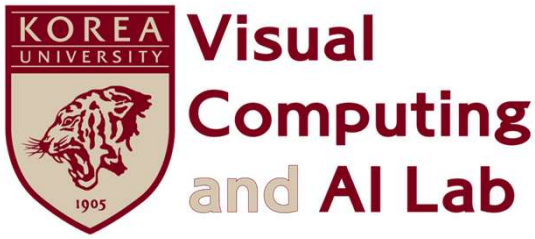
INTRODUCTION TO COMPUTER VISION

Lecture 2 – Image Formation

Gyeongsik Moon

Visual Computing and AI Lab

Korea University



Slides Credit: Prof. Dr. Andreas Geiger

1. Primitives and Transformations
2. Geometric Image Formation
3. Photometric Image Formation
4. Image Sensing Pipeline

1. Primitives and Transformations

Primitives and Transformations

- Geometric primitives are the **basic building blocks** used to describe 3D shapes
- In this unit, we introduce **points, lines and planes**
- Furthermore, the **most basic transformations** are discussed

Camera 에서 멀어질 수록 2D 좌표는 가까워짐.

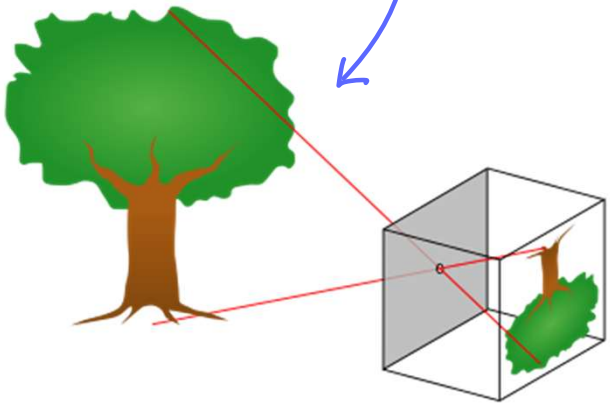
2D vs. 3D

- In the 3D space, parallel lines do not intersect with each other
- In the 2D space, we have a vanishing point!
  - All parallel lines intersect at the vanishing point
  - Why this happens? -> Perspective projection / pinhole camera model
  - How can we represent such vanishing point? -> homogeneous coordinates

realistic 하지만

실제로 보는것에 불일치.

(평행) (소실점)



Homogeneous coordinates

소실점을 표현하는 (설명할 수 있는) 방법.

- What is this?
  - Instead of (x,y), we use (wx,wy,w).
  - For any w, (wx, wy, w) represent the same point in the homogeneous coordinate (w is a positive real number. w=0 is a singularity.)
- Why we need this?
  - Can represent the vanishing point
  - Use only matrix multiplication for transformations (rotation and translations, …)

$X' = RX + T \Rightarrow Px$

Rotation matrix Transition vector

2D Points

2D points can be written in **inhomogeneous coordinates** as

$$\mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{R}^2$$

or in **homogeneous coordinates** as

$$\tilde{\mathbf{x}} = \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{pmatrix} \in \mathbb{P}^2$$

$\begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \\ 2 \end{pmatrix}$

W의 스케일된 다르면 homogeneous system에서 같은 point.

where $\mathbb{P}^2 = \mathbb{R}^3 \setminus \{(0,0,0)\}$ is called **projective space.**

excluding origin

Remark: Homogeneous vectors that differ only by scale are considered equivalent and define an equivalence class. ⇒ Homogeneous vectors are defined only up to scale. 0을 제외한 크기에 구애받지 않음. 오직 방향만 중요.

2D Points

An **inhomogeneous vector** $\mathbf{x}$ is converted to a **homogeneous vector** $\tilde{\mathbf{x}}$ as follows

$$\tilde{\mathbf{x}} = \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{pmatrix} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix} = \bar{\mathbf{x}}$$

w=1

with **augmented vector** $\bar{\mathbf{x}}$.To convert in the opposite direction we divide by $\tilde{w}$:

$$\bar{\mathbf{x}} = \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \frac{1}{\tilde{w}}\tilde{\mathbf{x}} = \frac{1}{\tilde{w}}\begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{pmatrix} = \begin{pmatrix} \tilde{x}/\tilde{w} \\ \tilde{y}/\tilde{w} \\ 1 \end{pmatrix}$$

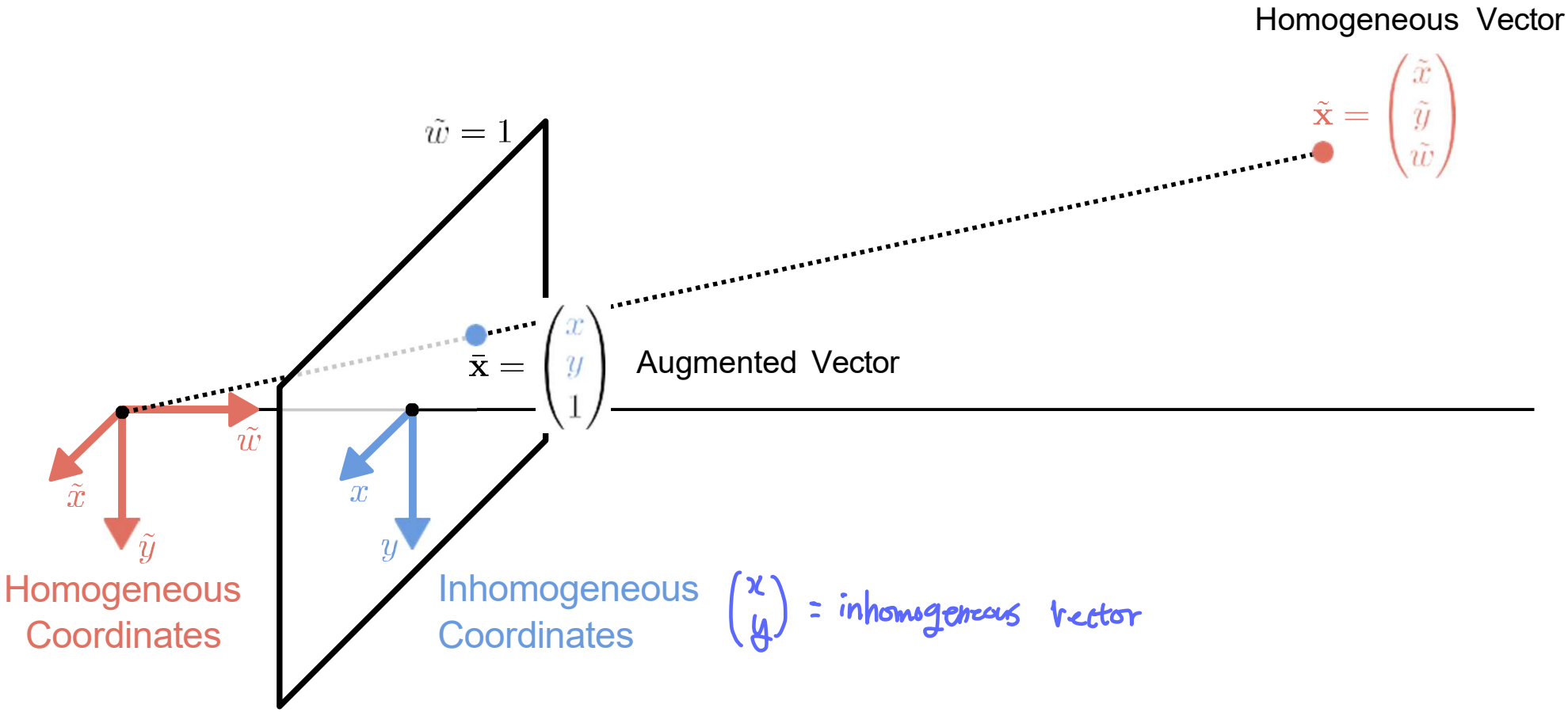
Homogeneous points whose last element is $\tilde{w} = 0$ are called **ideal points** or **points at infinity.** These points can't be represented with inhomogeneous coordinates!

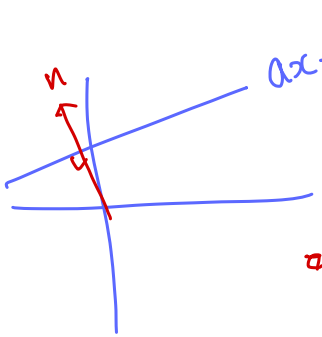
무한대에 위치한 점. : 위치가 아닌 방향만을 정의함

= 직선들이 평행할 때 교차하는 점.

$\tilde{w}=0$ 이면 직교좌표계로 전환이 안됨.

2D Points


Homogeneous Vector
$\tilde{\mathbf{x}} = \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{pmatrix}$
$\tilde{w} = 1$
$\bar{\mathbf{x}} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$
Augmented Vector
$\tilde{w}$
$\tilde{x}$
$\tilde{y}$
x
y
Homogeneous Coordinates
Inhomogeneous Coordinates
$\begin{pmatrix} x \\ y \end{pmatrix}$ = inhomogeneous vector



$ax+by+c=0$

$a\frac{dx}{dt}+b\frac{dy}{dt}=0$

$(a,b)\cdot\left(\frac{dx}{dt},\frac{dy}{dt}\right)=0.$

$\vec{d}=\left(\frac{dx}{dt},\frac{dy}{dt}\right)$: 직선의 방향벡터

따라서 (a,b)는 항상 직선의 수직. (=법선)

$ax+by+cz+d=0$

$\vec{n}=(a,b,c)$

(x_0,y_0)과 직선 $ax+by+c=0$ 사이의 거리

$\frac{|ax_0+by_0+c|}{\sqrt{a^2+b^2}}$ → (원점일때) $\frac{|c|}{\sqrt{a^2+b^2}}$

(a,b)를 (n_x,n_y)로 정규화

$\vec{n}\cdot((x,y)-(x_0,y_0))=0$

$n_x x+n_y y-(n_x x_0+n_y y_0)=0.$

d

$d=-\frac{c}{\sqrt{a^2+b^2}}$

2D Lines

2D lines can also be expressed using homogeneous coordinates $\tilde{\mathbf{l}}=(a,b,c)^\top$:

$$\{\bar{\mathbf{x}}\mid\tilde{\mathbf{l}}^\top\bar{\mathbf{x}}=0\}\quad\Leftrightarrow\quad\{x,y\mid ax+by+c=0\}$$

$\tilde{\mathbf{l}}^\top\bar{\mathbf{x}}=0$ 又가 직선위에 존재.

→ 직선 위의 점들의 집합.

We can **normalize** $\tilde{\mathbf{l}}$ so that $\tilde{\mathbf{l}}=(n_x,n_y,-d)^\top=(\mathbf{n},-d)^\top$ with $\|\mathbf{n}\|_2=1$. In this case, $\mathbf{n}$ is the normal vector perpendicular to the line and d is its distance to the origin.

An exception is the **line at infinity** $\tilde{\mathbf{l}}_\infty=(0,0,1)^\top$ which passes through all ideal points.

$0x+0y+1\cdot w=0.$

$w=0.$

Cross Product

$$\begin{pmatrix} a_2 b_3 - a_3 b_2 \\ -a_1 b_3 + a_3 b_1 \\ a_1 b_2 - a_2 b_1 \end{pmatrix}$$

Cross product expressed as the product of a skew-symmetric matrix and a vector:

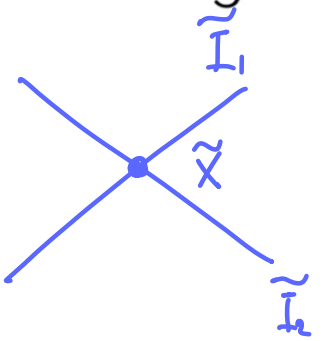
$$\mathbf{a} \times \mathbf{b} = [\mathbf{a}]_\times \mathbf{b} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} a_2\, b_3 - a_3\, b_2 \\ a_3\, b_1 - a_1\, b_3 \\ a_1\, b_2 - a_2\, b_1 \end{pmatrix}$$

Remark: In this course, we use squared brackets to distinguish matrices from vectors.

2D Line Arithmetic

In homogeneous coordinates, the **intersection** of two lines is given by:

$$\tilde{\mathbf{x}} = \tilde{\mathbf{l}}_1 \times \tilde{\mathbf{l}}_2$$



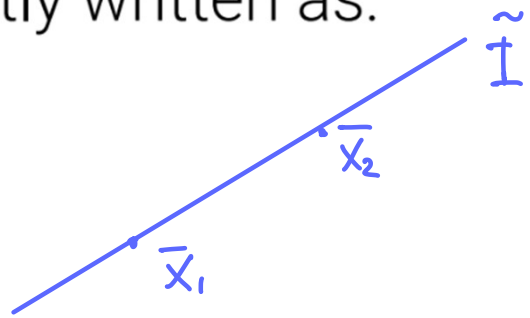
두 직선의 외적 = 두 직선을 동시에 만족하는 점의 동차좌표.

Similarly, the **line joining two points** can be compactly written as:

$$\tilde{\mathbf{l}} = \bar{\mathbf{x}}_1 \times \bar{\mathbf{x}}_2$$

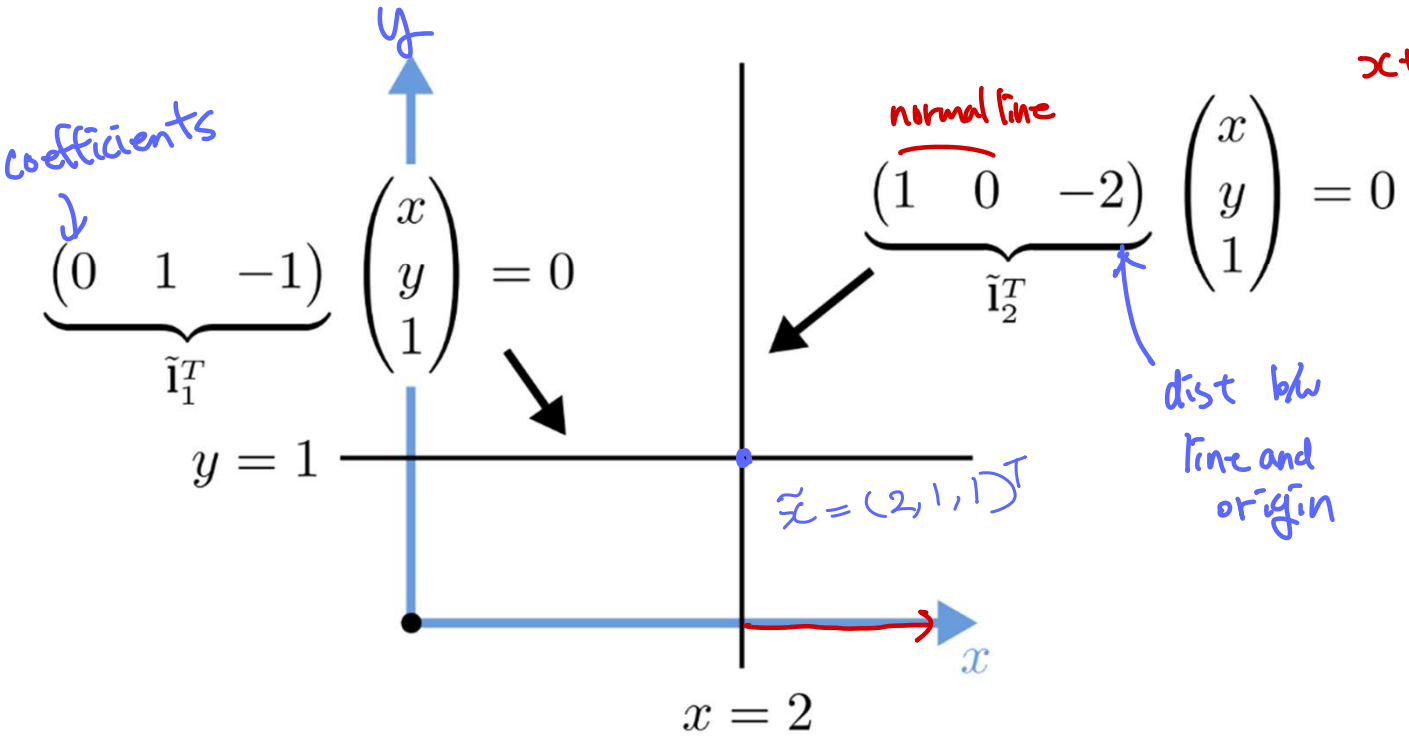
$\tilde{l}$은 두 점의 동차벡터와 모두 수직. 즉 외적.

$\tilde{l}\bar{x}_1 = 0$ and $\tilde{l}\bar{x}_2 = 0$



The symbol $\times$ denotes the cross product. Proof as exercise.

2D Line Arithmetic



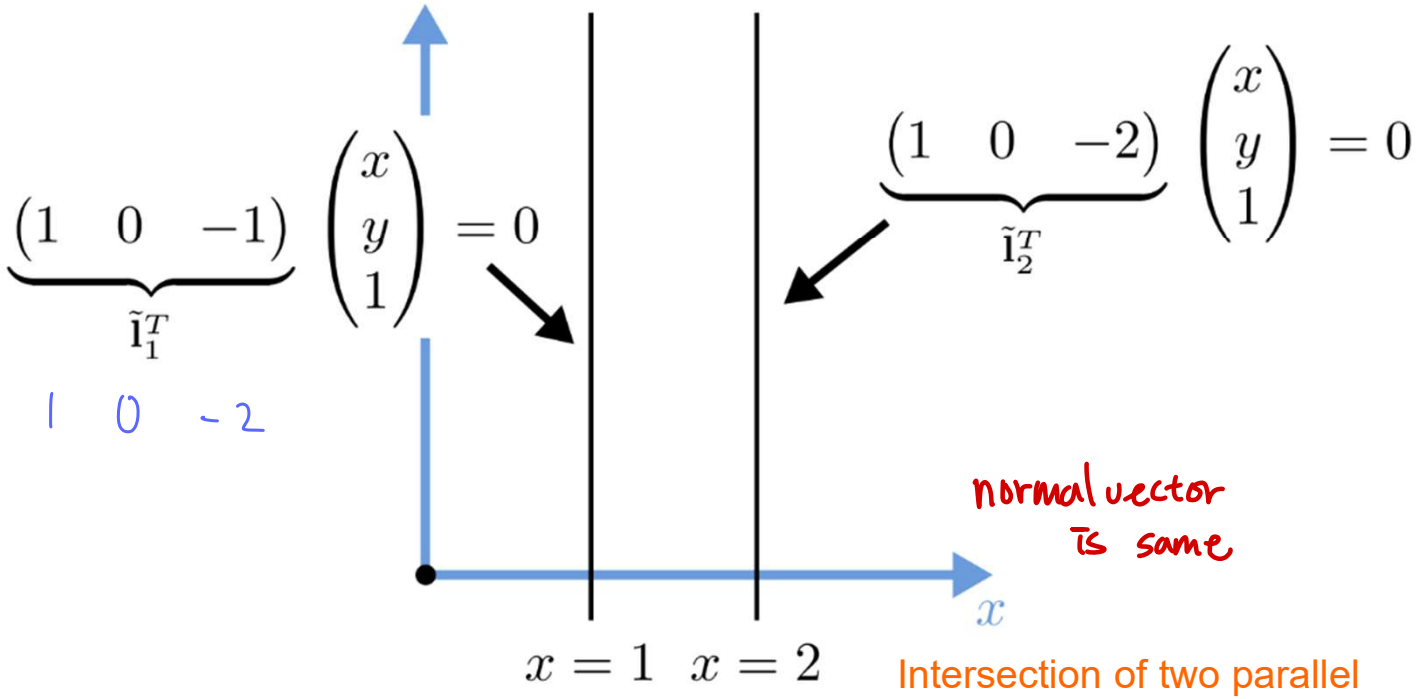
$$[\mathbf{x}]_\times = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}$$

$$\begin{pmatrix} -2 \\ -1 \\ -1 \end{pmatrix}$$

$$\tilde{\mathbf{l}}_1 \times \tilde{\mathbf{l}}_2 = [\tilde{\mathbf{l}}_1]_\times \tilde{\mathbf{l}}_2 = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} \begin{pmatrix} 1 \\ 0 \\ -2 \end{pmatrix} = \begin{pmatrix} -2 \\ -1 \\ -1 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix}$$

2D Line Arithmetic

$\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$ → (0,1) 방향의 infinity point.



$$[\mathbf{x}]_\times = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}$$

Intersection of two parallel lines -> vanishing point (x,y,0)

1) line at infinite and 2) points at infinity intersect with each other

$$\tilde{\mathbf{l}}_1 \times \tilde{\mathbf{l}}_2 = [\tilde{\mathbf{l}}_1]_\times \tilde{\mathbf{l}}_2 = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} 1 \\ 0 \\ -2 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

direction

$$\underbrace{\begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^\top}_{\tilde{\mathbf{l}}_\infty} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = 0$$

infinite dist from origin.

intersecting point. homo 좌표 시스템이라서 가능.

line @ inf.

point @ inf.

3D Points

3D points can be written in **inhomogeneous coordinates** as

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \in \mathbb{R}^3$$

or in **homogeneous coordinates** as

$$\tilde{\mathbf{x}} = \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \\ \tilde{w} \end{pmatrix} \in \mathbb{P}^3$$

with **projective space** $\mathbb{P}^3 = \mathbb{R}^4 \setminus \{(0,0,0,0)\}$.

3D Planes

3D planes can also be represented as homogeneous coordinates $\tilde{\mathbf{m}} = (a, b, c, d)^\top$:

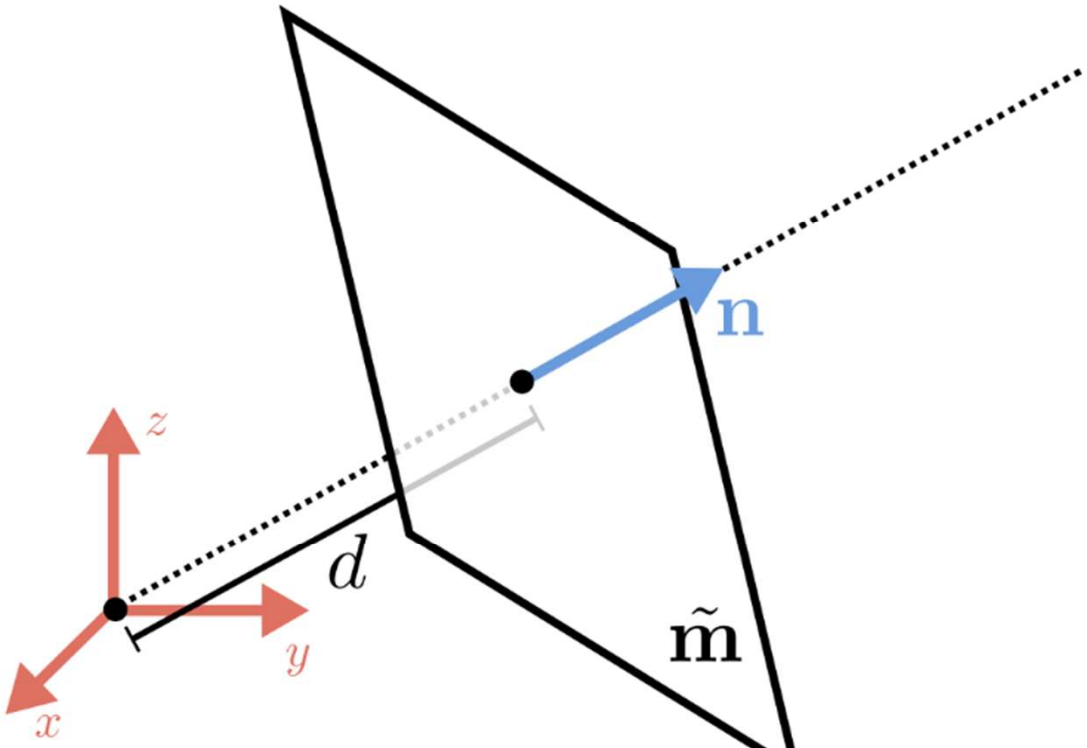
$$\{\bar{\mathbf{x}} \mid \tilde{\mathbf{m}}^\top \bar{\mathbf{x}} = 0\} \quad \Leftrightarrow \quad \{x, y, z \mid ax + by + cz + d = 0\}$$

Again, we can **normalize** $\tilde{\mathbf{m}}$ so that $\tilde{\mathbf{m}} = (n_x, n_y, n_z, -d)^\top = (\mathbf{n}, -d)^\top$ with $\|\mathbf{n}\|_2 = 1$. In this case, $\mathbf{n}$ is the normal perpendicular to the plane and d is its distance to the origin.

An exception is the **plane at infinity** $\tilde{\mathbf{m}} = (0, 0, 0, 1)^\top$ which passes through all ideal points (= points at infinity) for which $\tilde{w} = 0$.

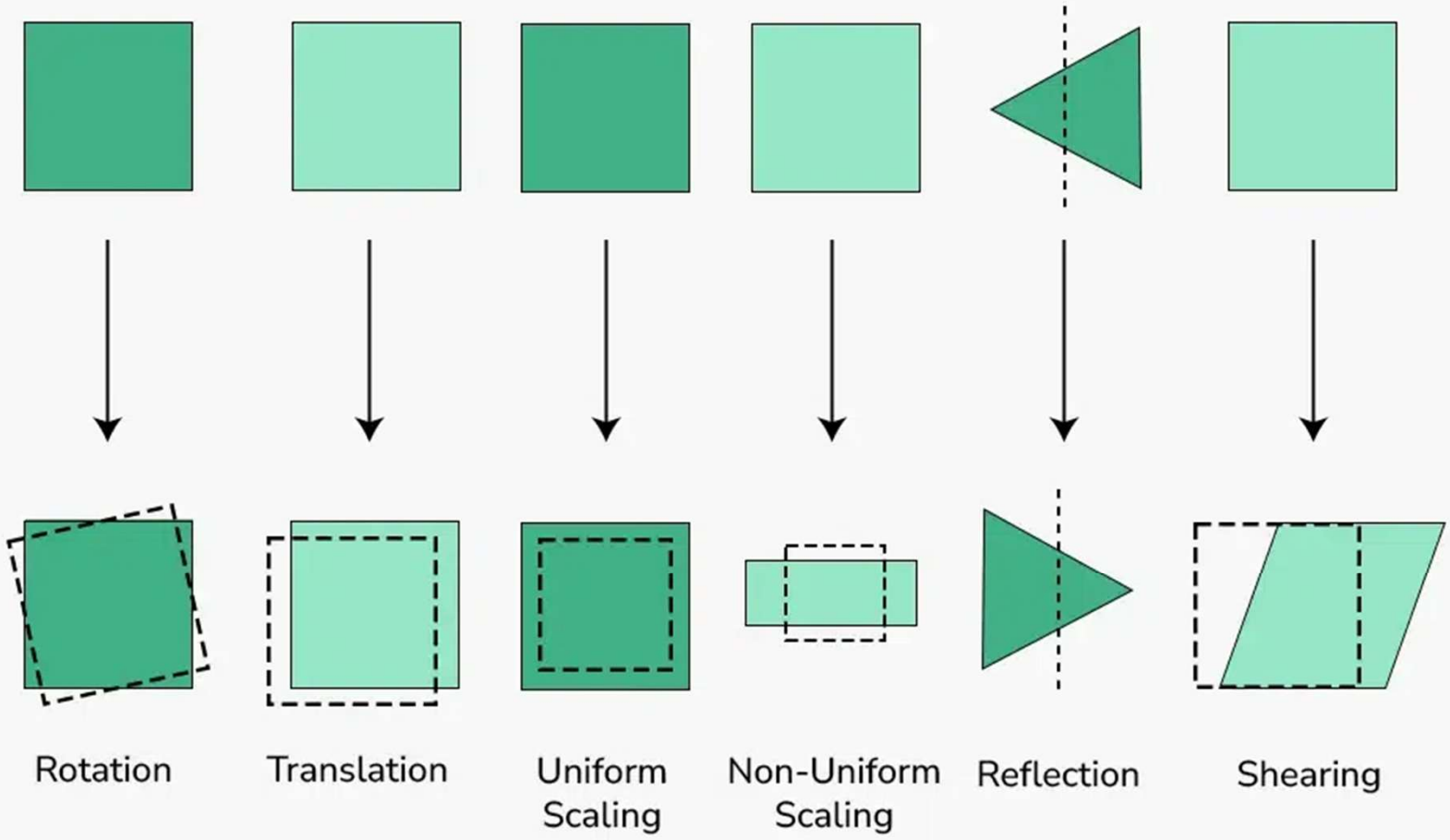
$(x, y, z, 0)$ $w = 0$ 이면 on this plane 인 point

3D Planes

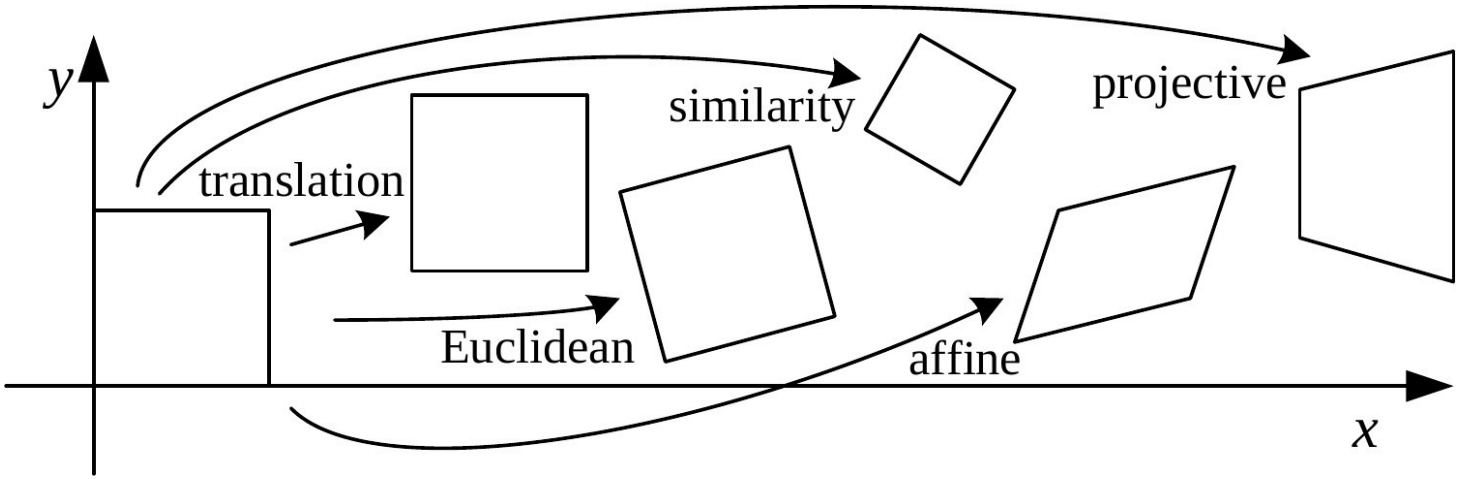


$$\underbrace{\begin{pmatrix} n_x & n_y & n_z & -d \end{pmatrix}}_{\tilde{\mathbf{m}}^T} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = 0$$

Geometric Transformation in Image Processing



2D Transformations

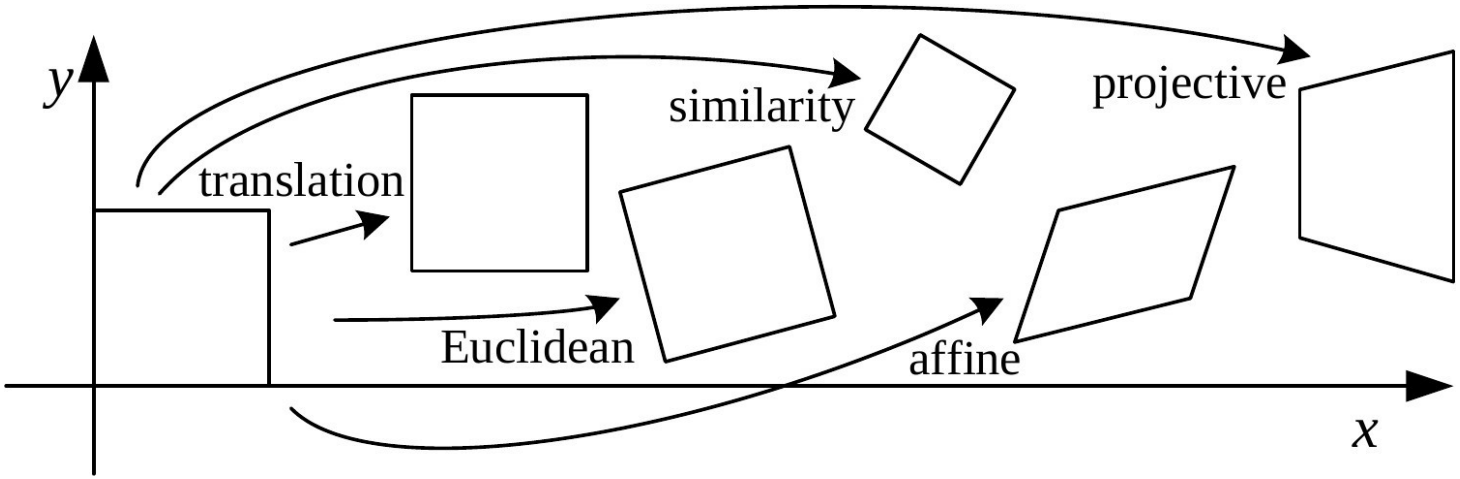


Translation: (2D Translation of the Input, 2 DoF) Degree of freedom

$$\mathbf{x}' = \mathbf{x} + \mathbf{t} \quad \Leftrightarrow \quad \bar{\mathbf{x}}' = \begin{bmatrix} \mathbf{I} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \bar{\mathbf{x}}$$

- Using homogeneous representations allows to chain/invert transformations
- Augmented vectors $\bar{\mathbf{x}}$ can always be replaced by general homogeneous ones $\tilde{\mathbf{x}}$

2D Transformations



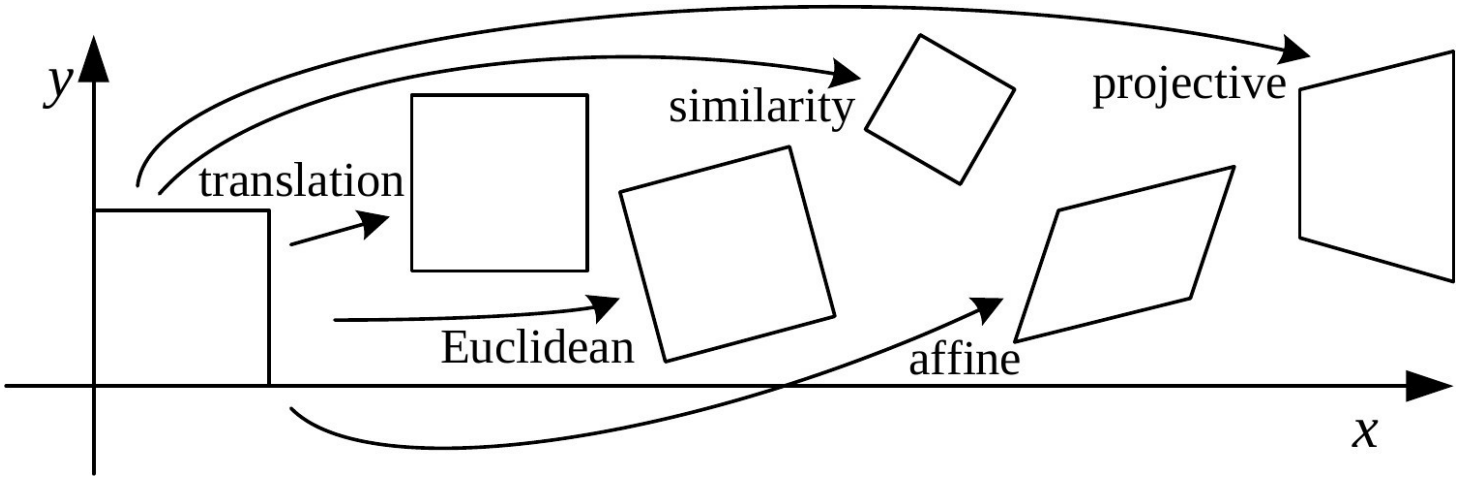
Euclidean: (2D Translation + 2D Rotation, 3 DoF)

rigid translation (3D에서 더 사용되는 용어)

$$\mathbf{x}' = \mathbf{R}\mathbf{x} + \mathbf{t} \quad \Leftrightarrow \quad \bar{\mathbf{x}}' = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \bar{\mathbf{x}}$$

- $\mathbf{R} \in SO(2)$ is an orthonormal rotation matrix with $\mathbf{R}\mathbf{R}^\top = \mathbf{I}$ and $\det(\mathbf{R}) = 1$
- Euclidean transformations preserve Euclidean distances

2D Transformations

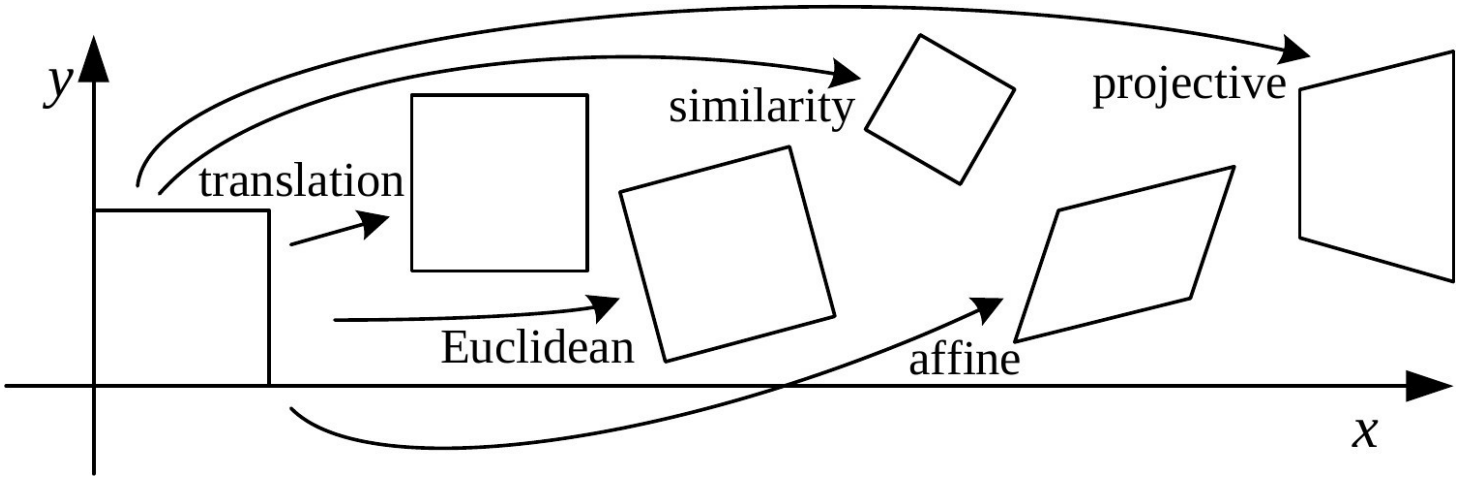


Similarity: (2D Translation + Scaled 2D Rotation, 4 DoF)

$$\mathbf{x}' = s\mathbf{R}\mathbf{x} + \mathbf{t} \quad \Leftrightarrow \quad \bar{\mathbf{x}}' = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \bar{\mathbf{x}}$$

- $\mathbf{R} \in SO(2)$ is a rotation matrix and s is an arbitrary scale factor
- The similarity transform preserves angles between lines

2D Transformations



Affine: (2D Linear Transformation, 6 DoF)

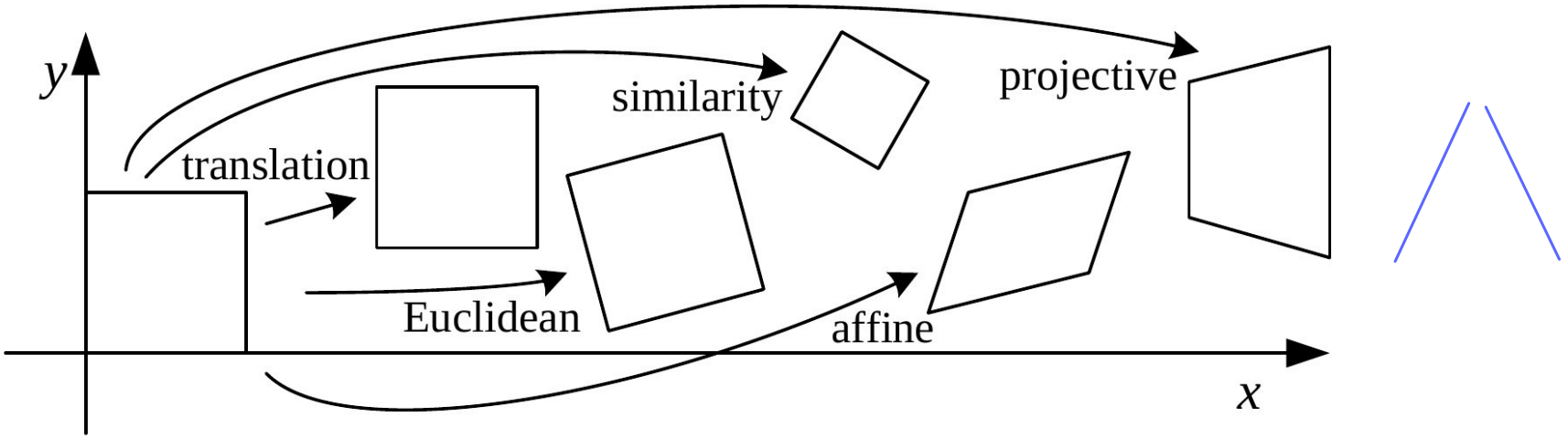
$$\mathbf{x}' = \mathbf{A}\mathbf{x} + \mathbf{t} \quad \Leftrightarrow \quad \bar{\mathbf{x}}' = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \bar{\mathbf{x}}$$

2×2

$$\begin{matrix} A_{11} & A_{12} & t_1 \\ A_{21} & A_{22} & t_2 \\ 0 & 0 & 1 \end{matrix}$$

- $\mathbf{A} \in \mathbb{R}^{2\times 2}$ is an arbitrary 2×2 matrix
- Parallel lines remain parallel under affine transformations

2D Transformations



Projective: (Homography, 8 DoF)

2D 3D 사이의 discrepancy 설명가능해짐.

Homography

$$\tilde{\mathbf{x}}' = \tilde{\mathbf{H}}\,\tilde{\mathbf{x}} \qquad \left(\bar{\mathbf{x}} = \frac{1}{\tilde{w}}\tilde{\mathbf{x}}\right)$$

$\tilde{H} = \begin{pmatrix} A & t \\ a\ b & 1 \end{pmatrix}$ ← 아래는 1임.

자유도 4

- $\tilde{\mathbf{H}} \in \mathbb{R}^{3\times 3}$ is an arbitrary homogeneous 3×3 matrix (specified up to scale)
- Projective transformations preserve straight lines

curve가 그려진 않음.

2D Transformations on Co-Vectors

Considering any perspective 2D transformation

$$\tilde{\mathbf{x}}' = \tilde{\mathbf{H}}\tilde{\mathbf{x}}$$

the transformed 2D line equation is given by:

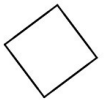
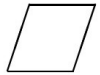
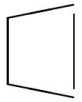
$$\tilde{\mathbf{l}}'^{\top}\tilde{\mathbf{x}}' = \tilde{\mathbf{l}}'^{\top}\tilde{\mathbf{H}}\tilde{\mathbf{x}} = (\tilde{\mathbf{H}}^{\top}\tilde{\mathbf{l}}')^{\top}\tilde{\mathbf{x}} = \tilde{\mathbf{l}}^{\top}\tilde{\mathbf{x}} = 0$$

Therefore, we have:

$$\tilde{\mathbf{l}}' = \tilde{\mathbf{H}}^{-\top}\tilde{\mathbf{l}}$$

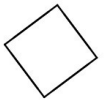
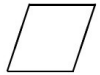
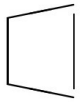
Thus, the action of a projective **transformation on a co-vector** such as a 2D line or 3D normal can be represented by the transposed inverse of the matrix.

Overview of 2D Transformations

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	

- Transformations form **nested set of groups** (closed under composition, inverse)
- Interpret as restricted 3×3 matrices operating on 2D homogeneous coordinates
- Transformations preserve properties below (similarity: parallelism, straight lines)

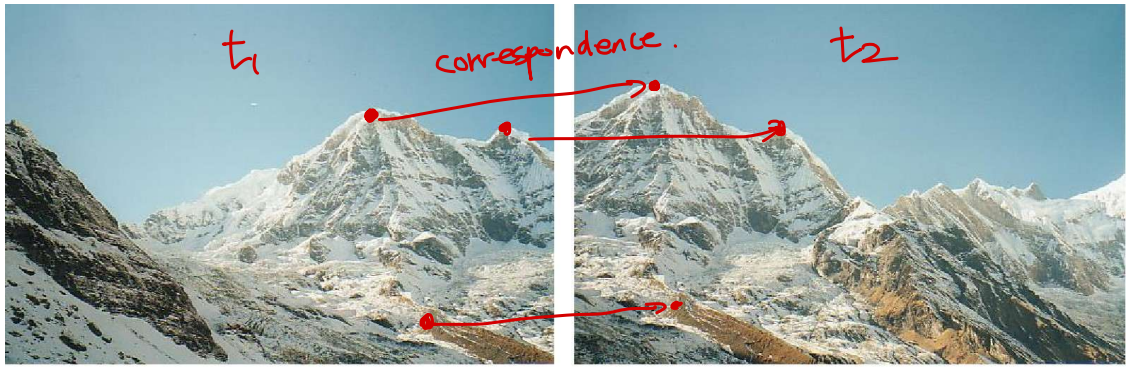
Overview of 3D Transformations

Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix}\mathbf{I} & \mathbf{t}\end{bmatrix}_{3\times4}$	3	orientation	
rigid (Euclidean)	$\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}_{3\times4}$	6	lengths	
similarity	$\begin{bmatrix}s\mathbf{R} & \mathbf{t}\end{bmatrix}_{3\times4}$	7	angles	
affine	$\begin{bmatrix}\mathbf{A}\end{bmatrix}_{3\times4}$	12	parallelism	
projective	$\begin{bmatrix}\tilde{\mathbf{H}}\end{bmatrix}_{4\times4}$	15	straight lines	

- 3D transformations are defined analogously to 2D transformations
- 3 ×4 matrices are extended with a fourth $[\mathbf{0}^\top\ 1]$ row for homogeneous transforms
- Transformations preserve properties below (similarity: parallelism, straight lines)

Application: Panorama Stitching

- Get correspondence between pixels from two images
- Construct projective matrix (Direct Linear Transformation algorithm)
- Bringing pixels with the correspondence together to get panorama image



using SIFT

$I_1 = HI_2$

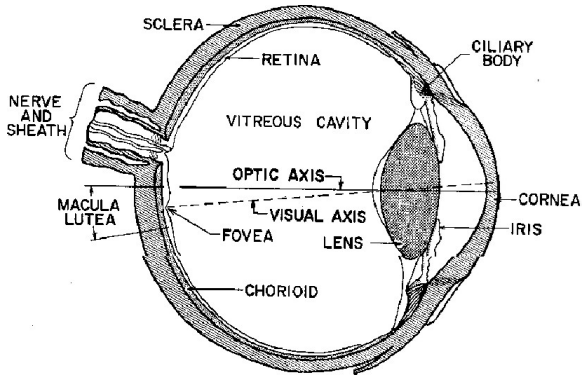
Brown and Lowe: Recognising Panoramas. ICCV, 2003

2. Geometric Image Formation

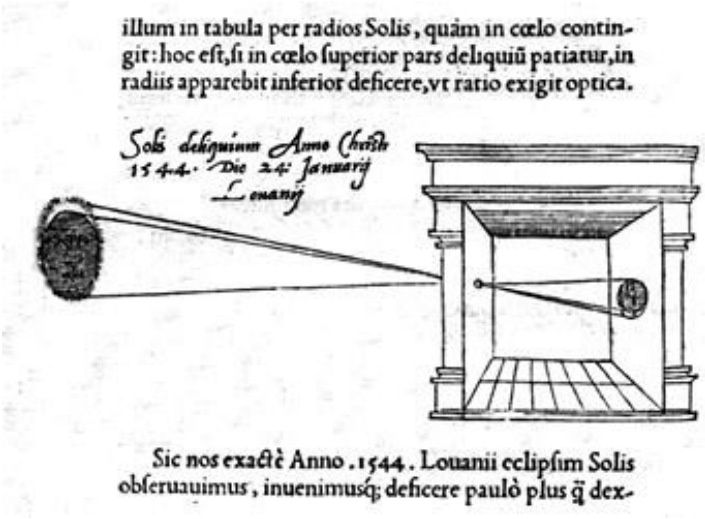
Capturing 3D World into 2D Image Plane

- We're living in 3D world, but what we see is **always** 2D projections of the 3D world
  - There is no 'depth' dimension; only height and width dimensions
- 3D games or 3D movies have their 3D representations, but they render the 3D representations to 2D screen space <- this is what we **see**

Origins of the Pinhole Camera



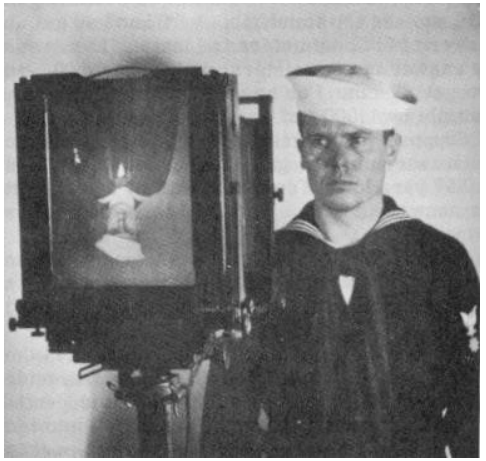
Animal Eye:
A long time ago


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-

Pinhole Perspective Projection:
Brunelleschi, 15th Century



Photographic Camera:
Nicéphore Niépce, 1816

Origins of the Pinhole Camera

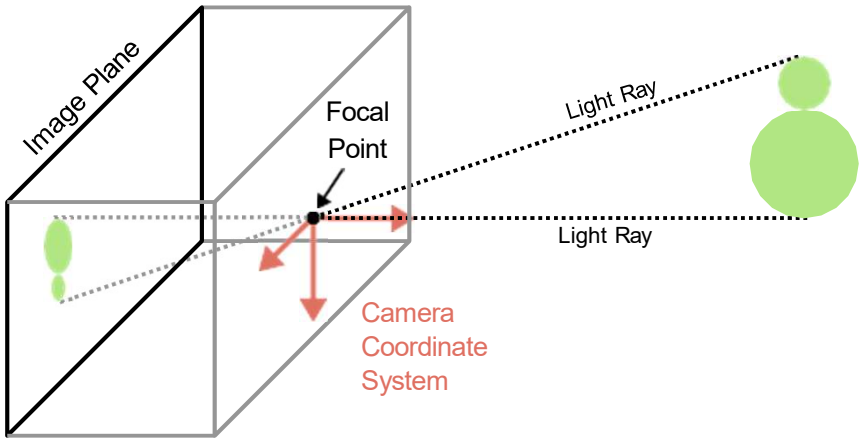
Camera Obscura
: 4th Century BC

Origins of the Pinhole Camera

https://www.abelardomorell.net/camera-obscura

Origins of the Pinhole Camera

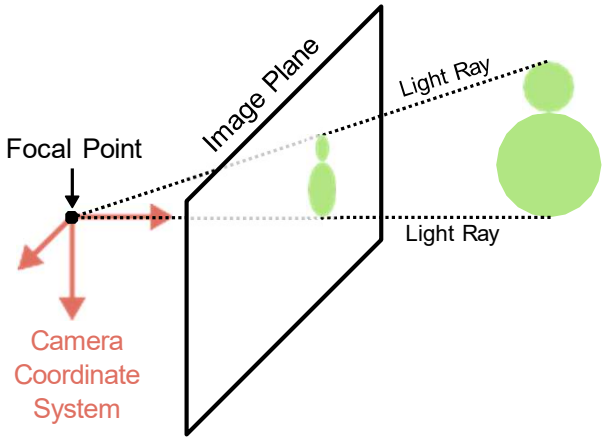
Physical Camera Model



Focal point 뒤에 있어서 upside down

Mathematical Camera Model

CV

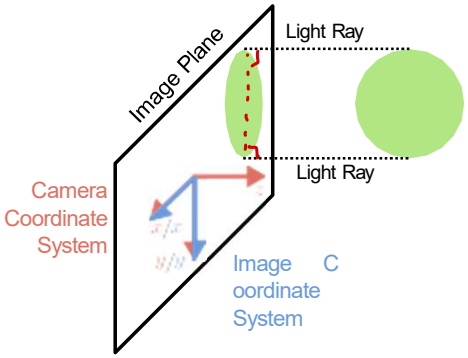


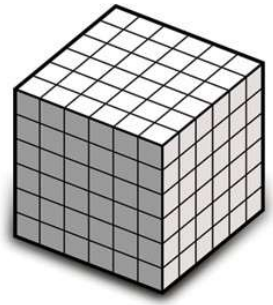
- In a physical pinhole camera, the image is projected up-side down onto the image plane which is located **behind** the focal point
- When modeling perspective projection, we assume the image plane **in front**
- Both models are **equivalent,** with appropriate change of image coordinates

Projection Models

Orthographic Projection

approximation





Field of View is small.

small FoV is good for capturing very distant object.

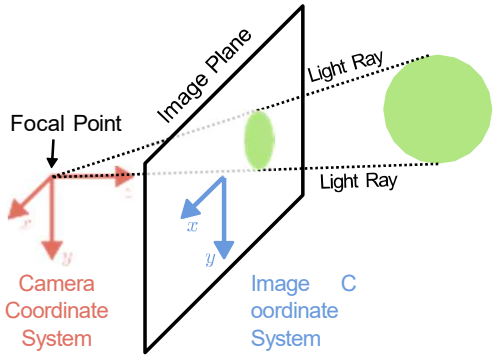
Opto Engineering Telecentric Lens

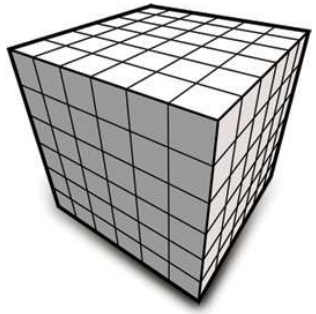
Canon 800mm Telephoto Lens

Perspective Projection

correct representation

not preserve parallel lines.





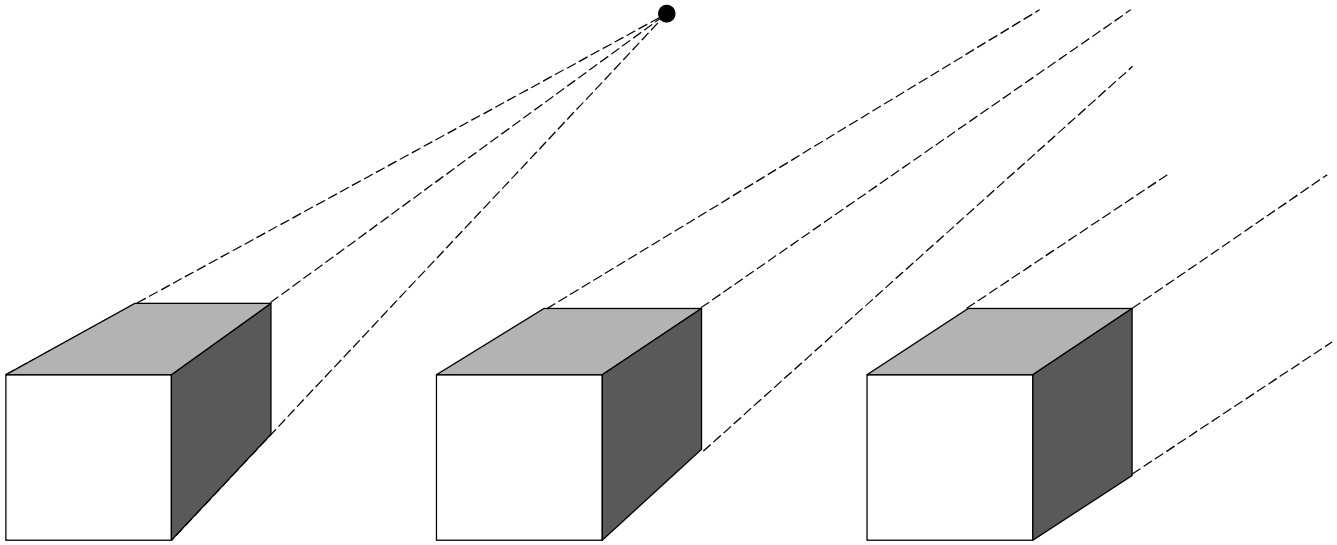
Nikon AF-S Nikkor 50mm

Sony DSC-RX100 V

Samsung Galaxy S20

- These two are the most important projections

Projection Models

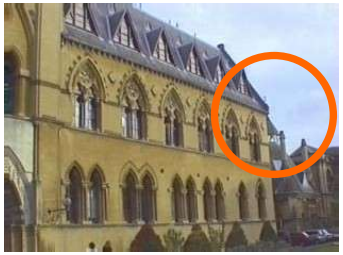


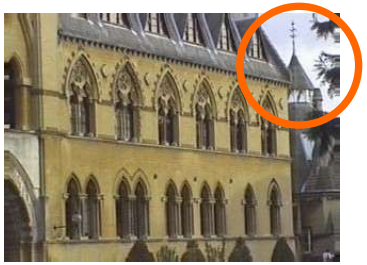
Perspective

Weak Perspective /Orthographic

Increasing Focal Length / Distance from Camera

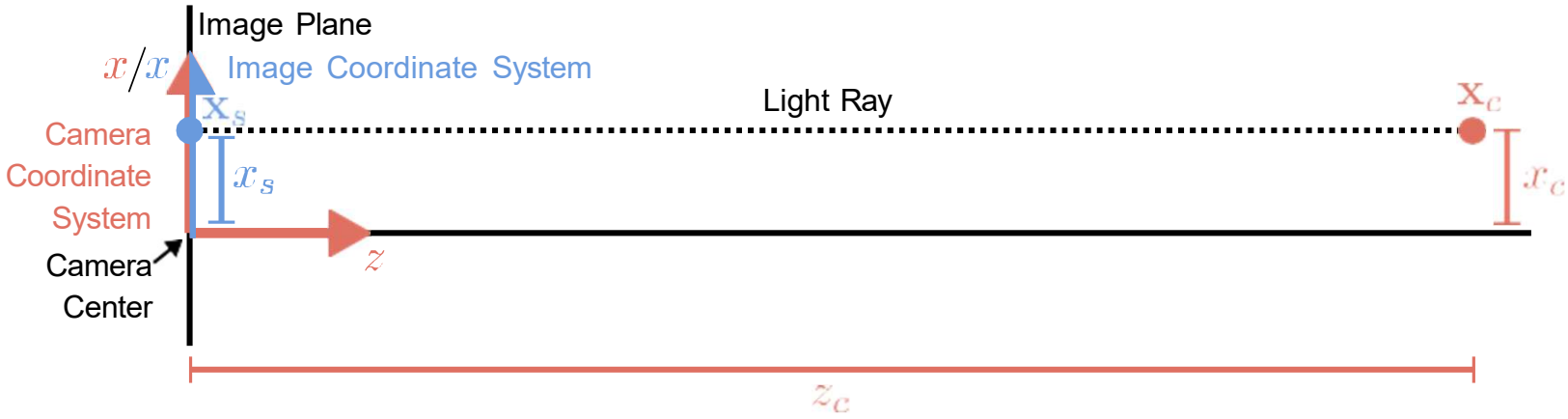
= Zooming





Different from simply zooming in the 2D image space

Orthographic Projection



$$x_s = x_c$$

Orthographic projection of a 3D point $\mathbf{x}_c \in \mathrm{R}^3$ to pixel coordinates $\mathbf{x}_s \in \mathrm{R}^2$:

- The x and y axes of the camera and image coordinate systems are shared
- Light rays are parallel to the z-coordinate of the camera coordinate system
- During projection, the z-coordinate is dropped, x and y remain the same
- Remark: the y coordinate is not shown here for clarity, but behaves similarly

Orthographic Projection

An **orthographic projection** simply **drops the z component** of the 3D point in camera coordinates $\mathbf{x}_c$ to obtain the corresponding 2D point on the image plane (= screen) $\mathbf{x}_s$.

$$\mathbf{x}_s = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{x}_c \quad \Leftrightarrow \quad \bar{\mathbf{x}}_s = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \bar{\mathbf{x}}_c$$

Orthography is exact for telecentric lenses and an approximation for telephoto lenses.

After projection the distance of the 3D point from the image can't be recovered.

$$\begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} \rightarrow \begin{pmatrix} x_s \\ y_s \end{pmatrix}$$

Scaled Orthographic Projection

0.1m object → 0.1 px → not make sense

In practice, world coordinates (which may measure dimensions in meters) must be scaled to fit onto an image sensor (measuring in pixels) $\Rightarrow$ **scaled orthography:**

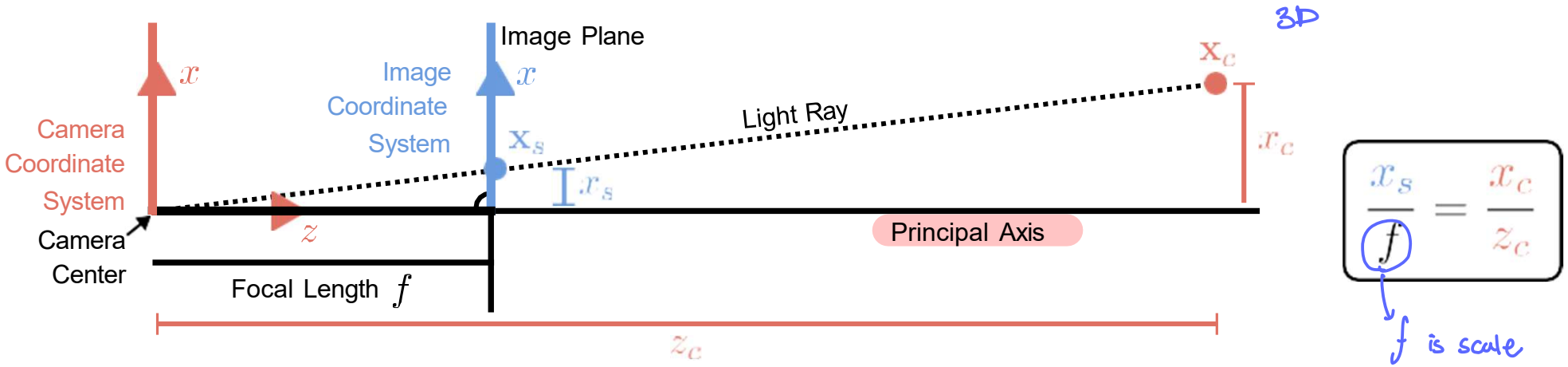
$$\mathbf{x}_s = \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \end{bmatrix} \mathbf{x}_c \quad \Leftrightarrow \quad \bar{\mathbf{x}}_s = \begin{bmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \bar{\mathbf{x}}_c$$

Remark: The unit for s is px/m or px/mm to convert metric 3D points into pixels.

predefined.

Under orthography, structure and motion can be estimated simultaneously using factorization methods (e.g., via singular value decomposition).

Perspective Projection



Perspective projection of a 3D point $\mathbf{x}_c \in \mathbb{R}^3$ to pixel coordinates $\mathbf{x}_s \in \mathbb{R}^2$:

- The light ray passes through the camera center, the pixel $\mathbf{x}_s$ and the point $\mathbf{x}_c$
- Convention: the principal axis (orthogonal to image plane) aligns with the z-axis
- Remark: the y coordinate is not shown here for clarity, but behaves similarly

x, y축 원점 위치는 conventionally 중심.

Perspective Projection

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

$$\begin{pmatrix} x_s \\ y_s \end{pmatrix} = \begin{pmatrix} f x_c / z_c \\ f y_c / z_c \end{pmatrix} \quad \Leftrightarrow \quad \tilde{\mathbf{x}}_s = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \bar{\mathbf{x}}_c$$

cancelled

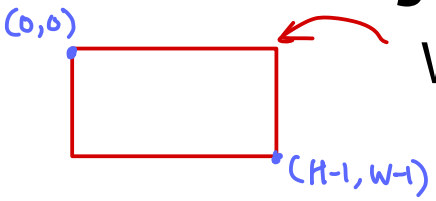
Note that this projection is **linear** when using **homogeneous coordinates.** After the projection it is not possible to recover the distance of the 3D point from the image.

Remark: The unit for f is px (=pixels) to convert metric 3D points into pixels.

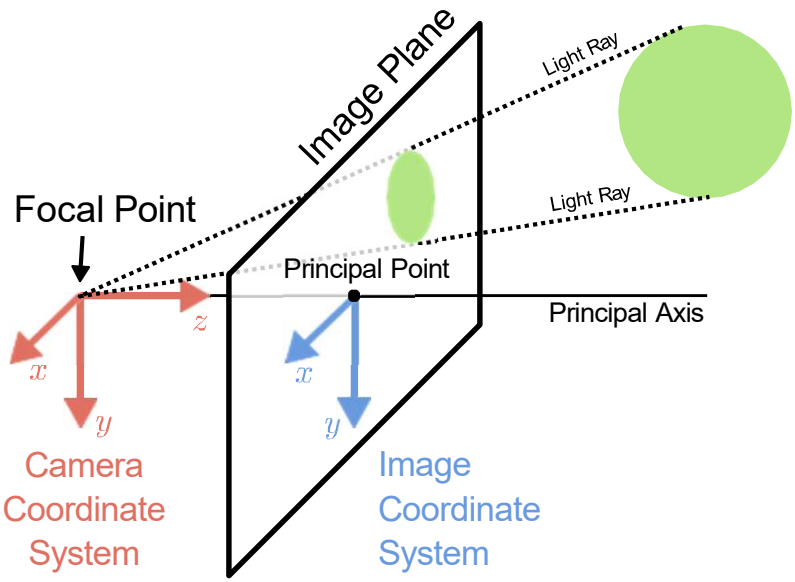
Perspective Projection

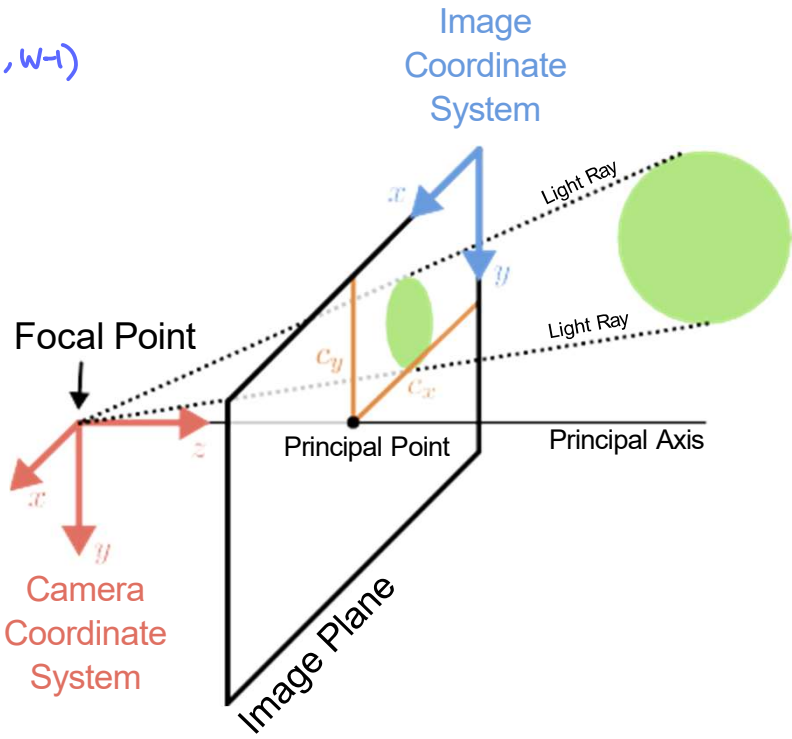
not important.

Without Principal Point Offset



With Principal Point Offset





- To ensure positive pixel coordinates, a **principal point offset c** is usually added
- This moves the image coordinate system to the corner of the image plane

f = zooming.

Perspective Projection

The **complete perspective projection model** is given by:

$$\begin{pmatrix} x_s \\ y_s \end{pmatrix} = \begin{pmatrix} f_x\, x_c/z_c + s\, y_c/z_c + c_x \\ f_y\, y_c/z_c + c_y \end{pmatrix} \quad \Leftrightarrow \quad \tilde{\mathbf{x}}_s = \begin{bmatrix} f_x & s & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \bar{\mathbf{x}}_c$$

scale

offset.

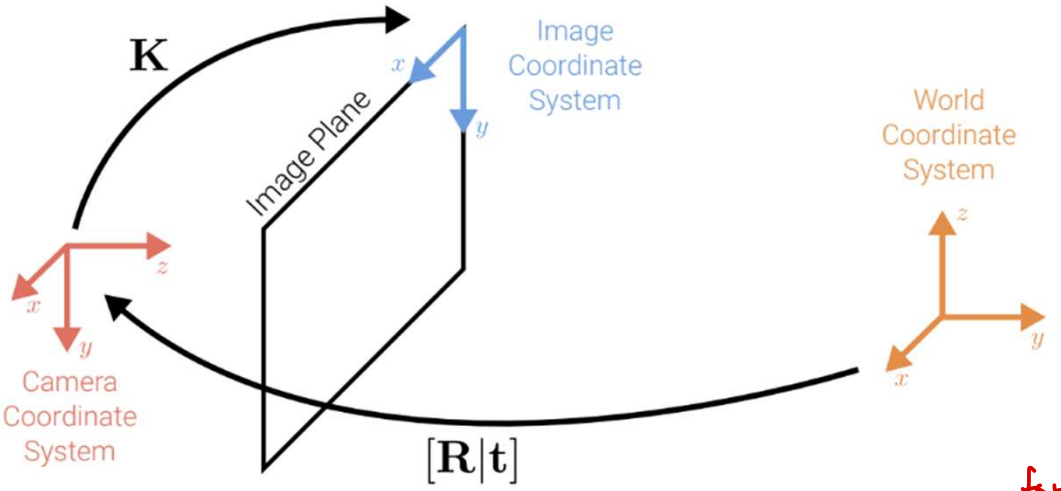
usual case

$$\begin{pmatrix} f & 0 & W/2 \\ 0 & f & H/2 \\ 0 & 0 & 1 \end{pmatrix}$$

- The left 3×3 submatrix of the projection matrix is called **calibration matrix $\mathbf{K}$**
- The parameters of $\mathbf{K}$ are called camera intrinsics (as opposed to extrinsic pose)
- Here, f_x and f_y are independent, allowing for different pixel aspect ratios
- The skew s arises due to the sensor not mounted perpendicular to the optical axis
- In practice, we often set $f_x = f_y$ and $s = 0$, but model $\mathbf{c} = (c_x, c_y)^\top$

center of image

Chaining Transformations



for relationship b/w different camera

Let $\mathbf{K}$ be the calibration matrix (intrinsics) and $[\mathbf{R}|\mathbf{t}]$ the camera pose (extrinsics).

We **chain both transformations** to project a point in world coordinates to the image:

$$\tilde{\mathbf{x}}_s = \begin{bmatrix} \mathbf{K} & \mathbf{0} \end{bmatrix} \bar{\mathbf{x}}_c = \begin{bmatrix} \mathbf{K} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \bar{\mathbf{x}}_w = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \bar{\mathbf{x}}_w = \mathbf{P}\, \bar{\mathbf{x}}_w$$

Remark: The 3×4 projection matrix $\mathbf{P}$ can be pre-computed.

Full Rank Representation

It is sometimes preferable to use a **full rank** 4×4 projection matrix:

$$\tilde{\mathbf{x}}_s = \begin{bmatrix} \mathbf{K} & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \bar{\mathbf{x}}_w = \tilde{\mathbf{P}}\, \bar{\mathbf{x}}_w$$

Now, the homogeneous vector $\tilde{\mathbf{x}}_s$ is a 4D vector and must be normalized wrt. its 3rd entry to obtain inhomogeneous image pixels:

$$\bar{\mathbf{x}}_s = \tilde{\mathbf{x}}_s / z_s = (x_s/z_s, y_s/z_s, 1, 1/z_s)^\top$$

Note that the 4th component of the inhomogeneous 4D vector is the **inverse depth.** If the inverse depth is known, a 3D point can be retrieved from its pixel coordinates via $\tilde{\mathbf{x}}_w = \tilde{\mathbf{P}}^{-1}\, \bar{\mathbf{x}}_s$ and subsequent normalization of $\tilde{\mathbf{x}}_w$ wrt. its 4th entry.

Summary

$X_S = f X_c / Z_c + C_x$

$X_c = (X_S - C_x) Z_c / f$

↳ 근데 Z가 소실되서 복구못해.

근데 AI로 근사치추정 가능.

- World coordinates (Blue)
  - A reference 3D coordinate system
- Camera coordinates (Orange)
  - Defined in the 3D space for each camera
- Image coordinates (Black)
  - Defined in the 2D space for each camera

ex

in

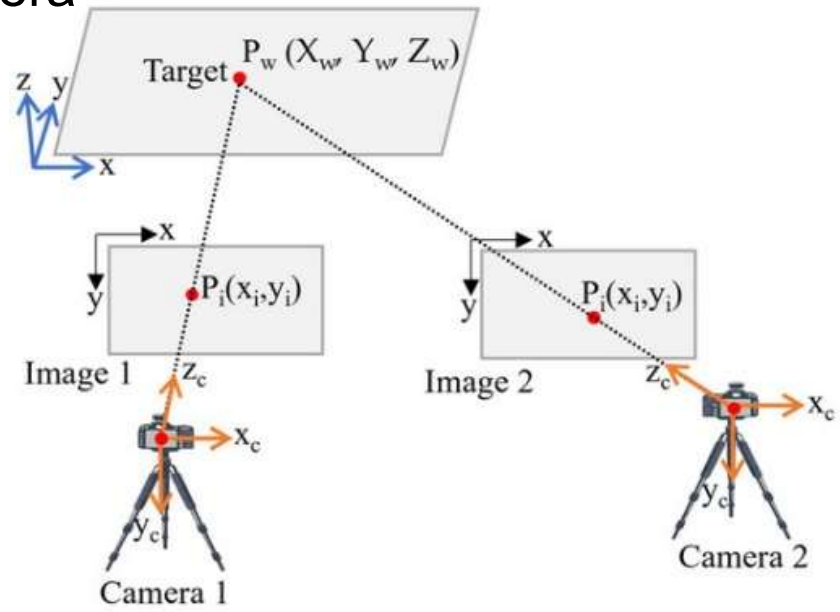
camera-relative

$W \rightarrow C$

$X_c = R X_w + t$

$C \rightarrow W$

$X_w = R^{-1}(X_c - t)$



Summary

- P = [R|t]K (projection matrix)
  - Multiplication of extrinsic and intrinsic matrices
  - World coordinates -> camera coordinates -> image coordinates
  - In general, 3x4 matrix
  - Multiplied with 3D homogeneous coordinates $(X_w, Y_w, Z_w, 1)^T$ and outputs 2D homogeneous coordinates $(u,v,w)^T$
  - $(u/w, v/w, 1)^T$ becomes final 2D homogeneous coordinates in the image space
    - This is where ‘distortion (e.g., vanishing point)’ is happening
  - In computer graphics (OpenGL), we pad P with (0,0,0,1) row vector at the last row -> 4x4 matrix

Summary

- [R|t] (extrinsic matrix)
  - World coordinates -> camera coordinates
  - R:rotation w.r.t. world coordinate system
  - t: translation w.r.t. world coordinate system
  - In general, 3x4 matrix
  - Multiplied with 3D homogeneous coordinates $(X_w, Y_w, Z_w, 1)^T$ and outputs 3D Euclidean coordinates $(X_c, Y_c, Z_c)^T$
  - In computer graphics (OpenGL), we pad [R|t] with (0,0,0,1) row vector at the last row -> 4x4 matrix

Summary

- K (intrinsic matrix)
  - camera coordinates -> image space
  - In general, 3x3 matrix
  - Multiplied with 3D Euclidean coordinates $(X_c,Y_c,Z_c)^T$ and outputs 2D homogeneous coordinates $(u,v,w)^T$
    - This is where 'distortion (e.g., vanishing point)' is happening
  - Or multiplied with 2D homogeneous coordinates $(X_c/Z_c,Y_c/Z_c,1)^T$ and outputs 2D homogeneous coordinates $(u/w,v/w,1)^T$
    - This is where 'distortion (e.g., vanishing point)' is happening
  - In computer graphics (OpenGL), we pad K with (0,0,0,1) row/column vectors at the last row/column -> 4x4 matrix

3D to 2D

perspective / Orthographic projection happens.

Advanced: Lens Distortion

even straight lines are not preserved.

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

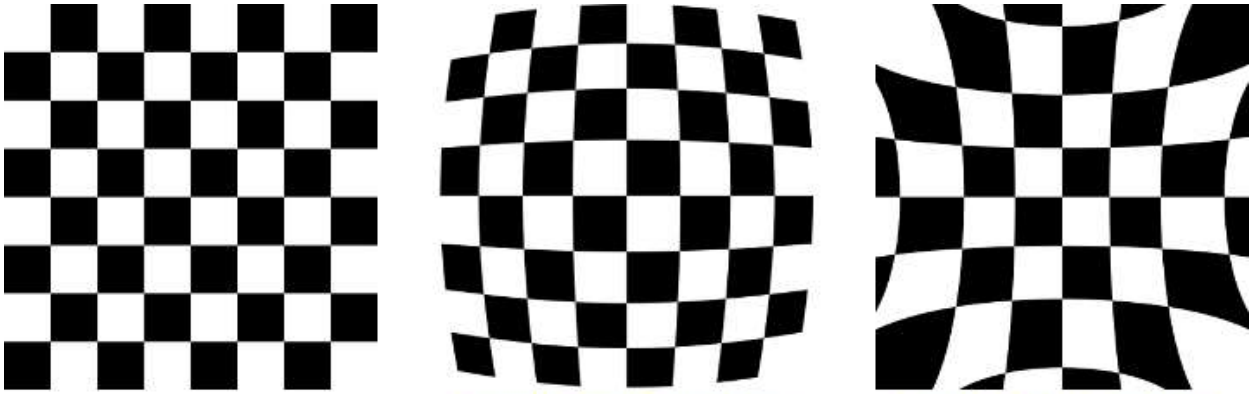
$$\mathbf{x}' = \underbrace{(1 + \kappa_1 r^2 + \kappa_2 r^4)}_{\text{Radial Distortion}} \begin{pmatrix} x \\ y \end{pmatrix} + \underbrace{\begin{pmatrix} 2\kappa_3 x y + \kappa_4(r^2 + 2x^2) \\ 2\kappa_4 x y + \kappa_3(r^2 + 2y^2) \end{pmatrix}}_{\text{Tangential Distortion}}$$

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

수식 안나옴.

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

Advanced: Lens Distortion

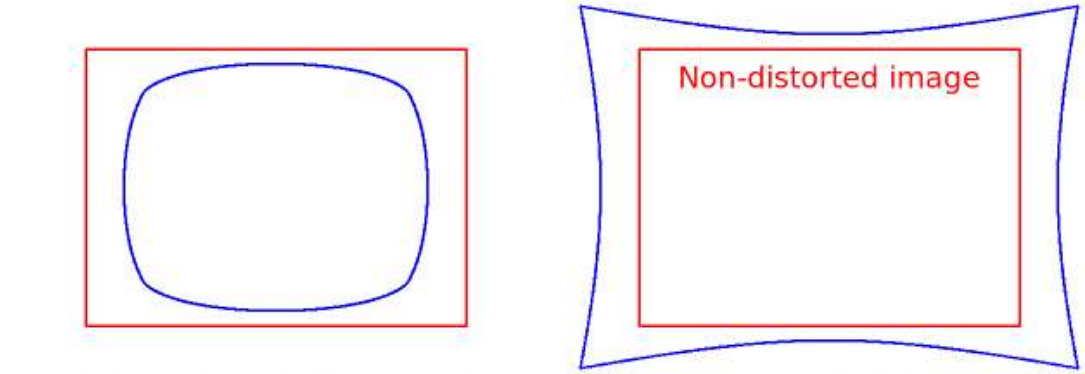


No distortion

Negative radial distortion
(Barrel distortion)

Positive radial distortion
(Pincushion distortion)

fish-eye camera

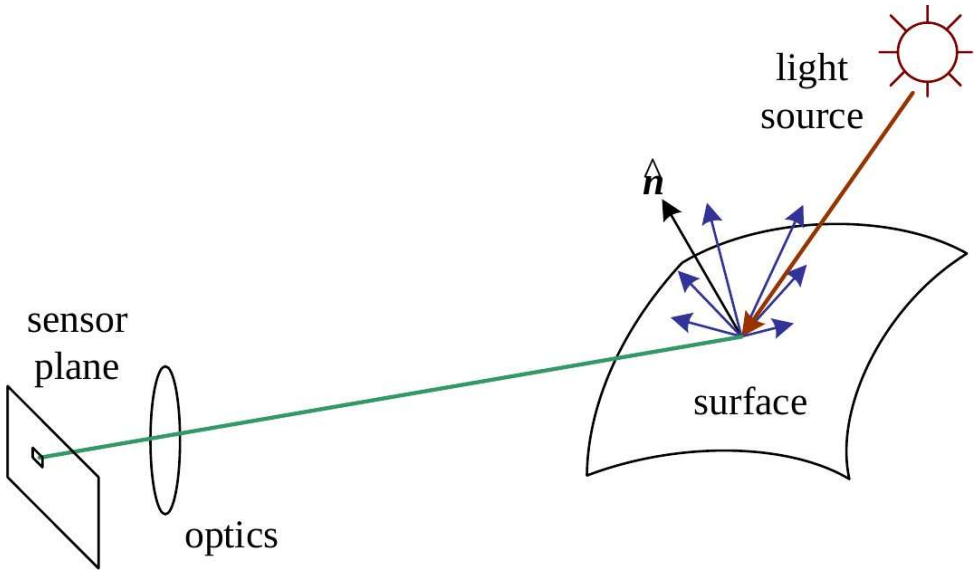


Negative radial distortion (k1=-1.5)
(Barrel distortion)

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

3. Photometric Image Formation

Photometric Image Formation



CV: often camera info not provided

- So far we have discussed how individual light rays travel through space
- We now discuss how an image is formed in terms of **pixel intensities and colors**
- Light is **emitted** by one or more light sources and **reflected** or **refracted** (once or multiple times) at surfaces of objects (or media) in the scene

Rendering Equation

Let $\mathbf{p} \in \mathbb{R}^3$ denote a 3D surface point, $\mathbf{v} \in \mathbb{R}^3$ the viewing direction and $\mathbf{s} \in \mathbb{R}^3$ the incoming light direction. The **rendering equation** describes how much of the light L_{in} with wavelength λ arriving at $\mathbf{p}$ is reflected into the viewing direction $\mathbf{v}$:

n, s 수직이면 0

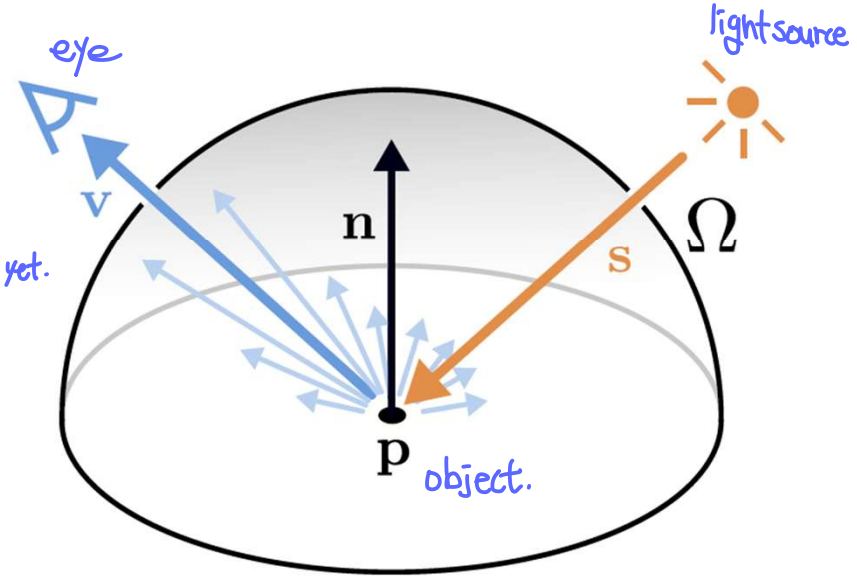
$$L_{\text{out}}(\mathbf{p}, \mathbf{v}, \lambda) = L_{\text{emit}}(\mathbf{p}, \mathbf{v}, \lambda) + \int_{\Omega} \text{BRDF}(\mathbf{p}, \mathbf{s}, \mathbf{v}, \lambda) \cdot L_{\text{in}}(\mathbf{p}, \mathbf{s}, \lambda) \cdot (-\mathbf{n}^{\top}\mathbf{s})\, d\mathbf{s}$$

- Ω is the unit hemisphere at normal $\mathbf{n}$
- The bidirectional reflectance distribution function $\text{BRDF}(\mathbf{p}, \mathbf{s}, \mathbf{v}, \lambda)$ defines how light is reflected at an opaque surface.
- $L_{\text{emit}} > 0$ only for light emitting surfaces

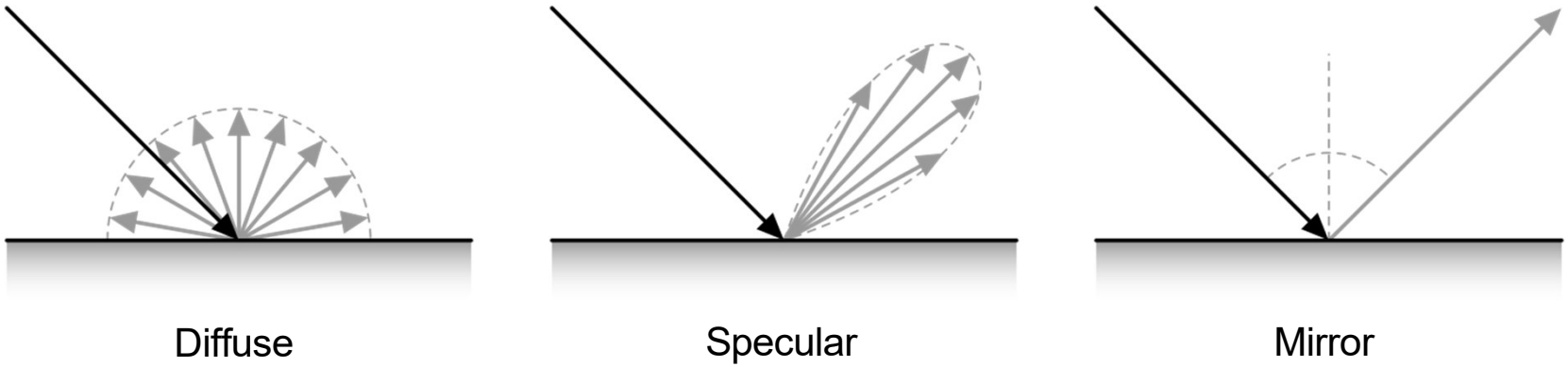
no detail provided. yet.

another light. screen이면 >0

p가 빛 안내는 일반 물체면 보통 0.



Diffuse and Specular Reflection



- Typical BRDFs have a **diffuse** and a **specular** component
- The diffuse (=constant) component scatters light uniformly in all directions
- This leads to shading, i.e., smooth variation of intensity wrt. surface norma
- I The specular component depends strongly on the outgoing light direction

Diffuse and Specular Reflection

Diffuse

constant reflection

Specular

localized reflection.

Combined

- Typical BRDFs have a **diffuse** and a **specular** component
- The diffuse (=constant) component scatters light uniformly in all directions
- This leads to shading, i.e., smooth variation of intensity wrt. surface normal
- The specular component depends strongly on the outgoing light direction

BRDF Examples

material 따라
hyperparameter 다름.

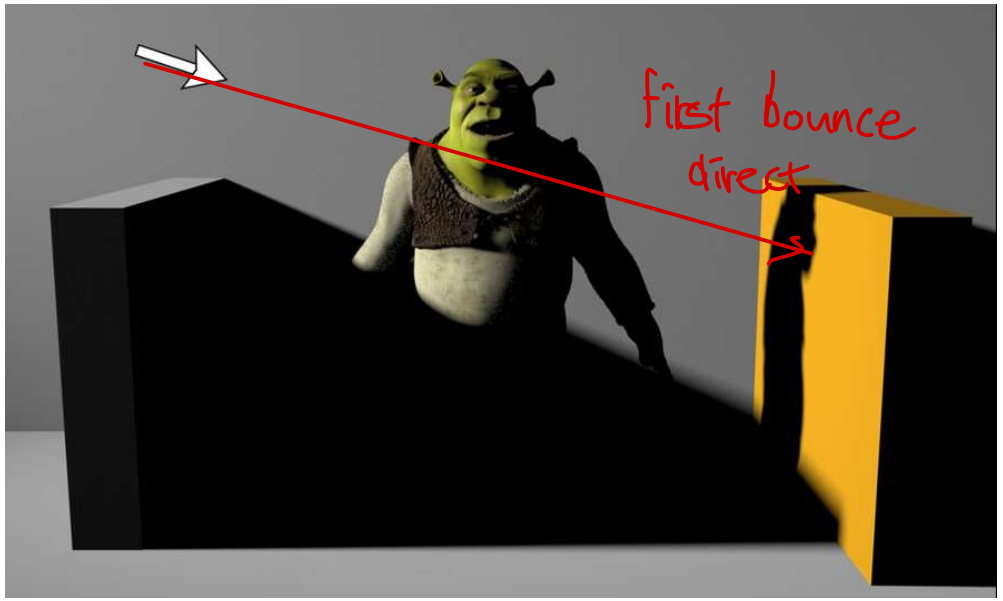
CG stuff.
no detail

- BRDFs can be very complex and spatially varying

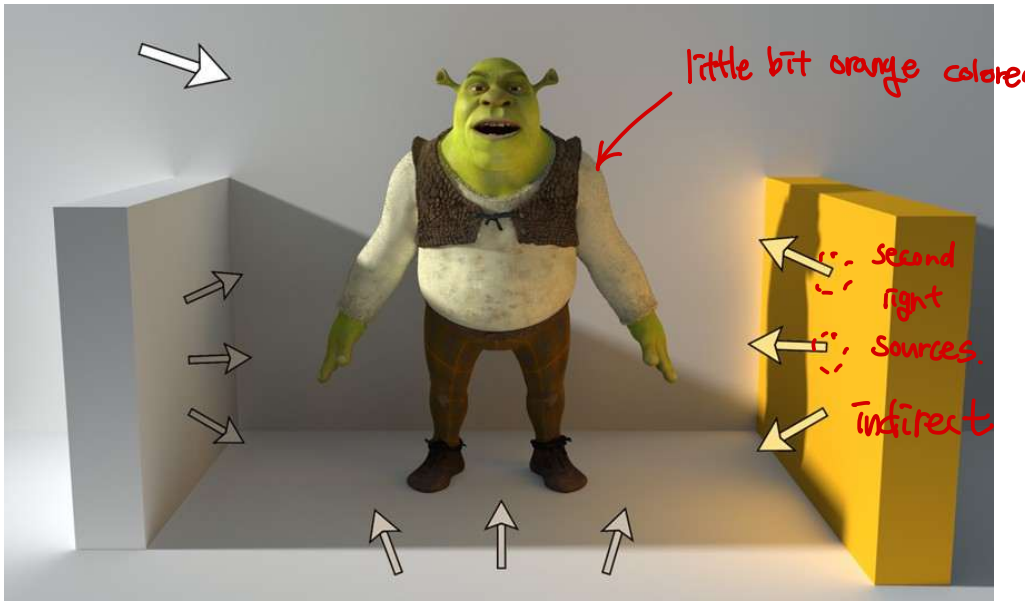
Fresnel Effect

- The amount of light reflected from a surface depends on the viewing angle

Global Illumination GI



Rendering with Direct Lighting



Rendering with Global Illumination ray tracing
indirect
실시간으로 하기
계산량 부족

- Modeling one light bounce is insufficient for rendering complex scenes
- Light sources can be shadowed by occluders and rays can bounce multiple times
- **Global illumination** techniques also take indirect illumination into account

Why Camera Lenses?

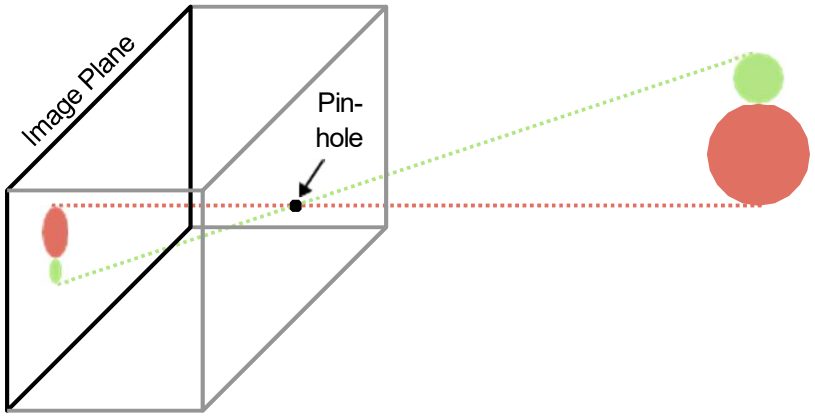
many light comes in → lack of consistency

- Large and very small pinholes result in **image blur** (averaging, diffraction)
  - ↳ longer exposure time
- Small pinholes require very **long shutter times** (⇒ motion blur)
- http://www.pauldebevec.com/Pinhole/

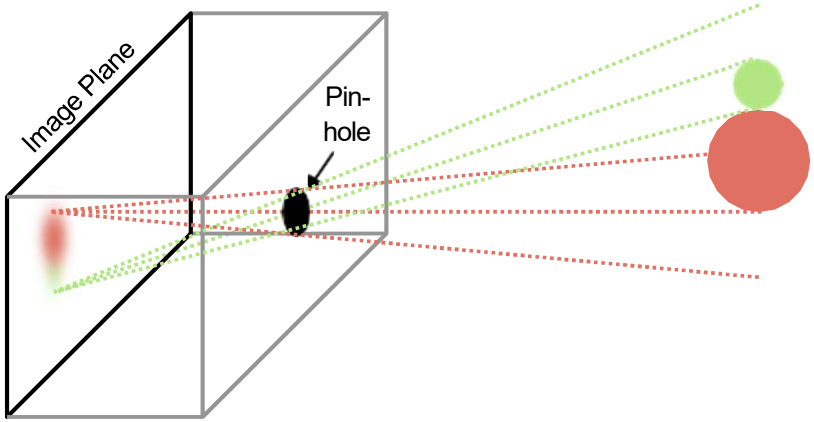
↑ ∵ if object moves.
lower consistency.

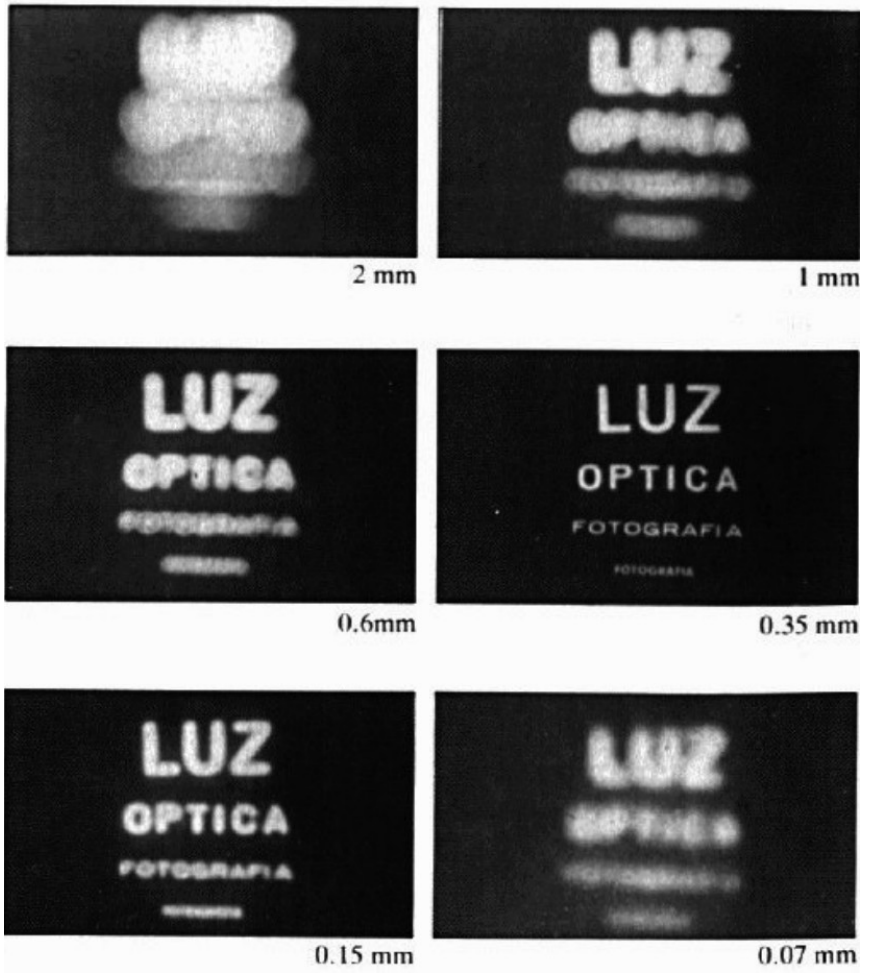
Why Camera Lenses?

Small Pinhole



Large Pinhole



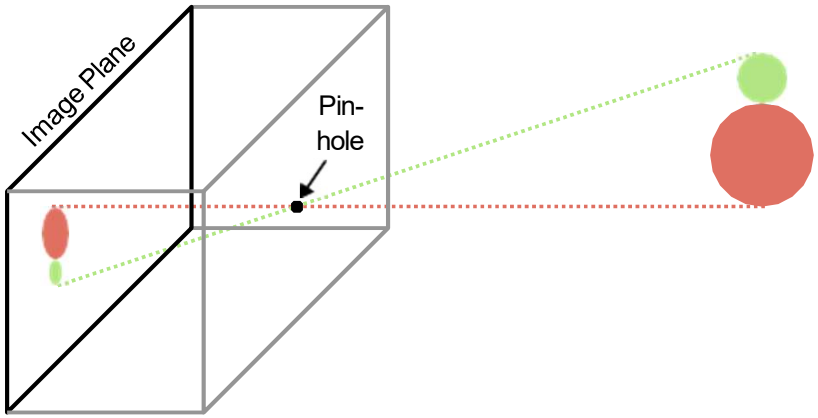


Optics

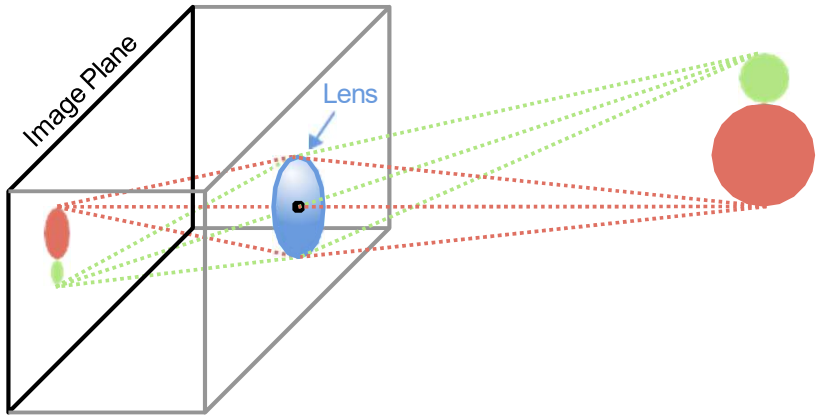
CG
CV. researcher 에게는 중요.

optic 공부할 때나해라. 너무 복잡할_

Pinhole Camera Model



Camera with Lens

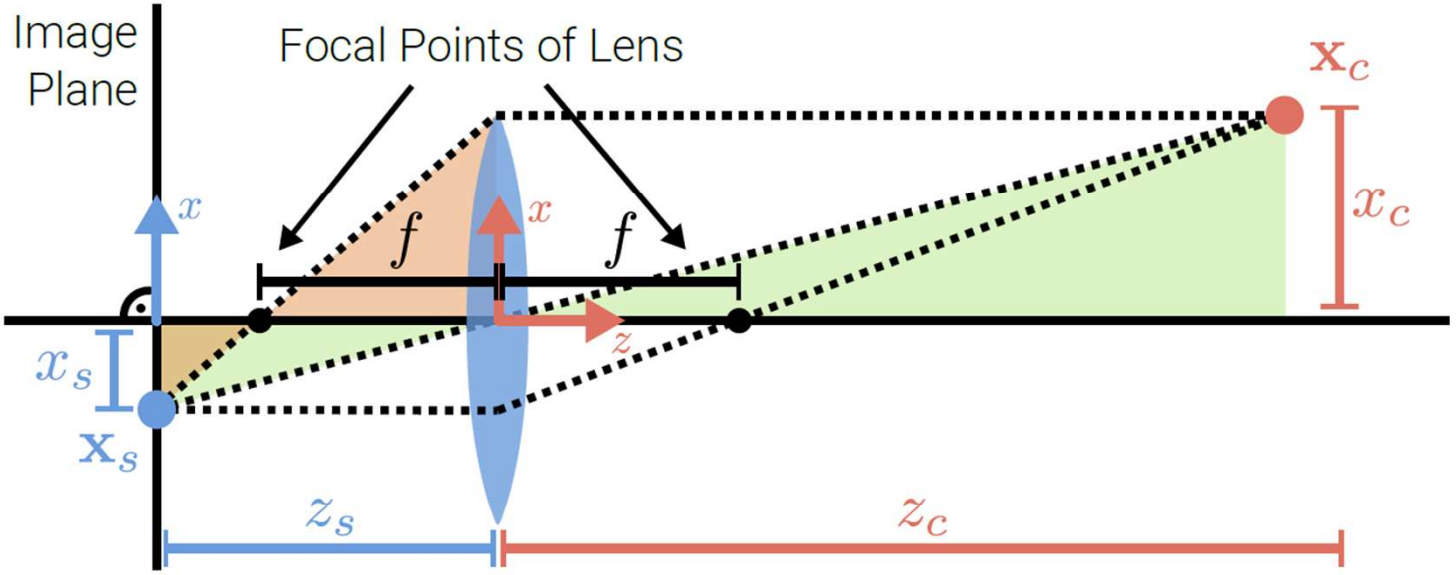


- Cameras use one or multiple **lenses** to accumulate light on the sensor plane
- Importantly, if a 3D point is in **focus**, all light rays arrive at the same 2D pixel

4 image become sharp

- For many applications it suffices to model lens cameras with a pinhole model
- However, to address **focus, vignetting and aberration** we need to model lenses

Thin Lens Model



$$\frac{x_s}{z_s - f} = \frac{x_c}{f}$$

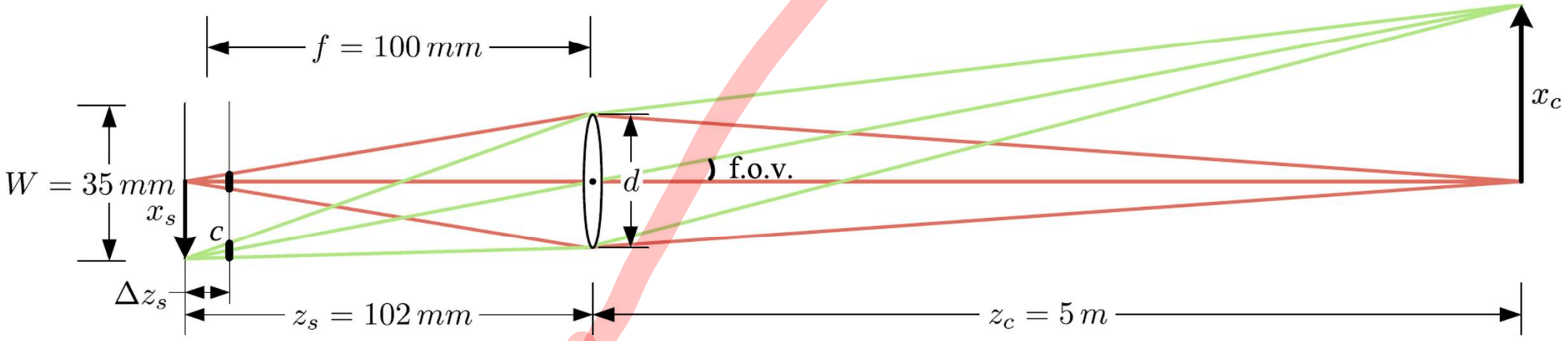
$$\frac{x_s}{z_s} = \frac{x_c}{z_c}$$

$$\frac{x_s}{x_c} = \frac{z_s - f}{f} \quad \wedge \quad \frac{x_s}{x_c} = \frac{z_s}{z_c} \quad \Rightarrow \quad \frac{z_s - f}{f} = \frac{z_s}{z_c} \quad \Rightarrow \quad \frac{z_s}{f} - 1 = \frac{z_s}{z_c} \quad \Rightarrow \quad \frac{1}{z_s} + \frac{1}{z_c} = \frac{1}{f}$$

- The **thin lens model** with spherical lens is often used as an approximation
- Properties: Axis-parallel rays pass the focal point, rays via center keep direction
- From Snell's law we obtain $f = \frac{R}{2(n-1)}$ with radius R and index of refraction n

important

Depth of Field (DOF)



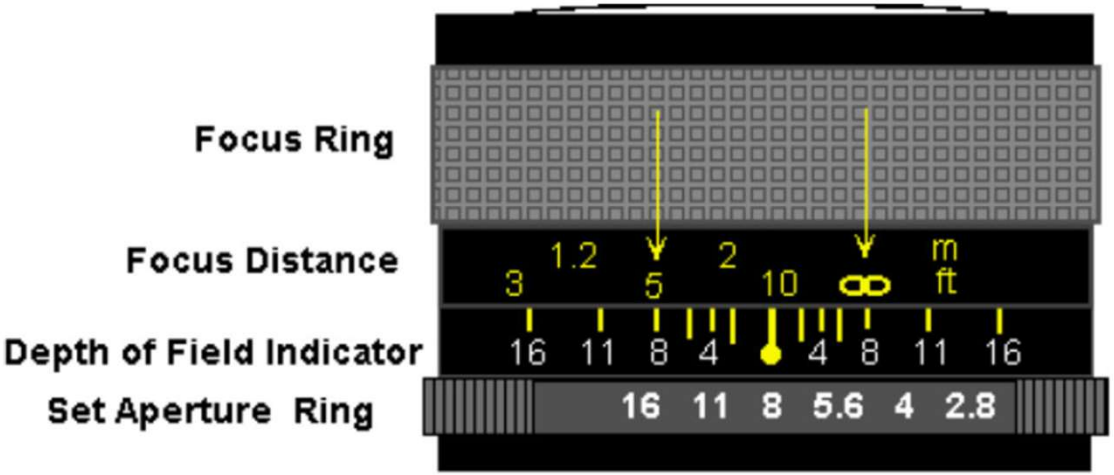
- The image is **in focus** if $\frac{1}{z_s} + \frac{1}{z_c} = \frac{1}{f}$ where f is the focal length of the lens
- For $z_c \to \infty$ we obtain $z_s = f$ (lens with focal length $f \approx$ pinhole at distance f)
- If the image plane is **out of focus**, a 3D point projects to the **circle of confusion** c

↳ 사진은 2D space에 projection 됨.

Depth of Field (DOF)

- To control the **size of the circle of confusion,** we change the lens **aperture**
- An aperture is a hole or an opening through which light travels
- The aperture limits the amount of light that can reach the image plane
- Smaller apertures lead to sharper, but more noisy images (less photons)

Depth of Field (DOF)



- The allowable depth variation that limits the circle of confusion c is called **depth of field** and is a function of both the focus distance and the lens aperture
- Typical DSLR lenses have depth of field indicators
- The commonly displayed **f-number** is defined as

$$N = \frac{f}{d} \qquad \text{(often denoted as } f/N\text{, e.g.: } f/1.4\text{)}$$

- In other words, it is the lens focal length f divided by the aperture diameter d

Depth of Field (DOF)

Aperture = f/1.4
DOF = 0.8 cm

Aperture = f/4.0
DOF = 2.2 cm

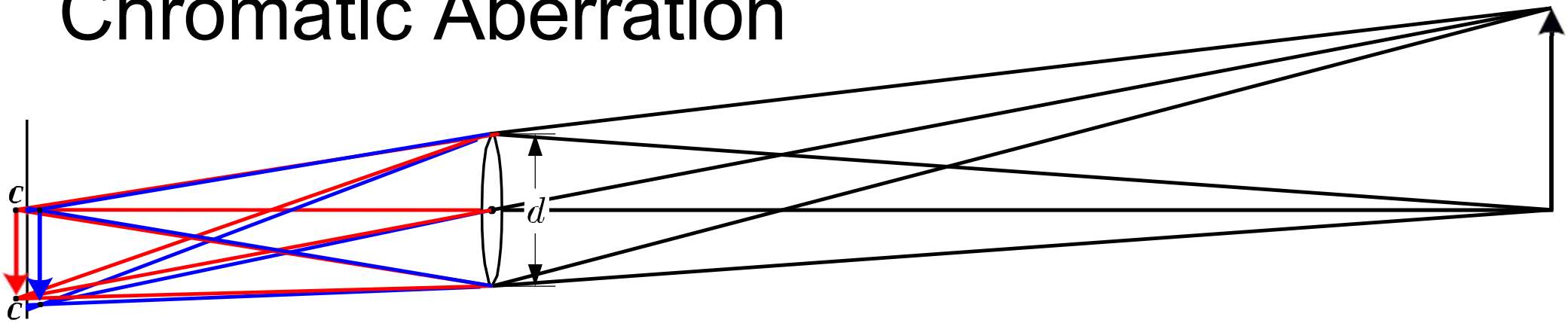
Aperture = f/22
DOF = 12.4 cm

Depth of Field (DOF):

- Distance between the nearest and farthest objects that are acceptably sharp
- Decreasing the aperture diameter (increasing the f-number) increases the DOF

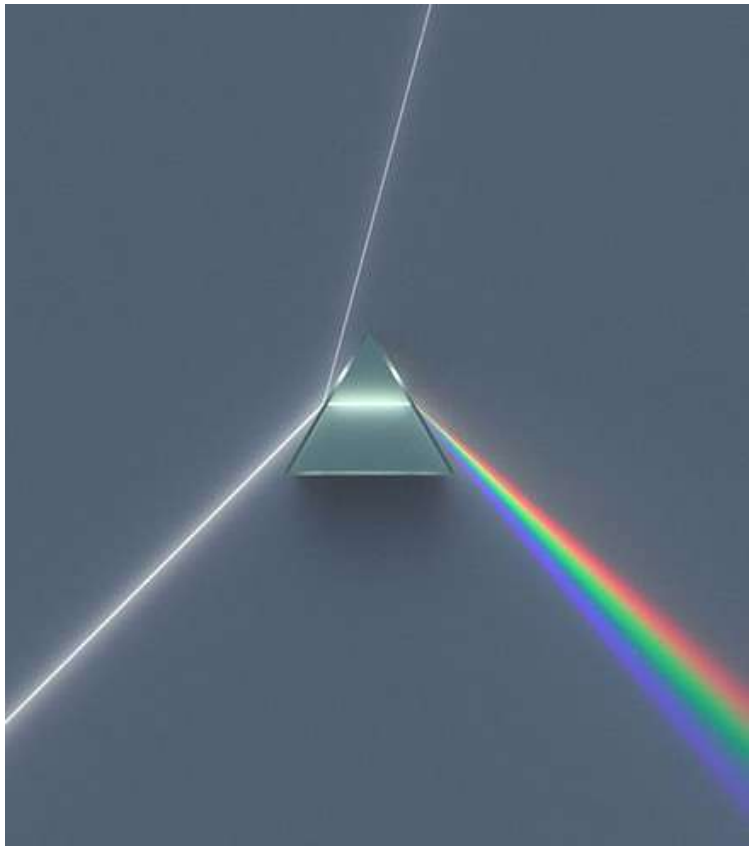
not important

Chromatic Aberration



- The **index of refraction** for glass varies slightly as a function of wavelength
- Thus, simple lenses suffer from **chromatic aberration** which is the tendency for light of different colors to focus at slightly different distances (blur, color shift)
- To reduce chromatic and other kinds of aberrations, most photographic lenses are compound lenses made of different glass elements (with different coatings)

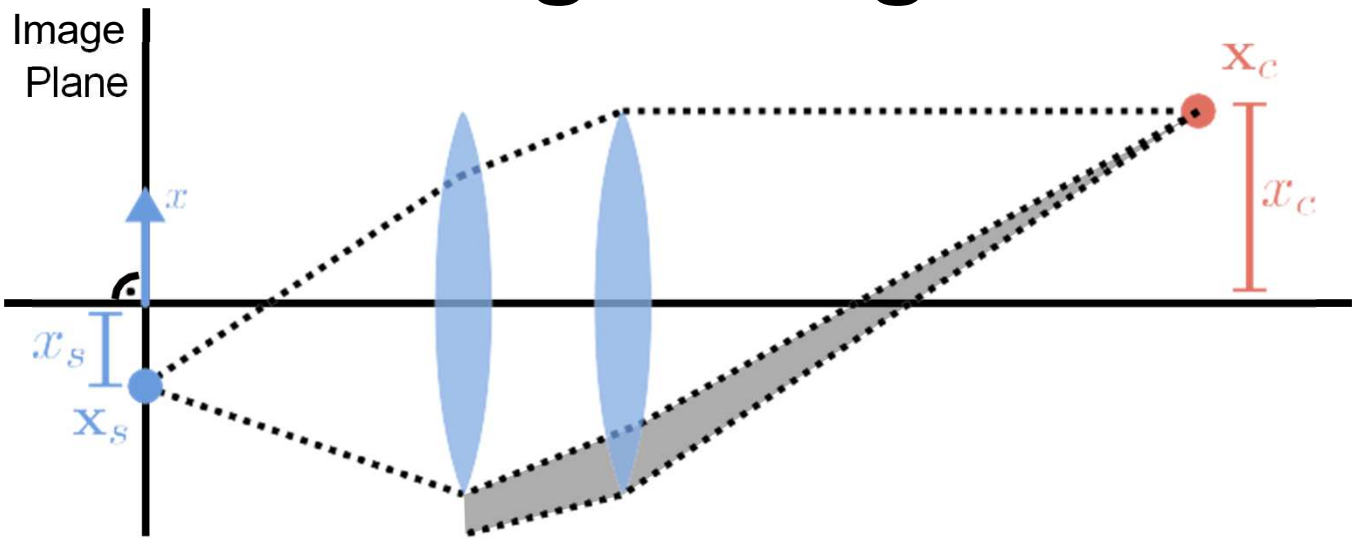
Chromatic Aberration



- Top: High-quality lens Bottom: Low-quality lens (blur, rainbow edges)

Vignetting

not important

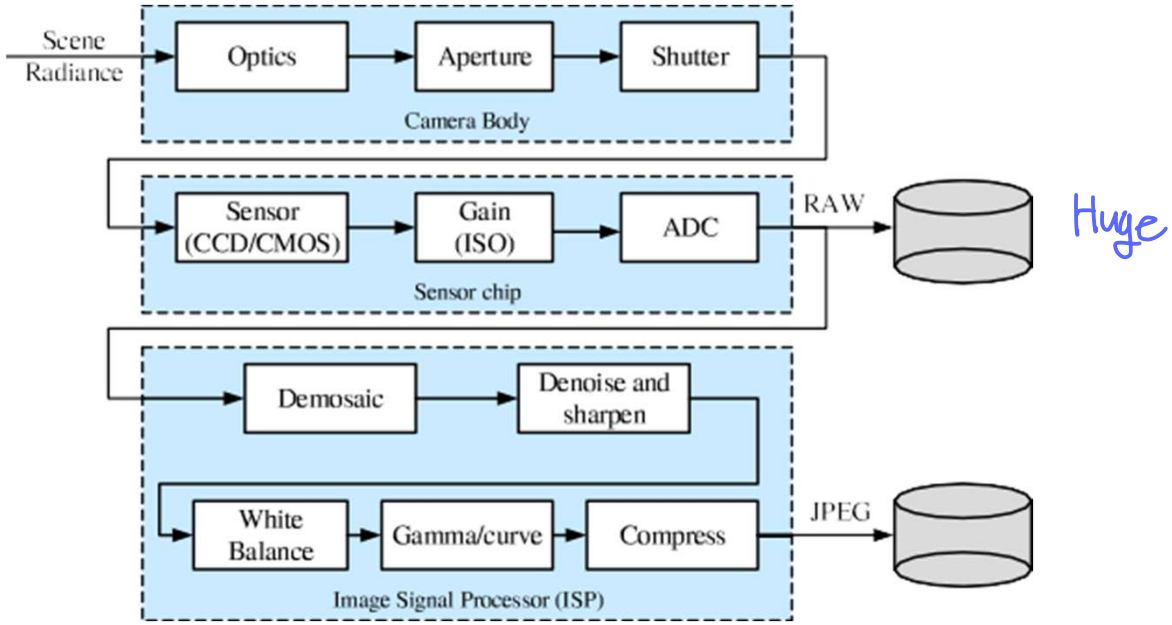


- Vignetting is the tendency for the brightness to fall off towards the image edge
- Composition of two effects: natural and mechanical vignetting
- Natural vignetting: foreshortening of object surface and lens aperture
- Mechanical vignetting: the shaded part of the beam never reaches the image
- Vignetting can be calibrated (i.e., undone)

Vignetting

4. Image Sensing Pipeline

Image Sensing Pipeline



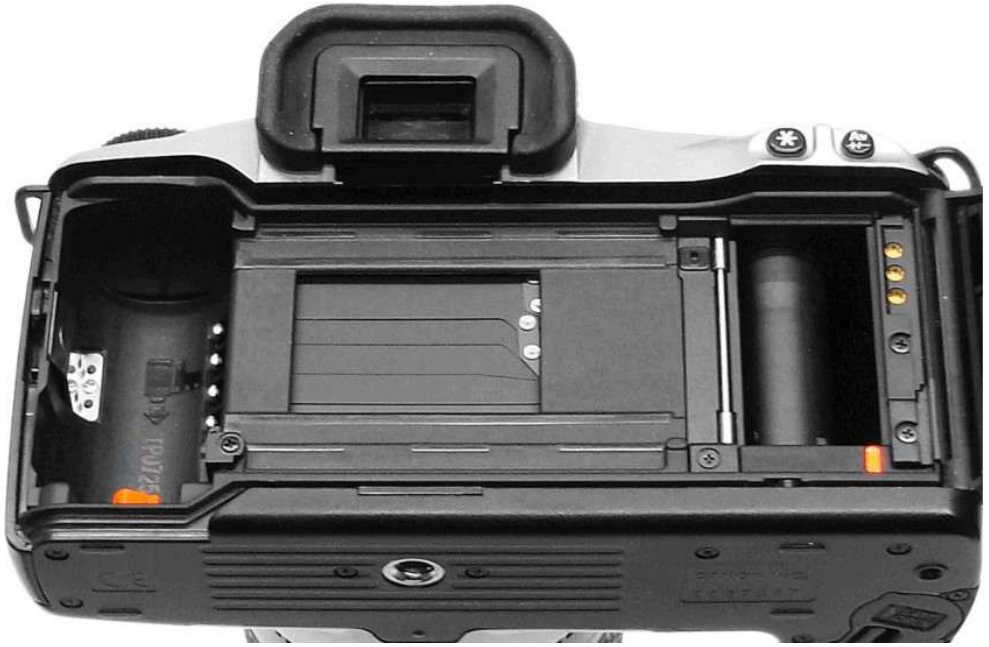
Huge

The **image sensing pipeline** can be divided into three stages:

- **Physical light transport** in the camera lens/body
- **Photon measurement** and conversion on the sensor chip
- **Image signal processing (ISP)** and image compression

CV

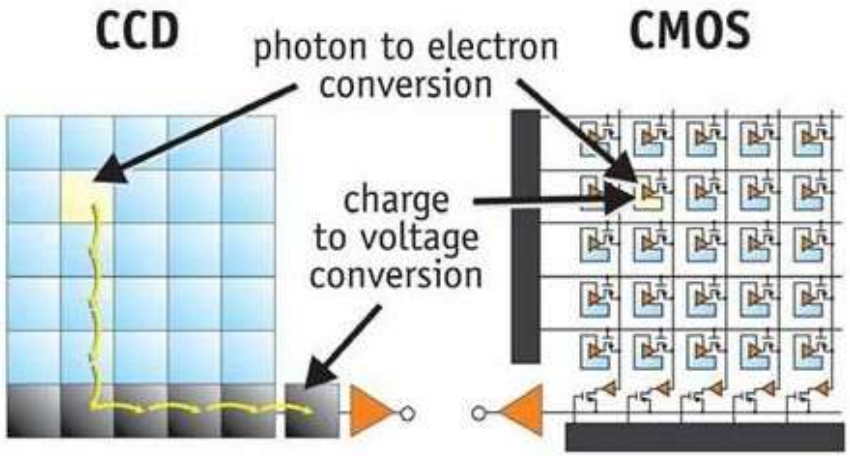
Shutter

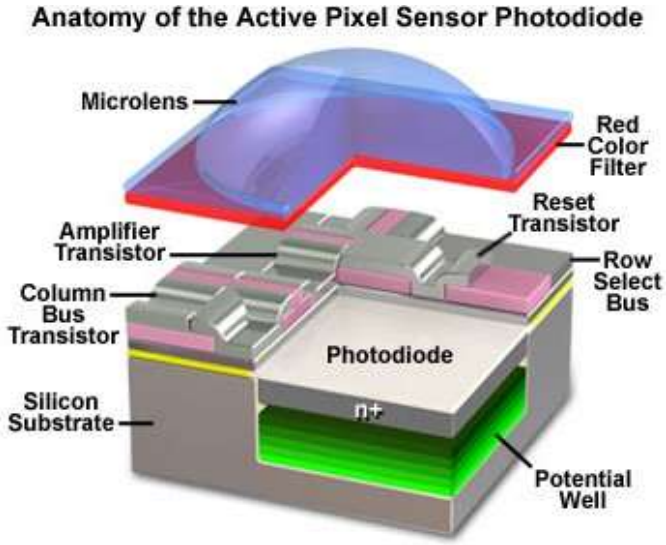


- A **focal plane shutter** is positioned just in front the image sensor / film
- Most digital cameras use a combination of mechanical and electronic shutter
- The shutter speed (exposure time) controls how much light reaches the sensor
- It determines if an image appears over-/underexposed, blurred or noisy

밝에는 노출시간을 늘여서 빛을 확보함 ⇒ 비나가 심해짐.

Sensor





- **CCDs** move charge from pixel to pixel and convert it to voltage at the output node
- **CMOS** images convert charge to voltage inside each pixel and are standard
- Larger chips (full frame = 35 mm) are more photo-sensitive ⇒ less noise

https://meroli.web.cern.ch/lecture_cmos_vs_ccd_pixel_sensor.html

Color Filter Arrays

only receive G color

G	R	G	R
B	G	B	G
G	R	G	R
B	G	B	G

Bayer RGB Pattern

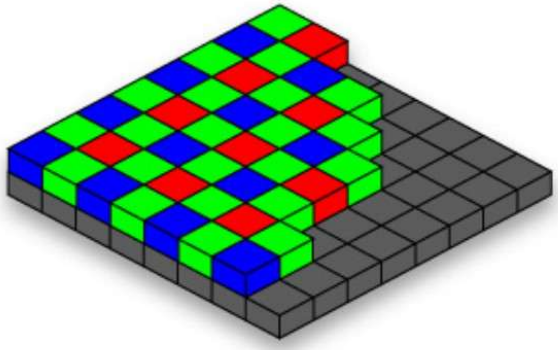
rGb	Rgb	rGb	Rgb
rgB	rGb	rgB	rGb
rGb	Rgb	rGb	Rgb
rgB	rGb	rgB	rGb

Interpolated Pixels

- To measure color, pixels are arranged in a **color array,** e.g.: Bayer RGB pattern
- Missing colors at each pixel are interpolated from the neighbors (demosaicing)

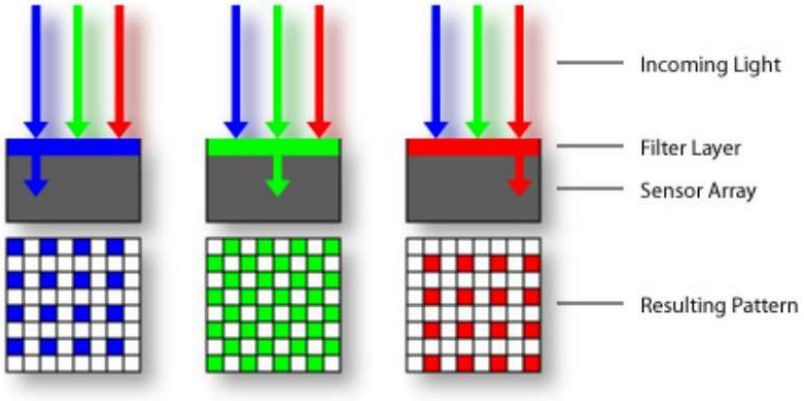
Color Filter Arrays

Bayer grid



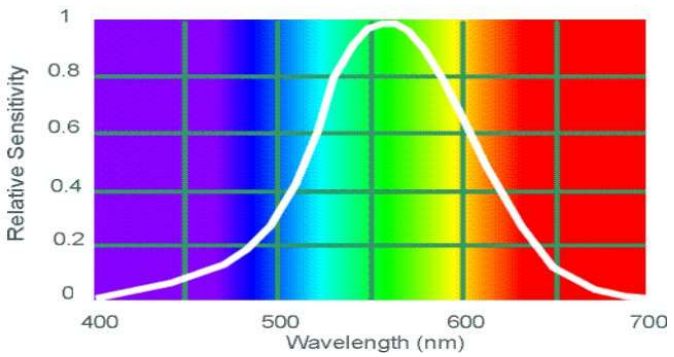
Estimate missing components from neighboring values (demosaicing)

Interpolation.

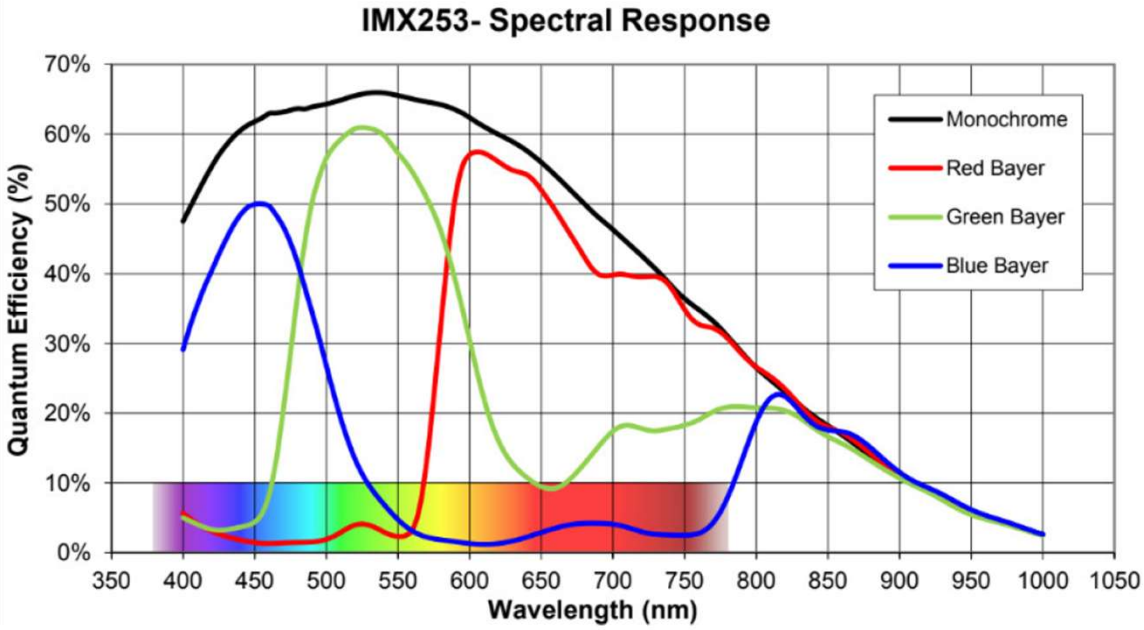


Why more green?

인간은 green에 더 sensetive 함.



Color Filter Arrays



- Each pixel **integrates the light spectrum** L according to its spectral sensitivity S:

$$R = \int L(\lambda)\, S_R(\lambda)\, d\lambda$$

- The spectral response curves are provided by the camera manufacturer

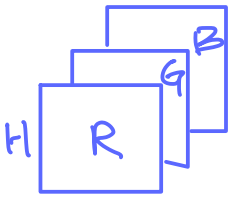
Color Spaces


RGB


R


G


B



H W numbers (intensity) 0-255

Red (255, 0, 0)

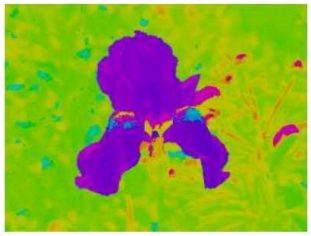
0 : dark
255 : bright.

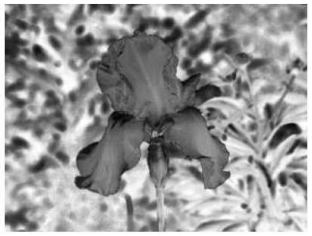

L*


a*


b*

Lab


H


S

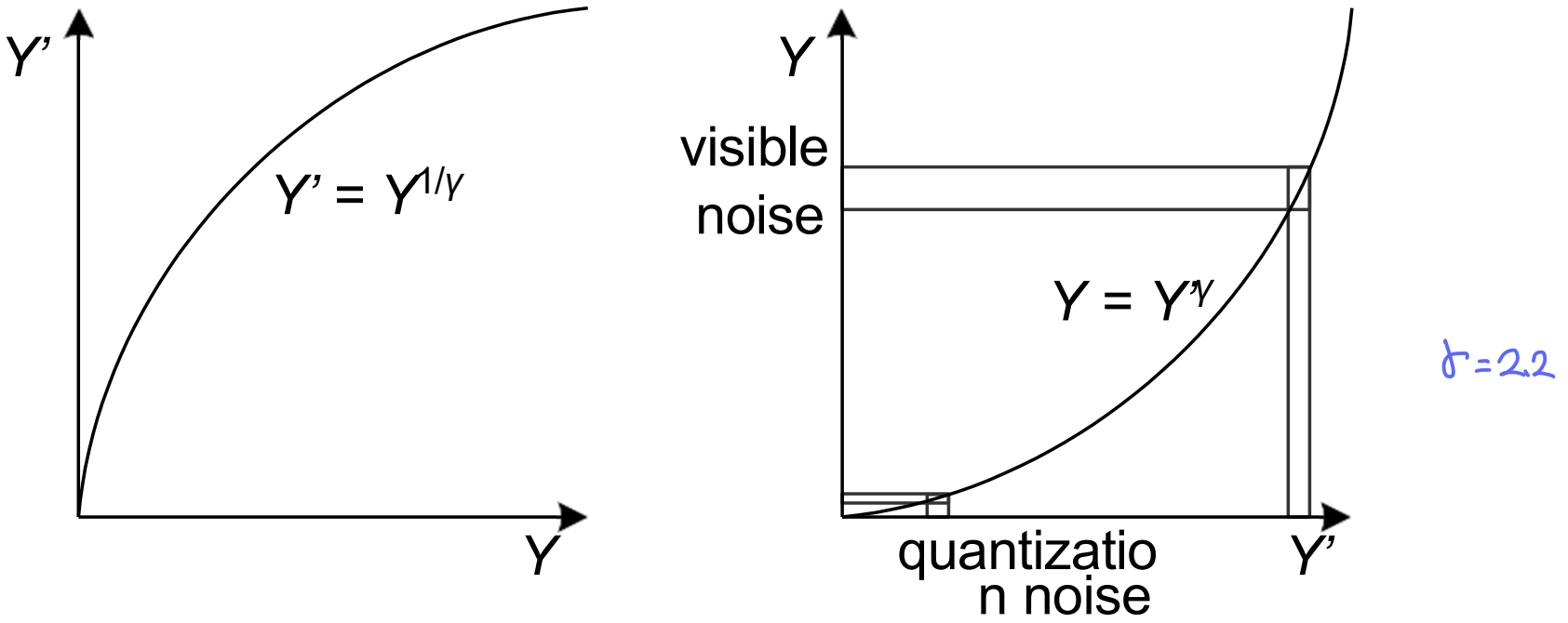

V

HSV

- Various different **color spaces** have been developed and are used in practice

→ RAW

Gamma Compression



- Humans are more sensitive to intensity differences in darker regions
- Therefore, it is beneficial to **nonlinearly transform** (left) the intensities or colors prior to discretization (left) and to undo this transformation during loading

Image Compression

brightness color

- Typically luminance is compressed with higher fidelity than chrominance
- Often, (8 × 8 pixel) patch-based discrete cosine or wavelet transforms are used
- **Discrete Cosine Transform (DCT)** is an approximation to PCA on natural images
- The coefficients are quantized to integers that can be stored with Huffman codes
- More recently, deep network based compression algorithms are developed

Lens Distortion