18-799: Applied Computer Vision

Spring 2026

Image Formation

Carnegie Mellon University
Africa

Objectives

In today's class we will learn

- **Homogeneous Coordinates and transformation**
- **Geometric primitive**
- Geometric Image formation
- Photometric image formation

Homogeneous Coordinates

- In computer vision, image formation is a chain of transformations:
- World $\longrightarrow$ Camera $\longrightarrow$ Image
- If each step is linear:

$$\mathbf{x}' = A\mathbf{x},$$
$$\mathbf{x}'' = B\mathbf{x}'$$

  – then the whole pipeline collapses into:

$$\mathbf{x}'' = (BA)\mathbf{x}.$$

Carnegie Mellon University Africa

Homogeneous Coordinates

- Let’s consider the function f , where:

$$f\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{bmatrix} x+1 \\ y \end{bmatrix}$$

- Can we represent f with a matrix?
  - That’s, is there a matrix A such that:

$$A\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x+1 \\ y \end{bmatrix}$$

Homogeneous Coordinates

- Points in Euclidean Coordinates
  - In standard 2D space, a point is represented as a vector

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

- Points in Homogeneous Coordinates
  - The same point is represented by the augmented vector

$$\bar{\mathbf{x}} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}.$$

- More generally, the Euclidean point $\mathbf{x}$ corresponds to the entire equivalence class

$$\tilde{\mathbf{x}} \sim \lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \lambda x \\ \lambda y \\ \lambda \end{bmatrix}, \lambda \neq 0$$

- All such homogeneous vectors represent the same geometric point after normalization.

Homogeneous Coordinates

- Euclidean point $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$ corresponds to the entire equivalence class in homogeneous coordinates:

$$\tilde{\mathbf{x}} \sim \lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \lambda x \\ \lambda y \\ \lambda \end{bmatrix}, \lambda \neq 0$$

  - Since λ can be anything ($\lambda \neq 0$), then $\tilde{\mathbf{x}}$ could be re-written $\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{bmatrix}$

- To convert a homogeneous vector $\tilde{\mathbf{x}}$ back to inhomogeneous form, we divide by the last coordinate $\tilde{w}$:

$$\bar{\mathbf{x}} = \frac{1}{\tilde{w}} \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{pmatrix} = \begin{pmatrix} \tilde{x}/\tilde{w} \\ \tilde{y}/\tilde{w} \\ 1 \end{pmatrix}, \tilde{w} \neq 0.$$

- The corresponding inhomogeneous point is obtained by dropping the last coordinate:

$$\mathbf{x} = \begin{pmatrix} \tilde{x}/\tilde{w} \\ \tilde{y}/\tilde{w} \end{pmatrix}.$$

Homogeneous Coordinates

- Let's re-consider the function f , where:

$$f\left(\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}\right) = \begin{bmatrix} x+1 \\ y \\ 1 \end{bmatrix}$$

- Can we represent f with a matrix?
  - That's, is there a matrix A such that:

$$A\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x+1 \\ y \\ 1 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Geometric primitive

- Geometric primitives are the basic building blocks used to describe 3D shapes

- Examples: Points, lines, planes

2D Points

- 2D points can be written in inhomogeneous coordinates as $\mathbf{x} = \binom{x}{y} \in \mathbb{R}^2$

- or in homogeneous coordinates as $\tilde{\mathbf{x}} = \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{pmatrix} \in \mathbb{P}^2$ where $\mathbb{P}^2 = \mathbb{R}^3 \setminus \{(0,0,0)\}$ is called projective space.

- Homogeneous vectors that differ only by scale are considered equivalent and define an equivalence class.

2D Points

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

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

  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

2D Lines

- 2D lines can also be expressed using homogeneous coordinates

$$\tilde{\mathbf{l}} = (a, b, c)^\top : \{\bar{\mathbf{x}} \mid \tilde{\mathbf{l}}^\top \bar{\mathbf{x}} = 0\} \Leftrightarrow \{x, y \mid ax + by + c = 0\}$$

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

Cross Product

- Cross product of two vector:

$$\mathbf{a} \times \mathbf{b} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix}$$

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

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

- **See** Levi-Civita symbol for more info.

Cross Product

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

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

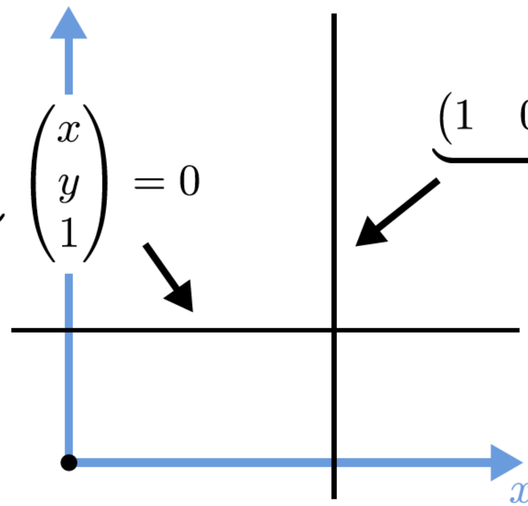
- Similarly, the line joining two points can be compactly written as:

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

2D Line: Example

$$\underbrace{\begin{pmatrix}0 & 1 & -1\end{pmatrix}}_{\tilde{\mathbf{l}}_1^T} \begin{pmatrix}x \\ y \\ 1\end{pmatrix} = 0$$

$$\underbrace{\begin{pmatrix}1 & 0 & -2\end{pmatrix}}_{\tilde{\mathbf{l}}_2^T} \begin{pmatrix}x \\ y \\ 1\end{pmatrix} = 0$$



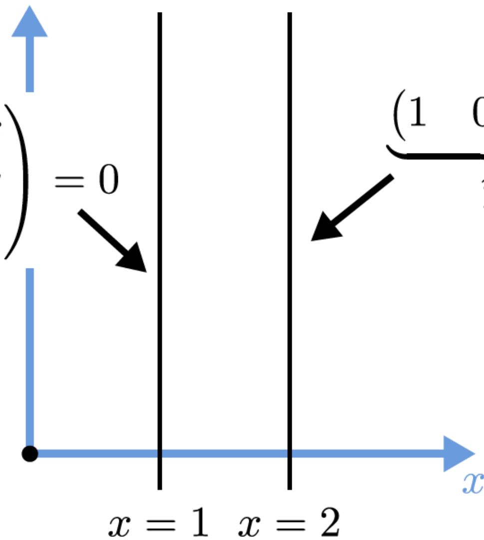
$$[\mathbf{x}]_\times = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}$$

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

2D Line: Example

$$\underbrace{\begin{pmatrix}1 & 0 & -1\end{pmatrix}}_{\tilde{\mathbf{l}}_1^T}\begin{pmatrix}x\\y\\1\end{pmatrix}=0$$

$$\underbrace{\begin{pmatrix}1 & 0 & -2\end{pmatrix}}_{\tilde{\mathbf{l}}_2^T}\begin{pmatrix}x\\y\\1\end{pmatrix}=0$$



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

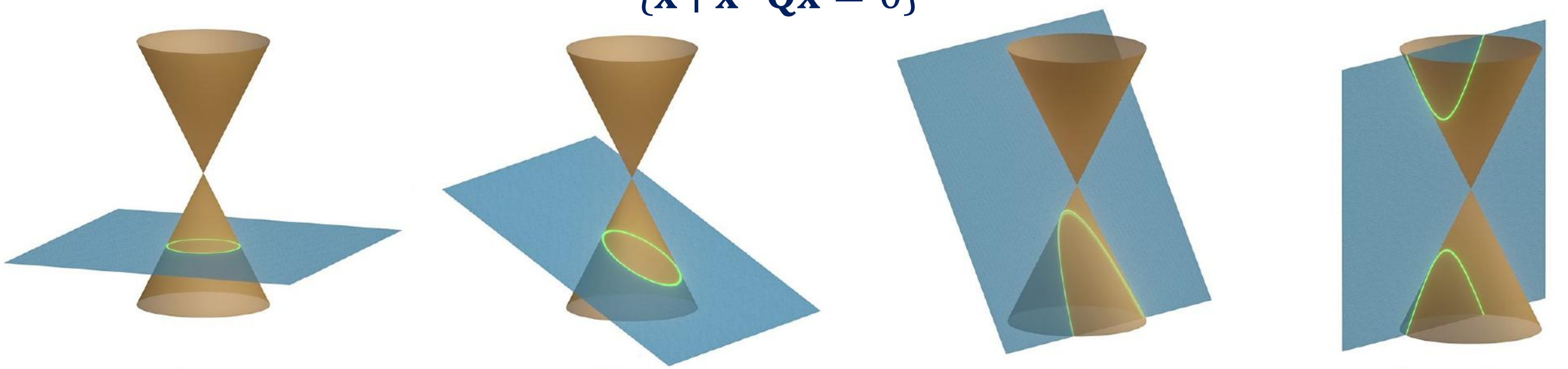
$$\tilde{\mathbf{l}}_1\times\tilde{\mathbf{l}}_2=[\tilde{\mathbf{l}}_1]_\times\tilde{\mathbf{l}}_2=\begin{bmatrix}0 & 1 & 0\\ -1 & 0 & -1\\ 0 & 1 & 0\end{bmatrix}\begin{pmatrix}1\\0\\-2\end{pmatrix}=\begin{pmatrix}0\\1\\0\end{pmatrix}$$

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

2D Conics

- More complex algebraic objects can be represented using polynomial homogeneous equations.
- For example, conic sections (arising as the intersection of a plane and a 3D cone) can be written using quadric equations:

$$\{\bar{\mathbf{x}} \mid \bar{\mathbf{x}}^\top \mathbf{Q} \bar{\mathbf{x}} = 0\}$$



Circle Ellipse Parabola Hyperbola

3D Quadrics

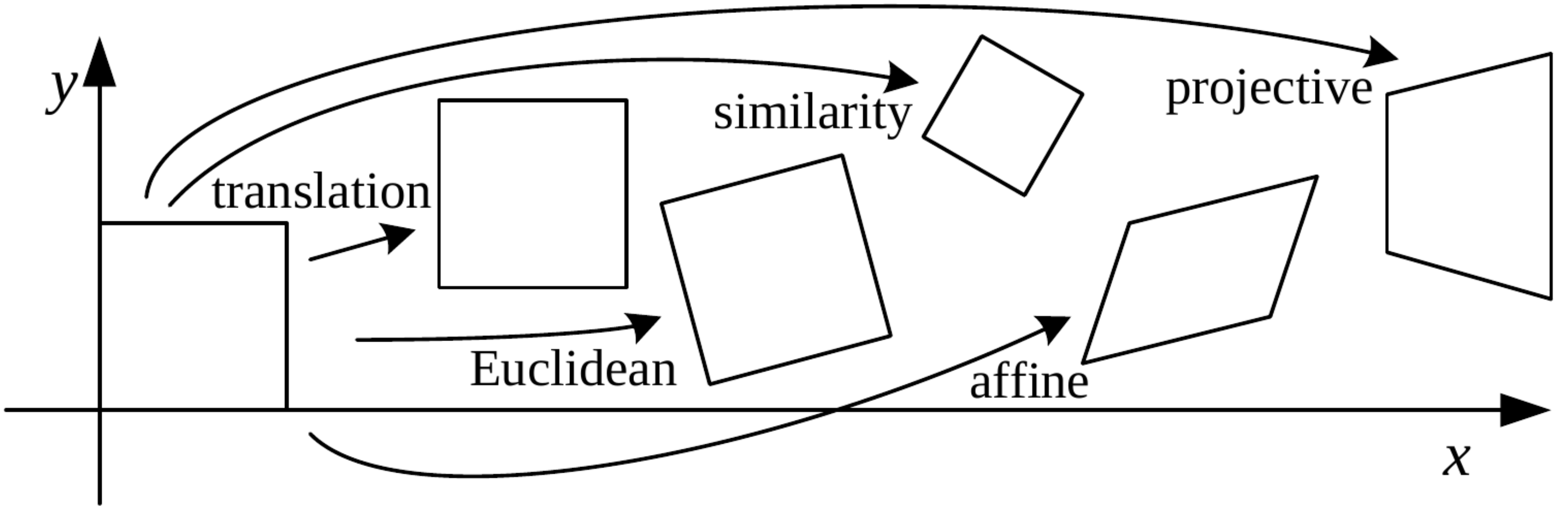
The 3D analog of 2D conics is a quadric surface:

$$\{\bar{\mathbf{x}} \mid \bar{\mathbf{x}}^\top \mathbf{Q} \bar{\mathbf{x}} = 0\}$$

Superquadrics (generalization of quadrics) for shape abstraction and compression.

Paschalidou, Ulusoy and Geiger: Superquadrics Revisited: Learning 3D Shape Parsing beyond Cuboids. CVPR, 2019

Translation

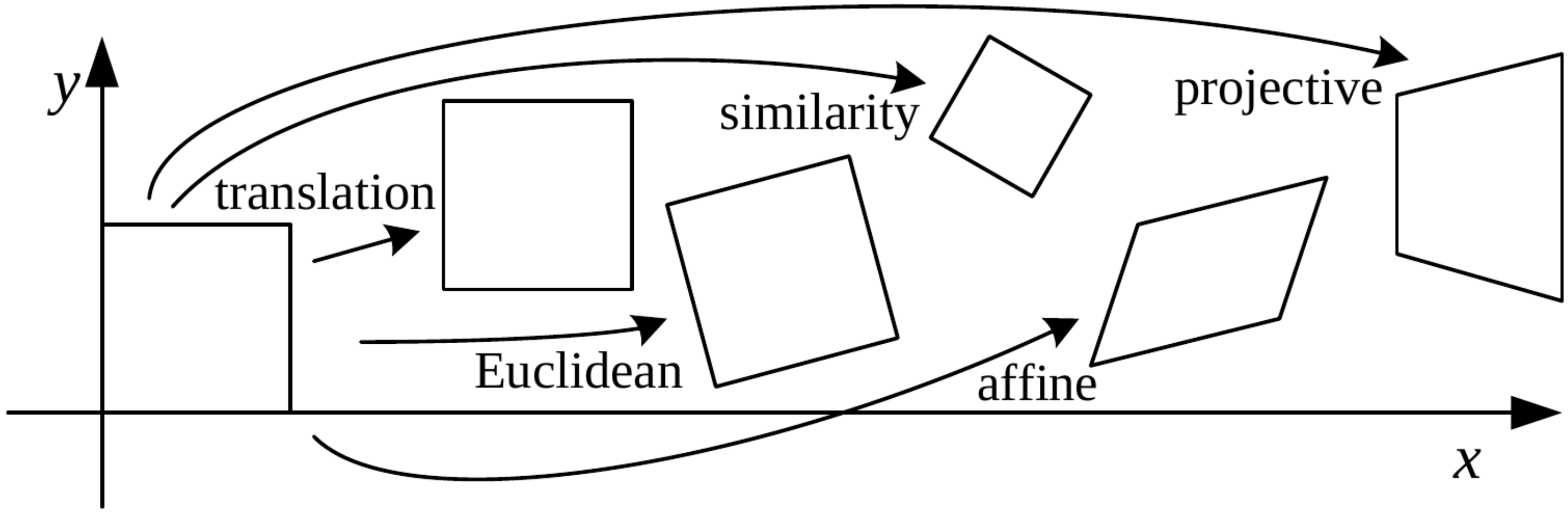


- (2D Translation of the Input, 2 DoF)

$$\mathbf{x}' = \mathbf{x} + \mathbf{t} \Leftrightarrow \bar{\mathbf{x}}' = \begin{bmatrix} \mathbf{I} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \bar{\mathbf{x}} \Rightarrow \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

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

Affine

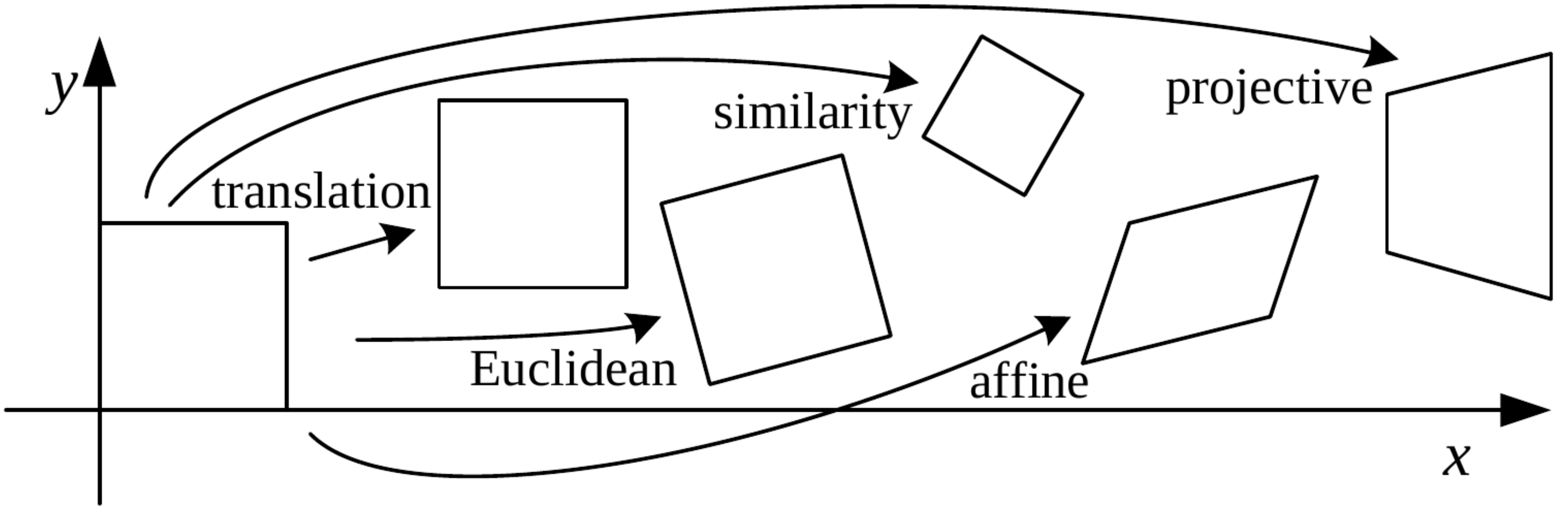


- Equivalent to 2D Linear Transformation, 6 DoF

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

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

Projective

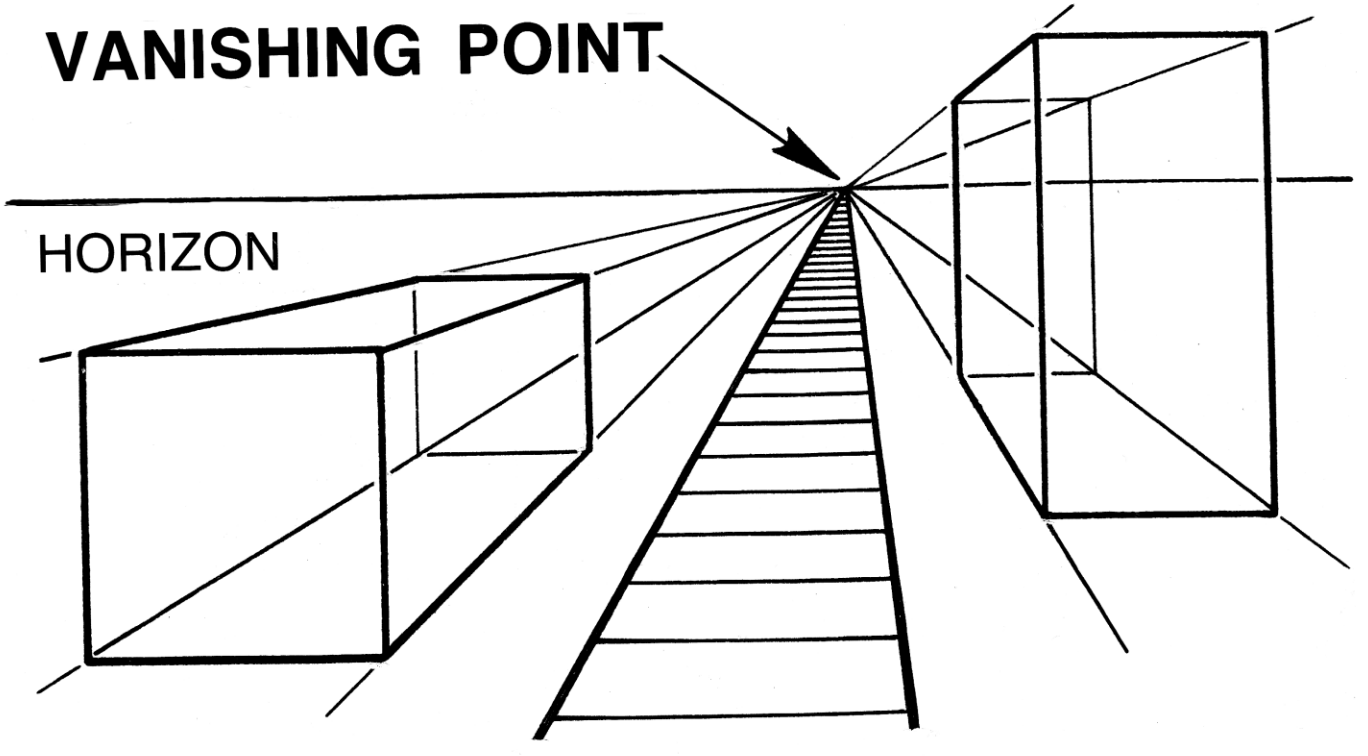


- Homography, 8 DoF

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

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

Projective



©commons.wikimedia.org

Projective

- A homography is defined in homogeneous coordinates as

$$\tilde{\mathbf{x}}' = H\tilde{\mathbf{x}}, H \in \mathbb{R}^{3\times 3}.$$

- Write the point (x, y) in homogeneous form:

$$\tilde{\mathbf{x}} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Let

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}.$$

- We now have:

$$\begin{bmatrix} \tilde{x}' \\ \tilde{y}' \\ \tilde{w}' \end{bmatrix} = \begin{bmatrix} h_{11}x + h_{12}y + h_{13} \\ h_{21}x + h_{22}y + h_{23} \\ h_{31}x + h_{32}y + h_{33} \end{bmatrix}.$$

Projective

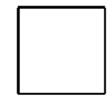
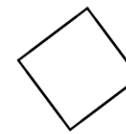
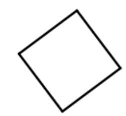
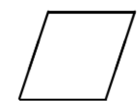
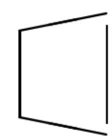
- Homogeneous coordinates represent points up to scale.
  To recover Euclidean coordinates, we divide by the last component:

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

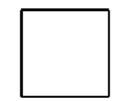
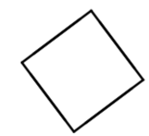
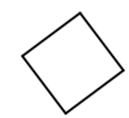
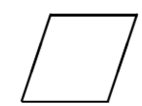
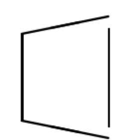
- Substituting gives exactly:

$$\begin{aligned} x' &= \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}} \\ y' &= \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}} \end{aligned}$$

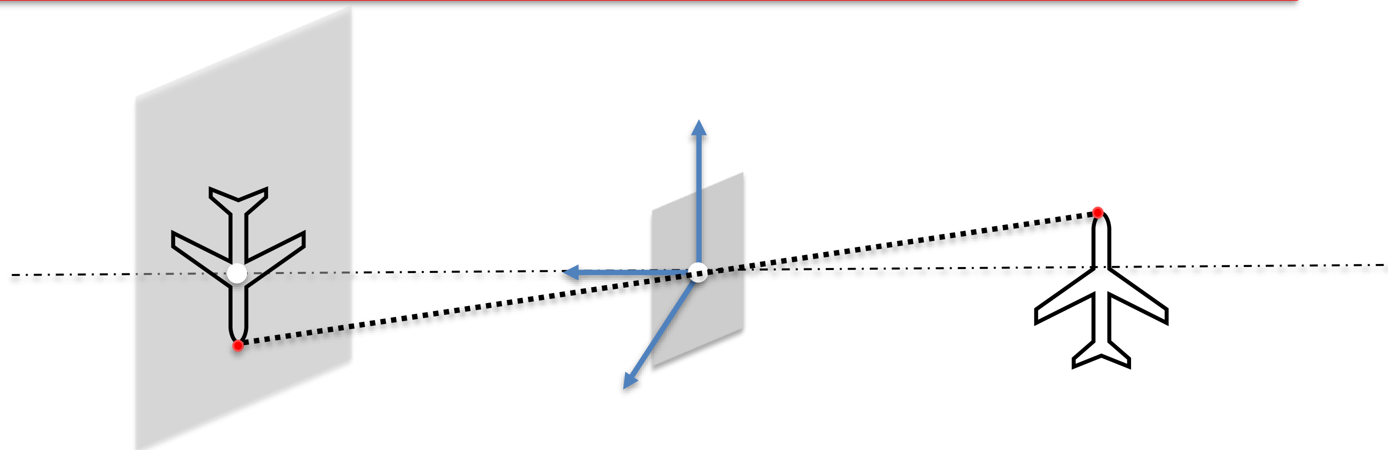
2D Transformation

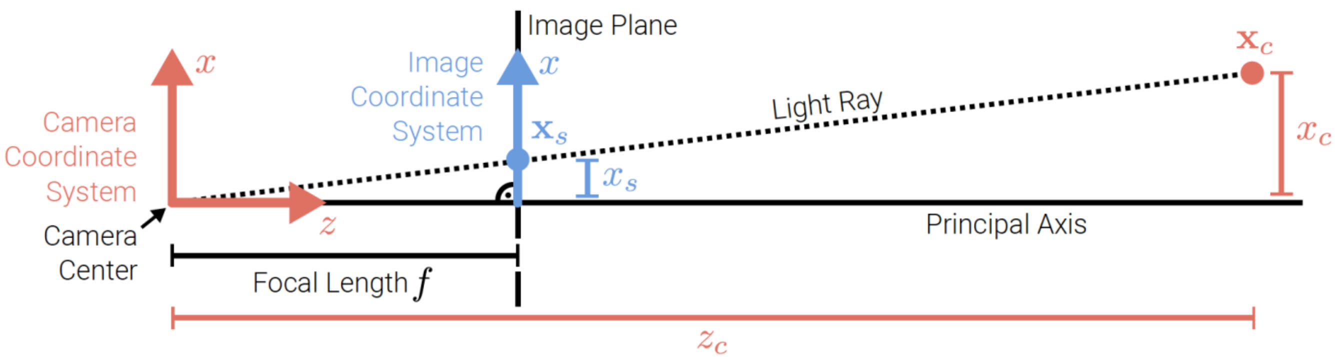
Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} \mathbf{I} & \mathbf{t} \end{bmatrix}_{2\times 3}$	2	orientation	
rigid (Euclidean)	$\begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}_{2\times 3}$	3	lengths	
similarity	$\begin{bmatrix} s\mathbf{R} & \mathbf{t} \end{bmatrix}_{2\times 3}$	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	

3D Transformation

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

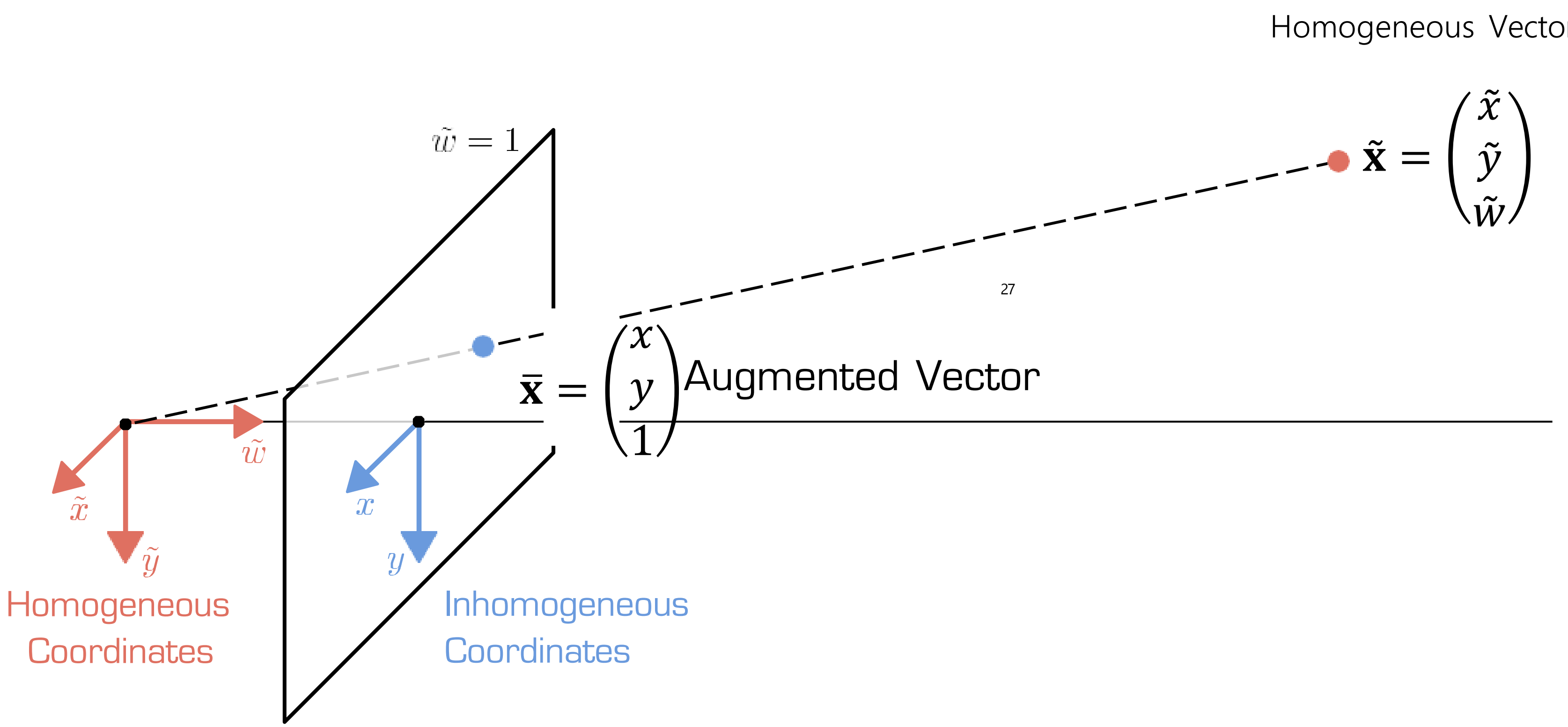
Geometric Image Formation





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

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

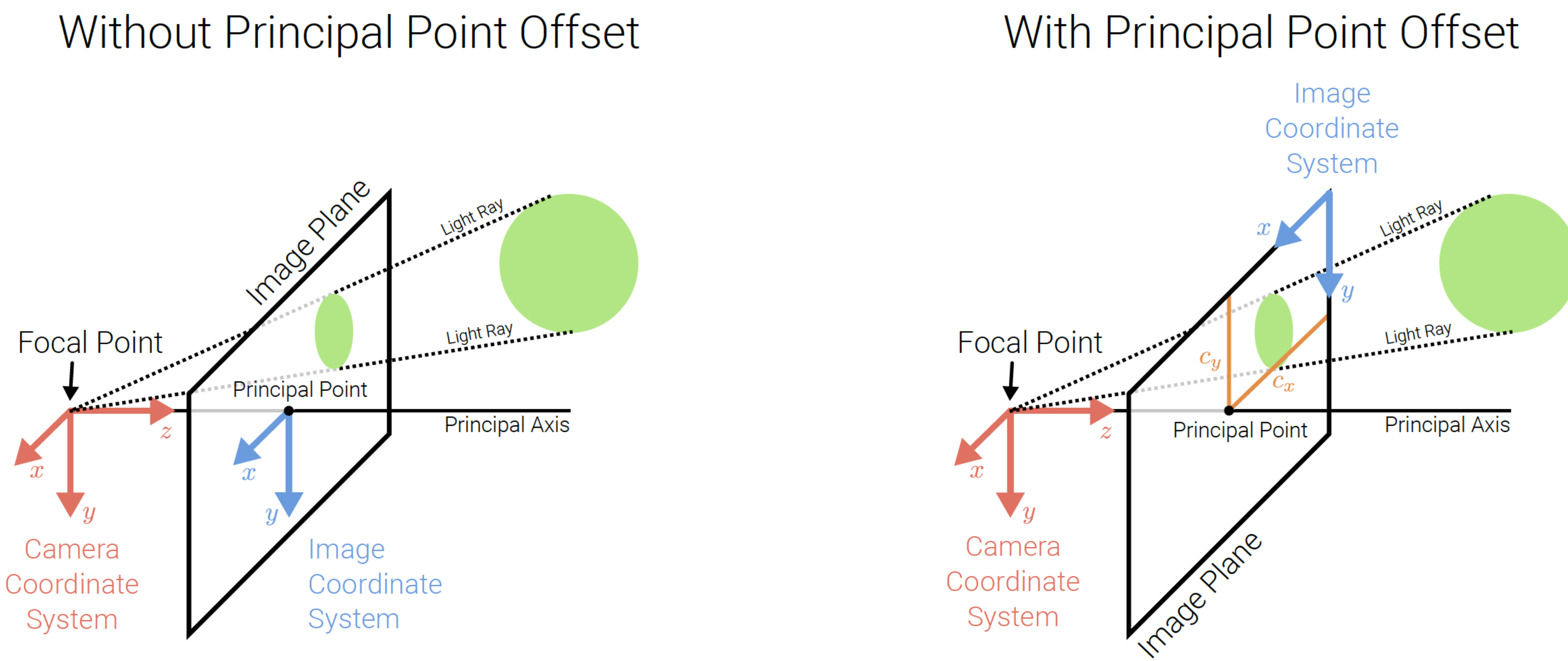
Perspective Projection

- In perspective projection, we have:

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

- 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.
- Calibration matrix

Perspective Projection



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

Perspective Projection

- The complete perspective projection model is given by:

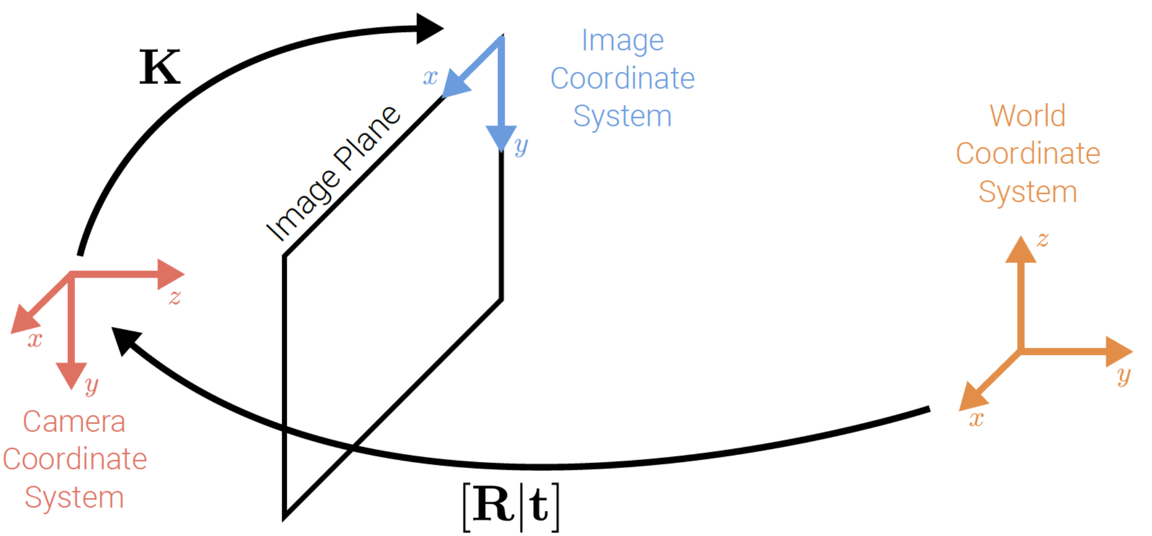
$$\begin{pmatrix} x_s \\ y_s \end{pmatrix} = \begin{pmatrix} f_x x_c/z_c + s y_c/z_c + c_x \\ f_y y_c/z_c + c_y \end{pmatrix} \Leftrightarrow \tilde{\mathbf{x}}_s = \begin{bmatrix} \boldsymbol{f_x} & \boldsymbol{s} & \boldsymbol{c_x} & 0 \\ \mathbf{0} & \boldsymbol{f_y} & \boldsymbol{c_y} & 0 \\ \mathbf{0}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

- 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 $\mathbf{c} = (c_x, c_y)^\top$

Chaining Transformation

- Let $\mathbf{K}$ be the calibration matrix (intrinsics) and $[\mathbf{R} \mid \mathbf{t}]$ the camera pose (extrinsics).
- We chain both transformations to project a point in world coordinates to the image:

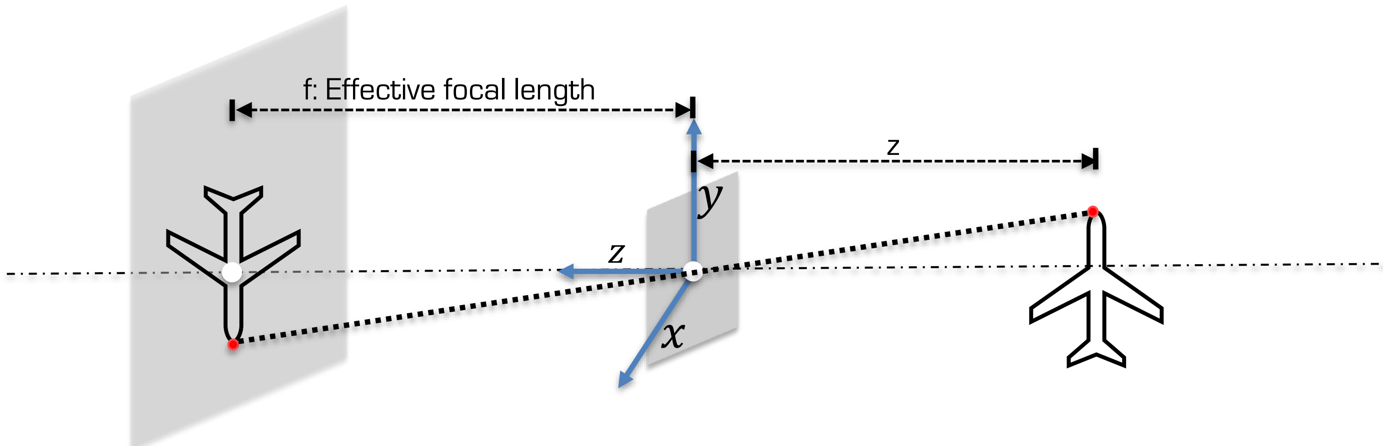
$$\tilde{\mathbf{x}}_s = [\mathbf{K} \quad \mathbf{0}]\bar{\mathbf{x}}_c = [\mathbf{K} \quad \mathbf{0}]\begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}\bar{\mathbf{x}}_w = \mathbf{K}[\mathbf{R} \quad \mathbf{t}]\bar{\mathbf{x}}_w = \mathbf{P}\bar{\mathbf{x}}_w$$



Perspective Projection

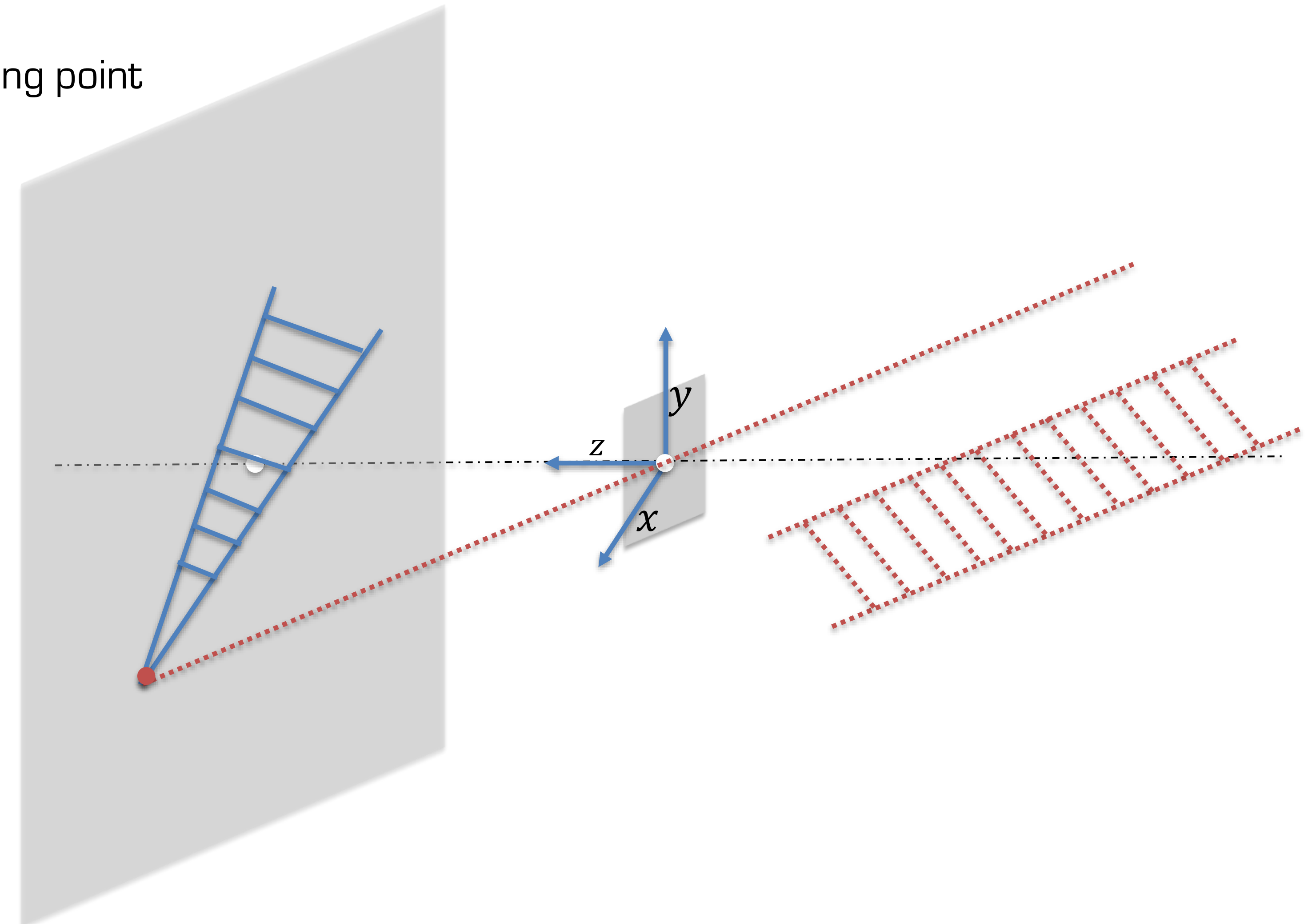
Image Magnification: $$m = \frac{f}{z}$$

- Image size inversely proportional to depth



Perspective Projection

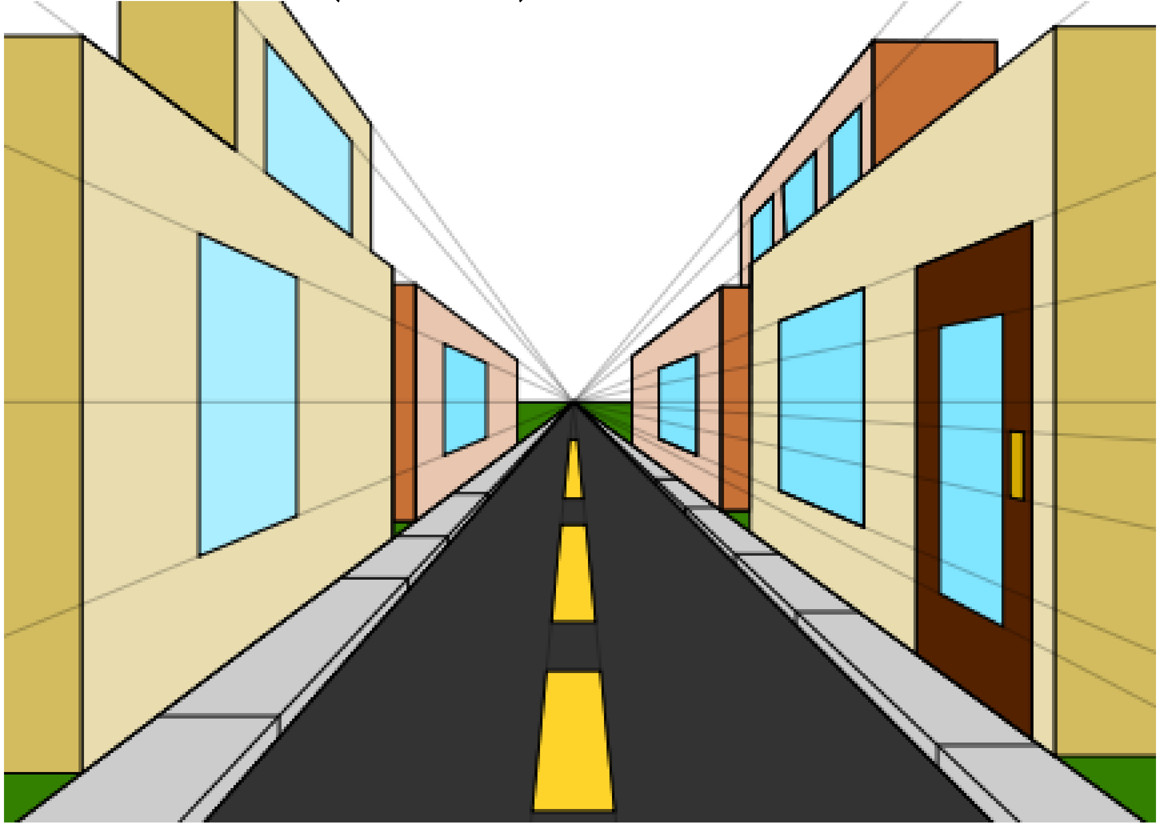
Vanishing point



Carnegie Mellon University Africa

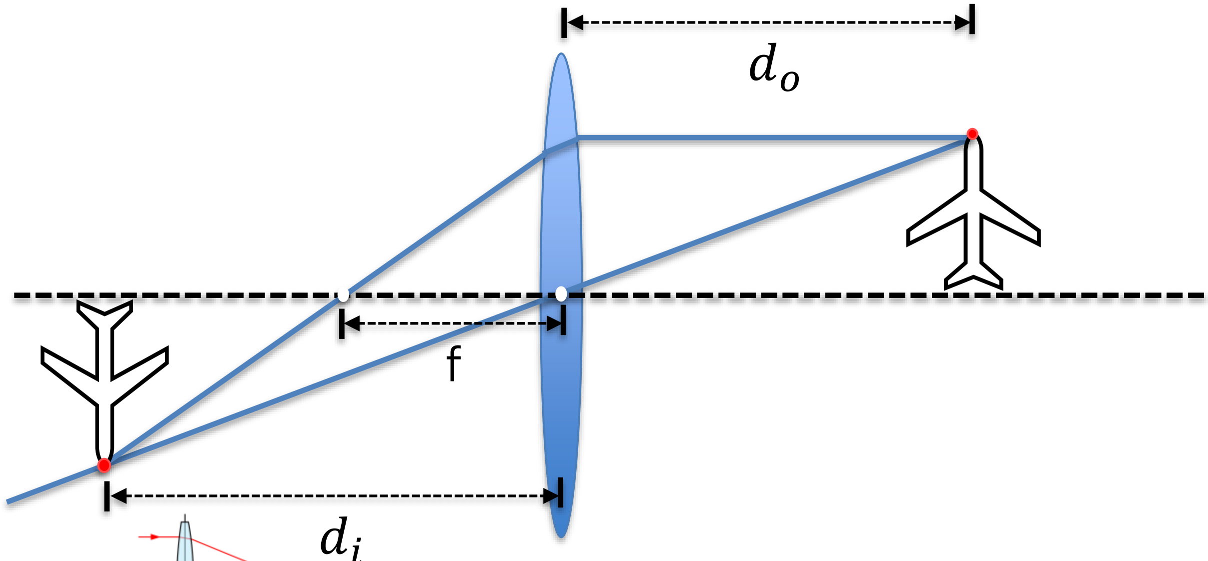
Perspective Projection

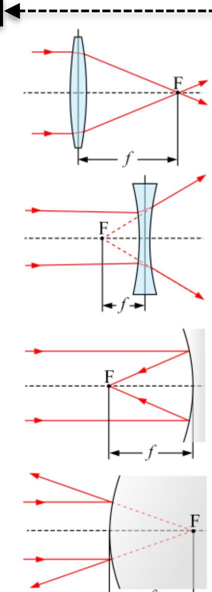
Vanishing point: $\left(x_{vp}, y_{vp}\right) = \left(f\frac{l_x}{l_z}, f\frac{l_y}{l_z}\right)$



https://commons.wikimedia.org/wiki/File:1_point_perspective.svg

Lens





Focal length (f): It determines the lens' bending power

This Lens' law: $\frac{1}{d_i} + \frac{1}{d_o} = \frac{1}{f}$

Magnifiction (m): $= \frac{h_i}{h_o} = \frac{d_i}{d_o}$

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

Lens

Aperture: Area of the lens receiving light

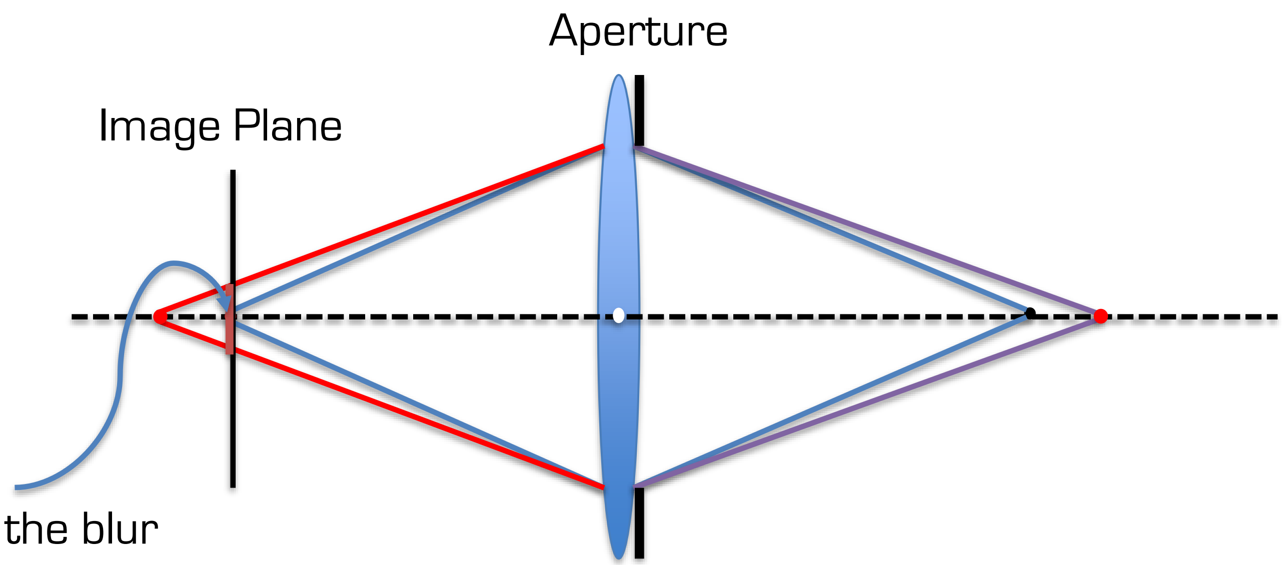
Lens

Aperture: Area of the lens receiving light

f-number: a measure of the light-gathering ability of a camera lens it equals focal length divided by the aperture.

- Aperture: $= \frac{f}{f\text{−}Number}$
- where N is called the f-Number of lens.

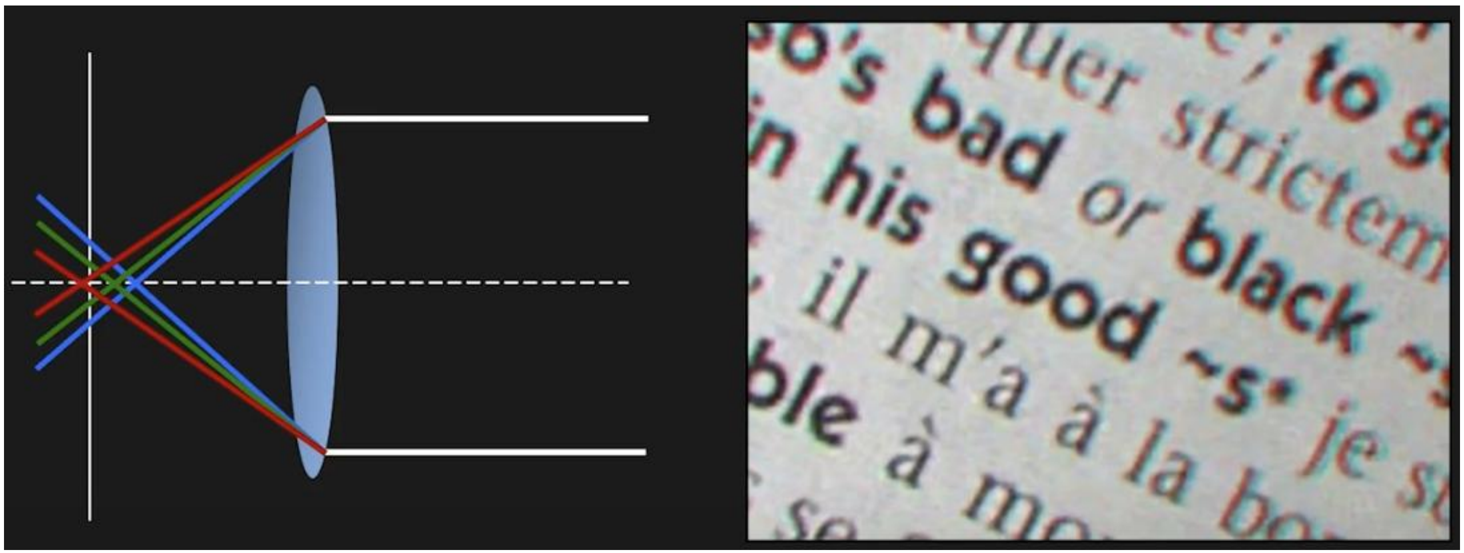
Lens Defocus



How to find the blur circle

Lens Distortion

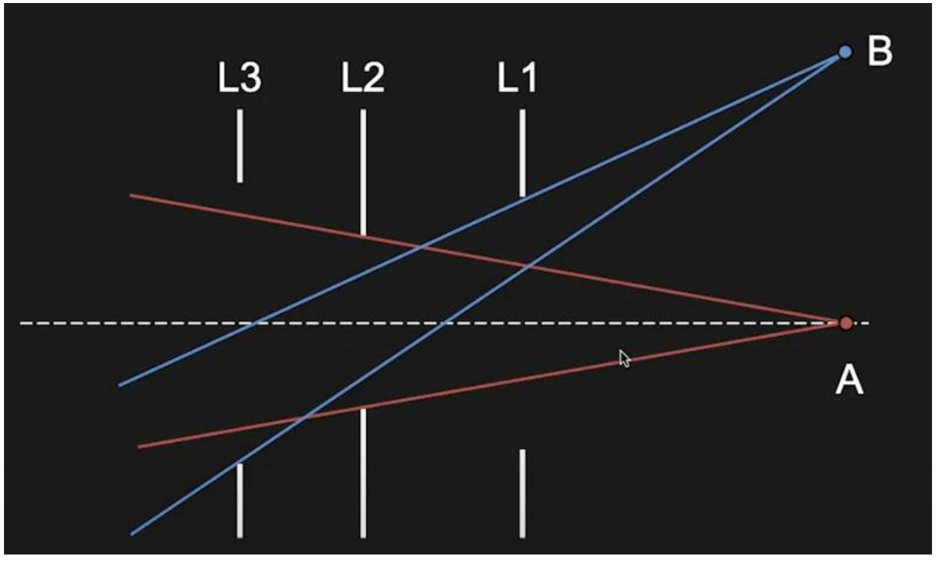
- Chromatic Aberration



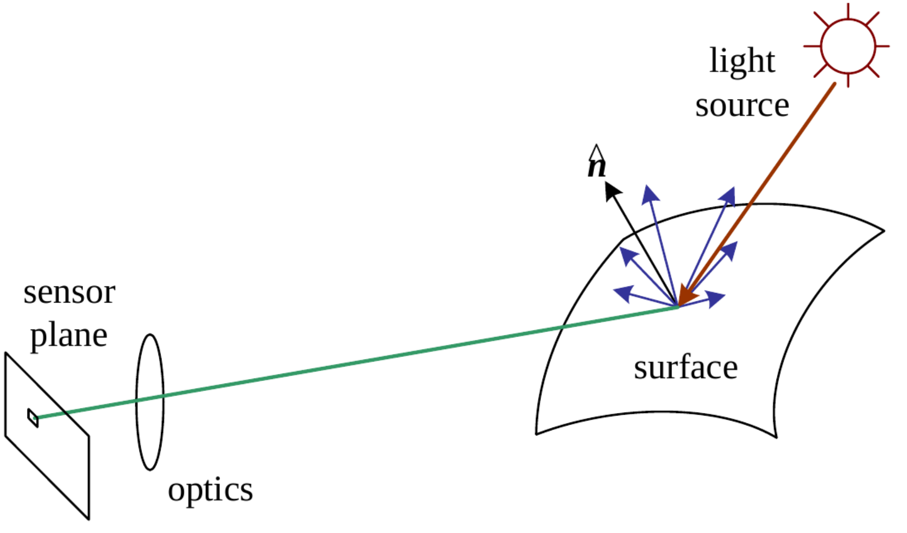
Refractive index (and hence focal length) of lens is different for different wavelengths.

Lens Distortion

- Vignetting



Photometric Image Formation



- 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

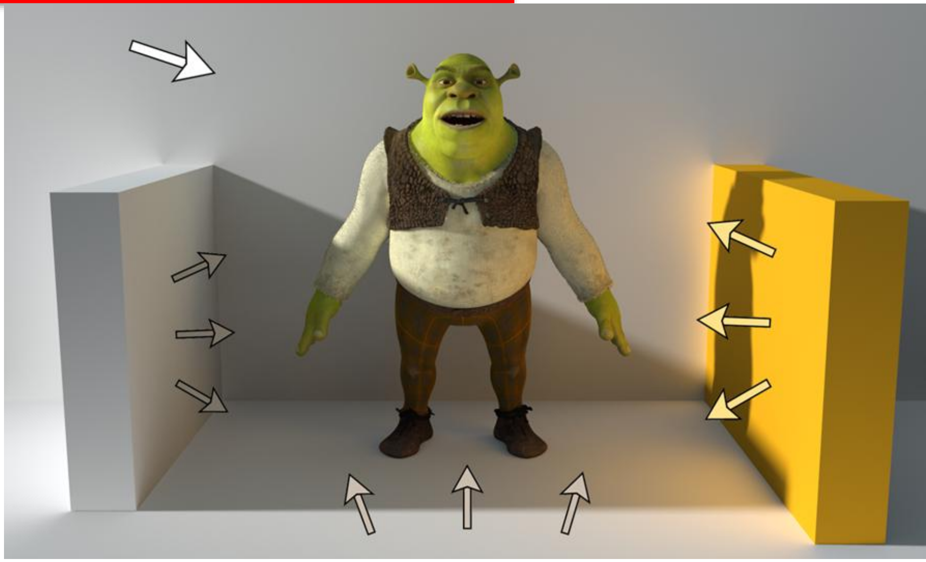
- Let $\mathbf{p} \in \mathbb{R}^3$ denote a 3D surface point, $\mathbf{v} \in \mathbb{R}^3$ the viewing direction and $\mathbf{s} \in \mathbb{R}^3$ the incoming light direction.

- The rendering equation describes how much of the light L_{in} with wavelength λ arriving at $\mathbf{p}$ is reflected into the viewing direction $\mathbf{v}$:

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

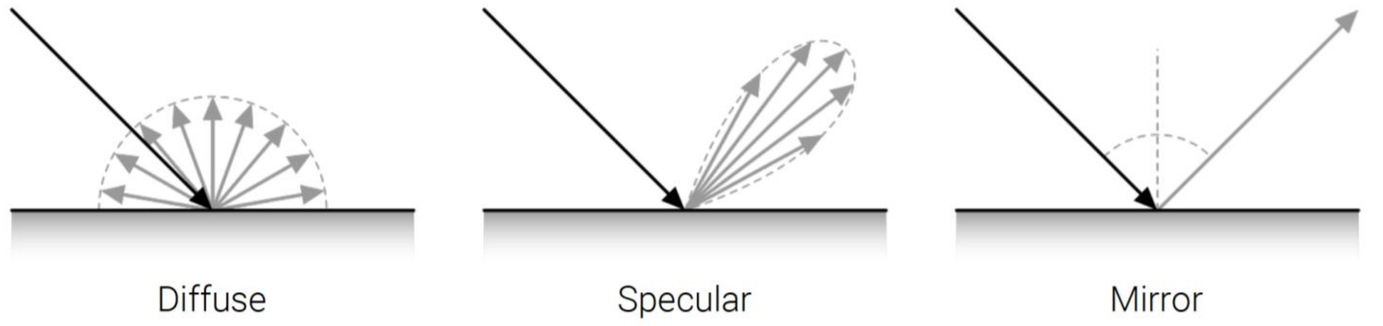
  - Ω is the unit hemisphere at normal $\mathbf{n}$
  - The bidirectional reflectance distribution function $\mathrm{BRDF}(\mathbf{p}, \mathbf{s}, \mathbf{v}, \lambda)$ defines how light is reflected at an opaque surface.
  - $L_{\text{emit}} > 0$ only for light emitting surfaces

Global Illumination



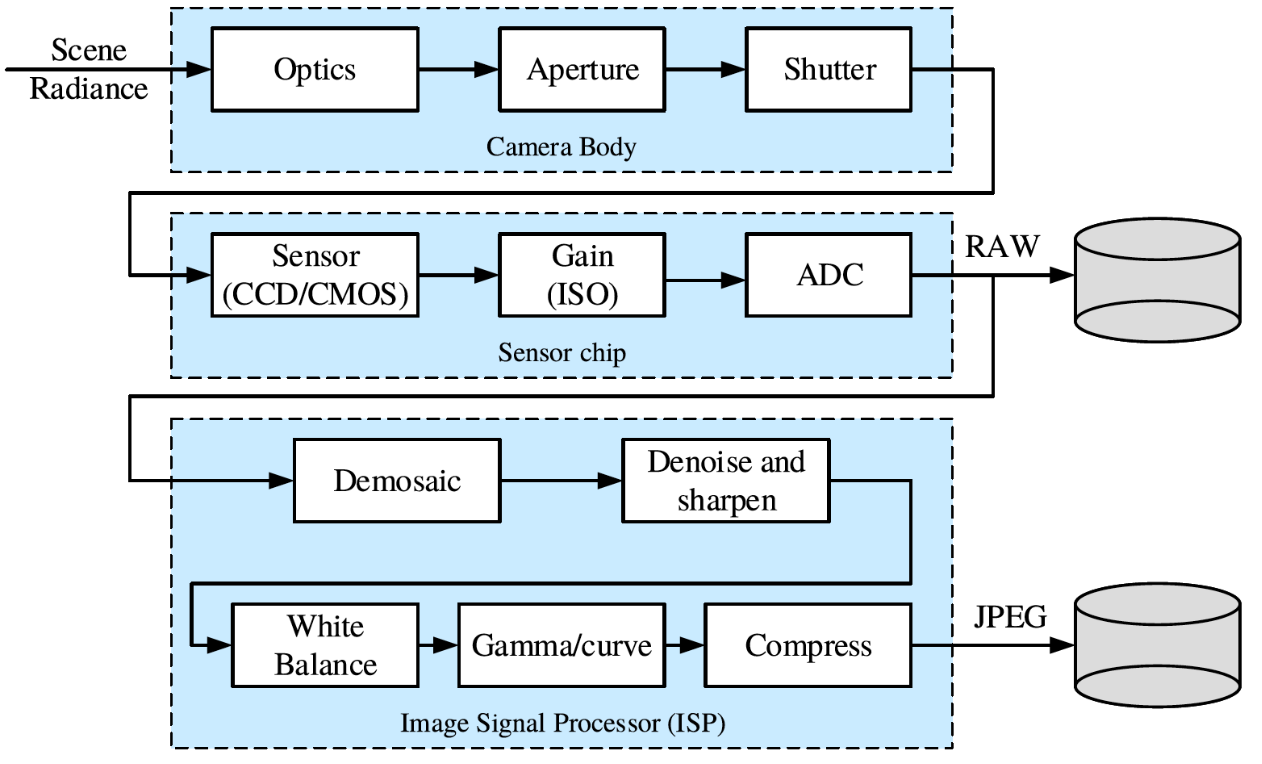
- Modeling one light bounce is insufficient for rendering complex scenes
- I Light sources can be shadowed by occluders and rays can bounce multiple times
- I Global illumination techniques also take indirect illumination into account

Diffuse and Specular Reflection



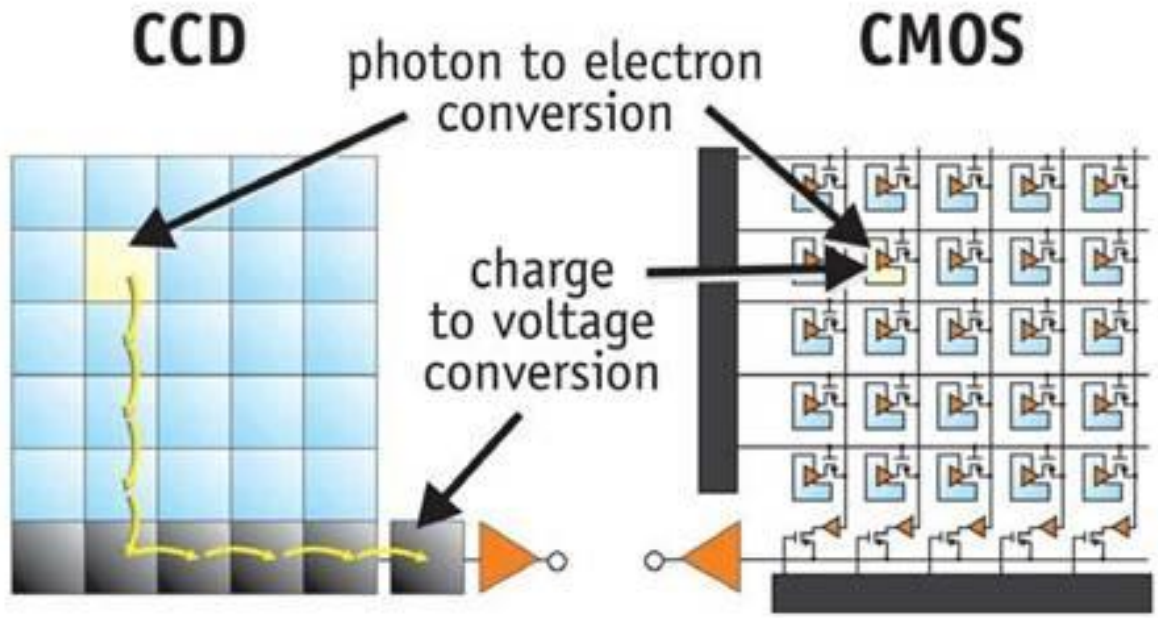
- Typical BRDFs have a **diffuse** and a **specular** component
- The diffuse (=constant) component scatters light uniformly in all directions
- This leads to shading, i.e., smooth variation of intensity wrt. surface normal
- The specular component depends strongly on the outgoing light direction

Image Sensor



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

Image Sensor



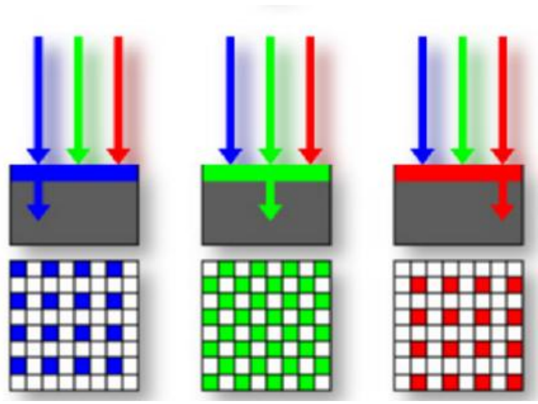
- I CCDs move charge from pixel to pixel and convert it to voltage at the output node
- I CMOS images convert charge to voltage inside each pixel and are standard

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

Carnegie Mellon University Africa

Image Sensor

- **Color Filter:**



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

Bayer RGB Pattern

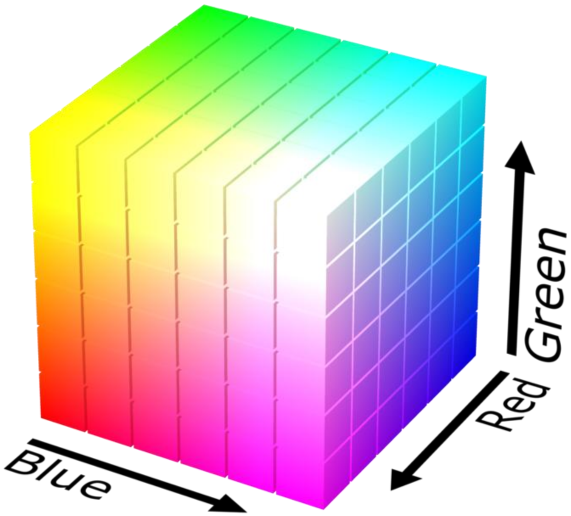
rGb	Rgb	rGb	Rgb
rgB	rGb	rgB	rGb
rGb	Rgb	rGb	Rgb
rgB	rGb	rgB	rGb

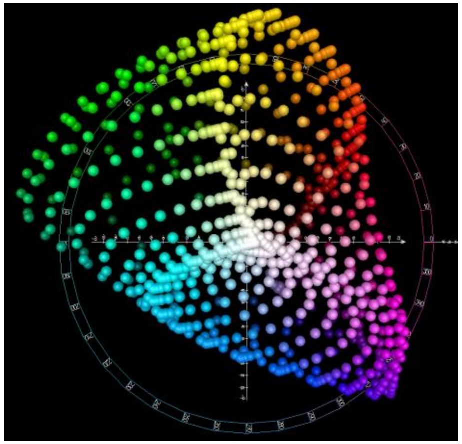
Interpolated Pixels

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

Image Sensor

- **Color Filter:**





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

Carnegie Mellon University Africa

Reference

- https://uni-tuebingen.de/fakultaeten/mathematisch-naturwissenschaftliche-fakultaet/fachbereiche/informatik/lehrstuehle/autonomous-vision/lectures/computer-vision/