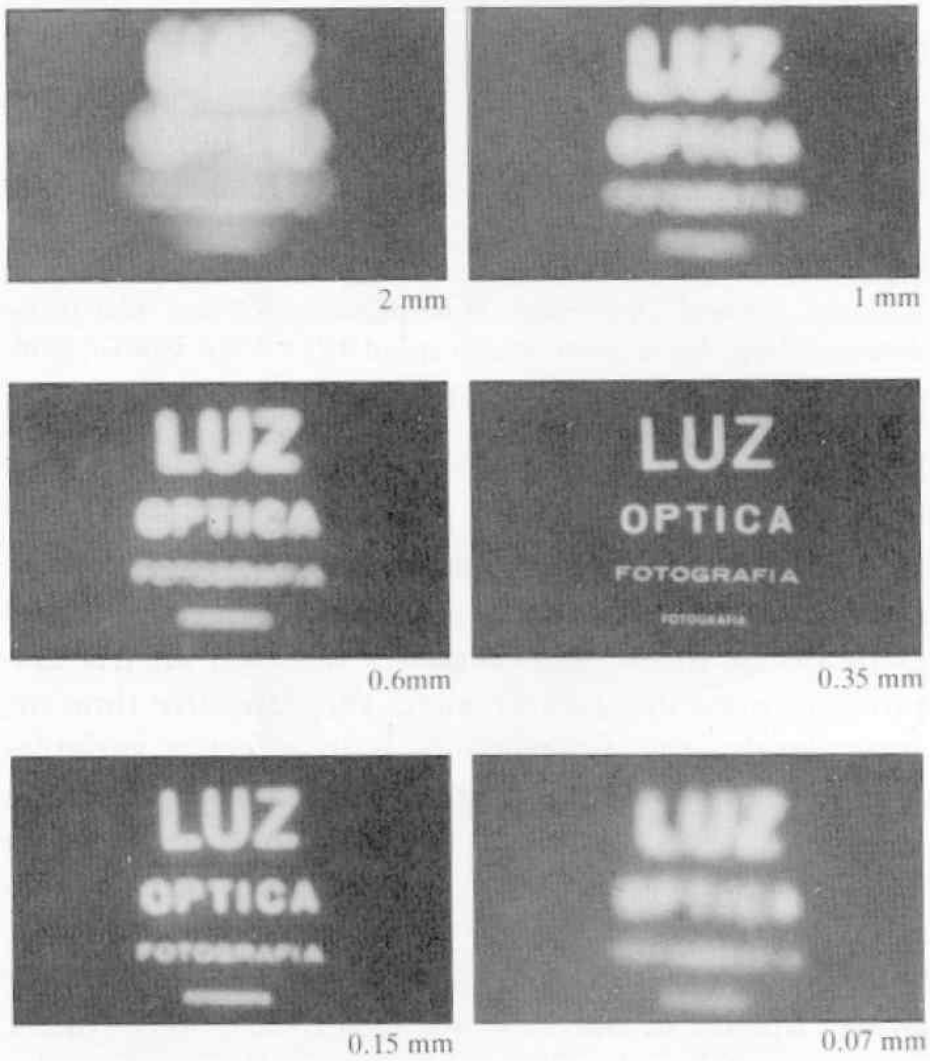
Ideal: len size = 0
Real: len size ≠ 0

Pinhole

- Ideal: a single ray reaching each point in the image
  - hole size가 매우 작아야함
- Real: a cone of light rays reaching each point in the image
  - 한 점에 여러 ray가 다가온다.

Pinhole is too big!

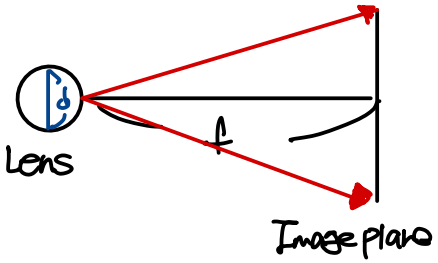
- Many directions are averaged, **blurring** the image
  - Cone of light 때문에 한 점에 인접한 점의 정보가 섞여서 blur된다.



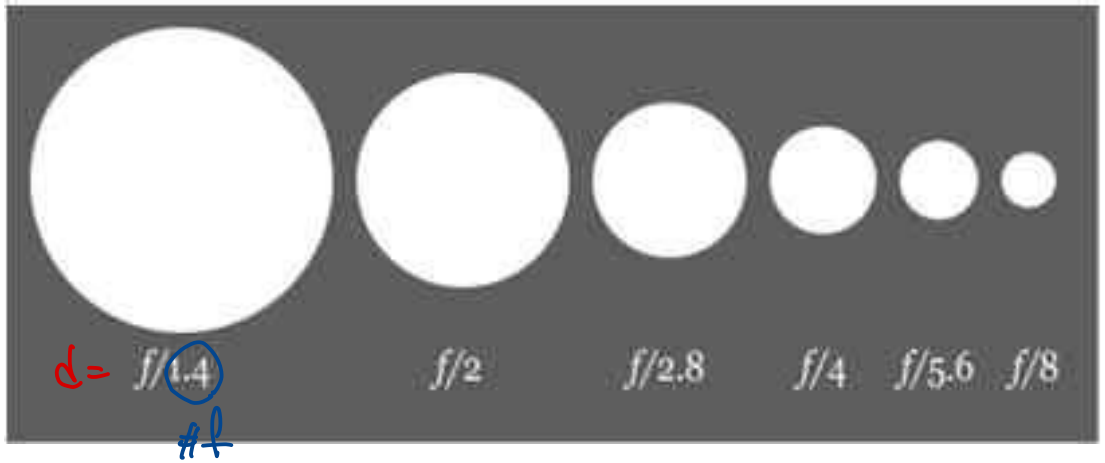
F-Number

F-number (조리개 값) (f# = f/d)

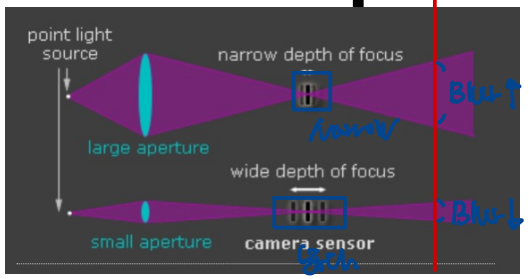
- Determines the image **brightness**
- Affects **depth of field**



- Small f# → (big aperture)
- Large f# → (small aperture)

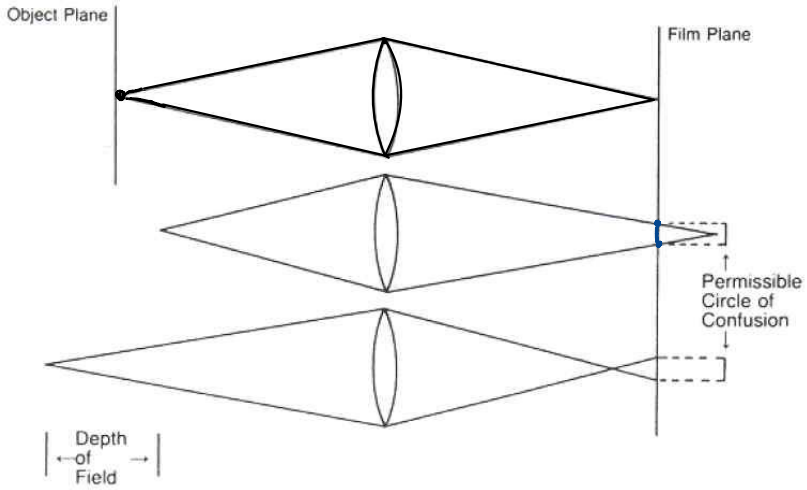


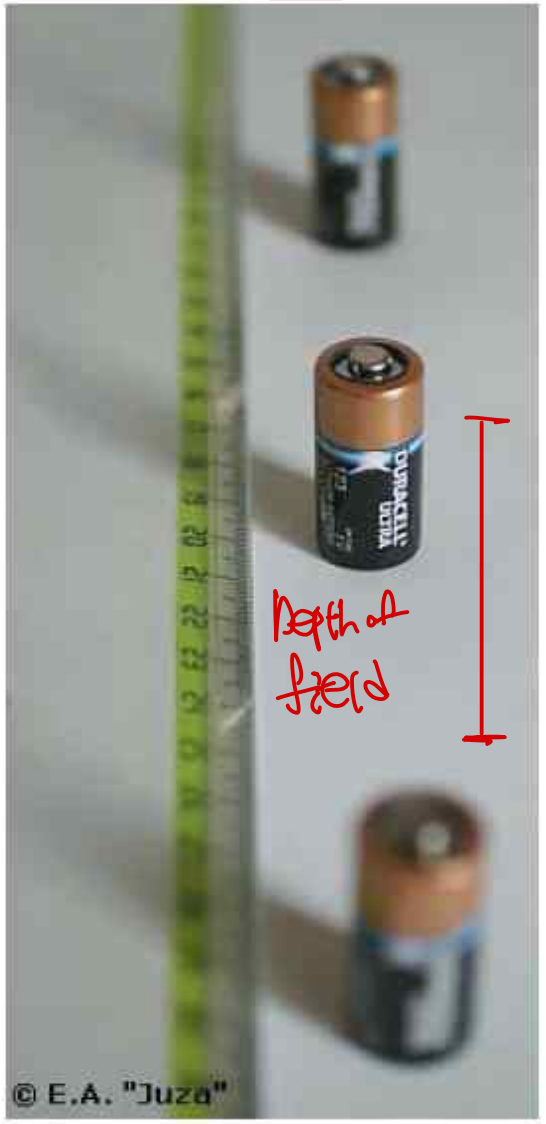
Depth of Field and Aperture



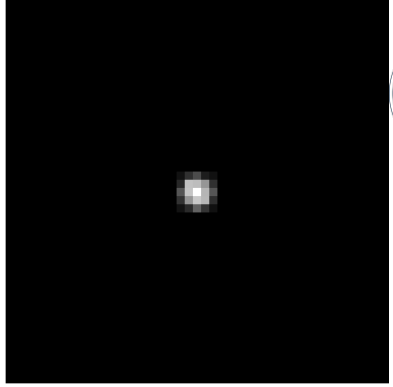
Depth of Field (피사계 심도)

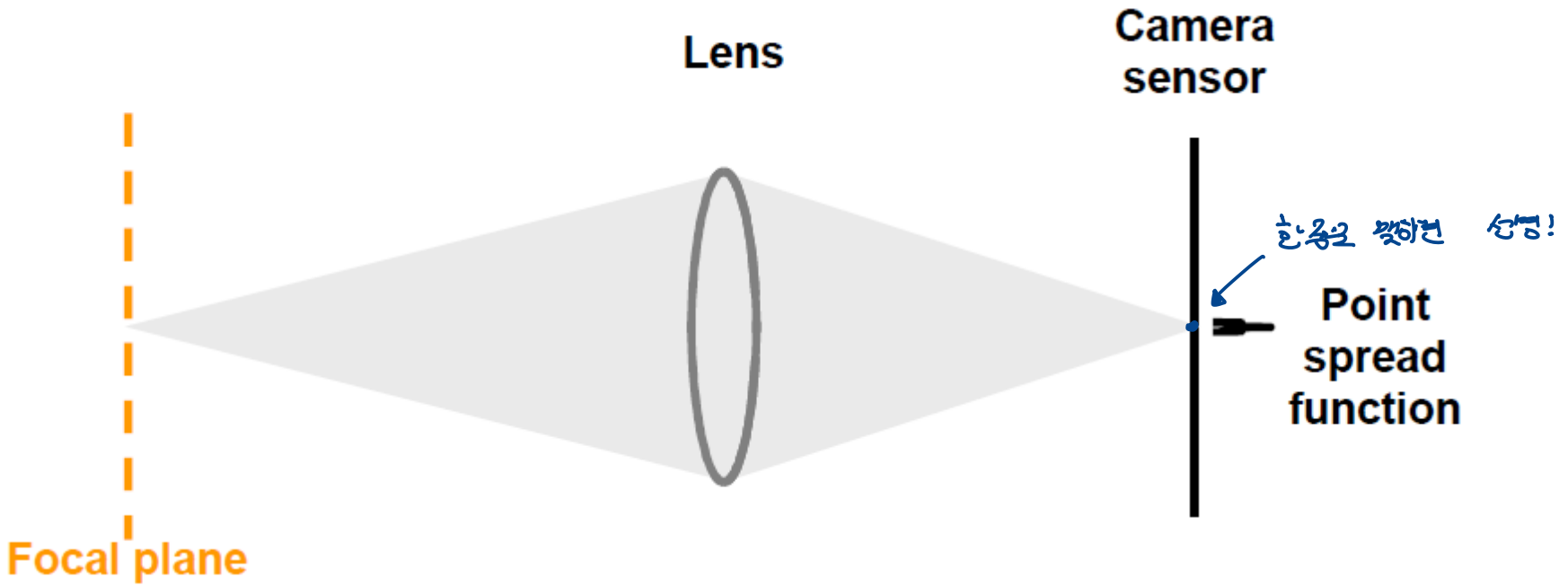
- The range of distances where objects are imaged sharp
- Depends on f-number (**aperture**)
- Small f# → (big aperture) → narrow depth of field
- Large f# → (small aperture) → broad depth of field



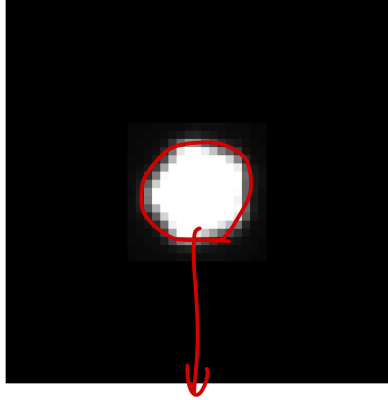


Lens and Defocus

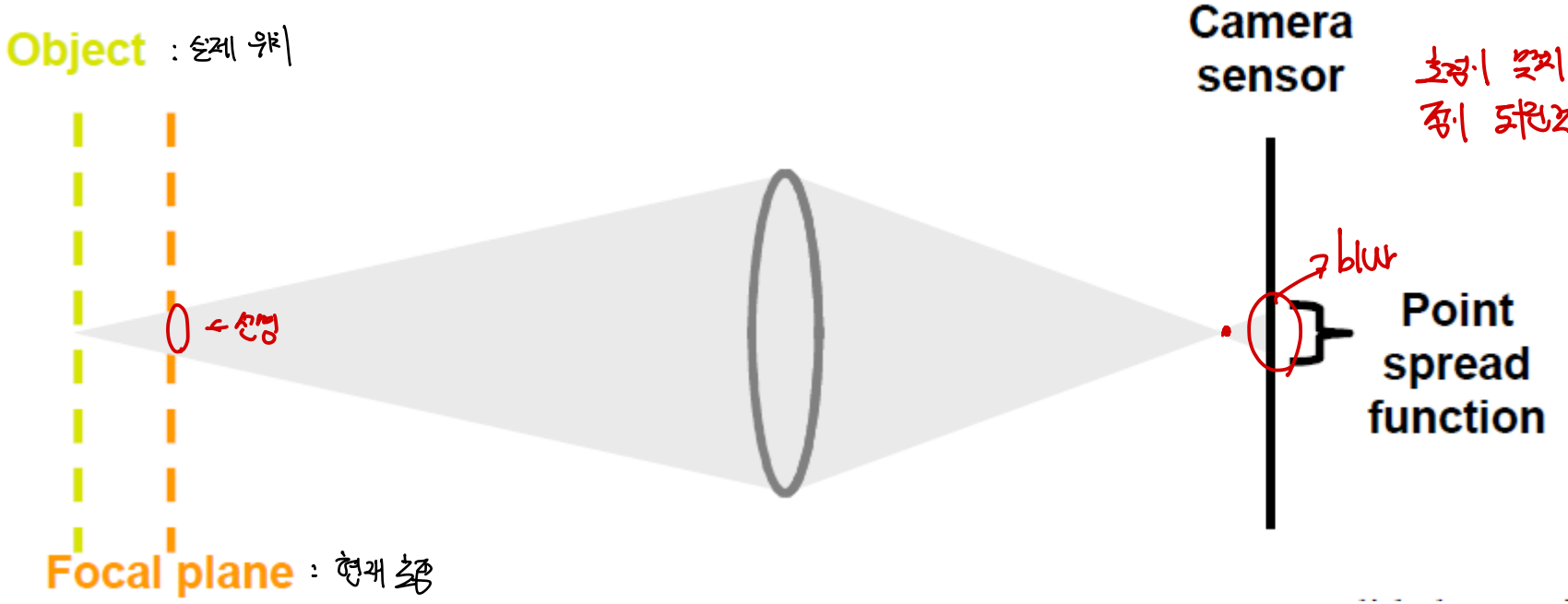




Lens and Defocus



point spread



Brightness of Images

Camera Sensor에 들어오는 빛의 양

Brightness of image is determined by the amount of light falling on the **Sensor**.

조리개

1. Image brightness is determined by **Aperture**:

- Small f# → (big aperture) → I large (bright)
- Large f# → (small aperture) → I small (dark)

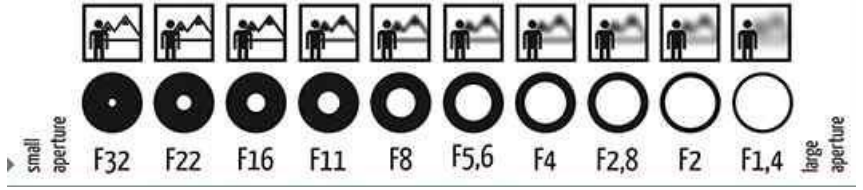
$$I \propto \frac{1}{(f\#)^2} \propto A$$

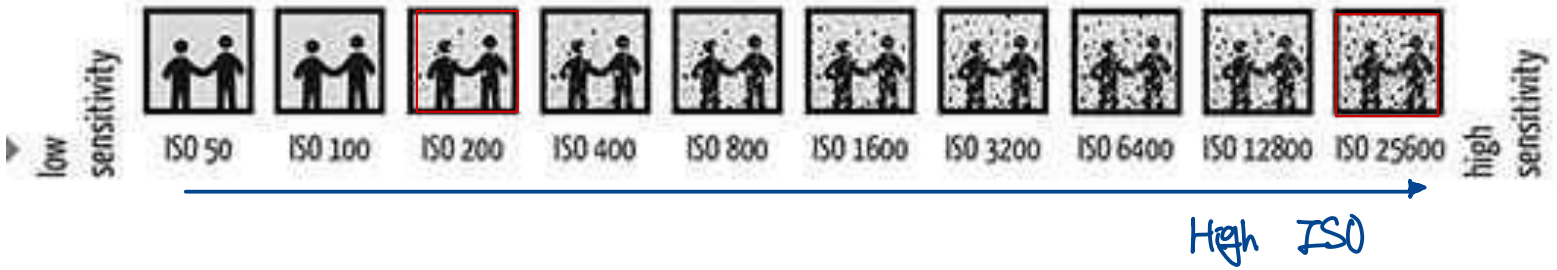
Brightness also affected by Sensor parameters:

2. **ISO Gain** and 3. **Shutter speed**

Image **Quality** Control

- Aperture Size ➔ Depth of field (**Focus blur**)
- Shutter Speed ➔ **Motion Blur**
- ISO (gain) ➔ **Noise**





ISO↑ : 밝이 적은 상황에서 밝게 찍힘
그러나 noise가 증가

"How to **enhance** the image from such **noises** and **blur**?
Coming Soon."

ITE4052 Computer Vision

Light

Dong-Jin Kim

Spring 2025