電腦視覺與應用
Computer Vision and Applications

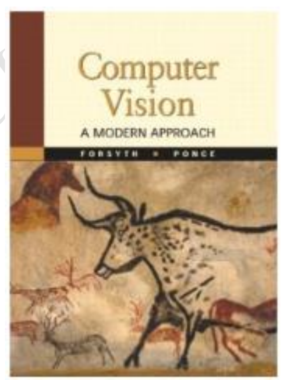
Lecture02-1-Pinhole camera

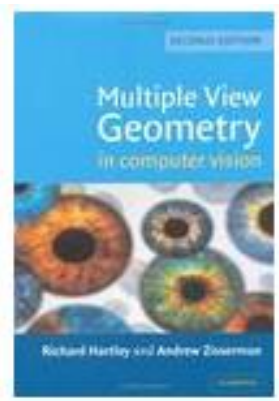
Tzung-Han Lin
National Taiwan University of Science and Technology
Graduate Institute of Color and Illumination Technology
e-mail: thl@mail.ntust.edu.tw

TAIWAN TECH National Taiwan University of Science and Technology
CIT 色彩與照明科技研究所 Graduate Institute of Color and Illumination Technology

Pinhole Camera

- Lecture reference coverage in following:
  - Computer Vision A Modern Approach, Chapter 1.
  - Multiple View Geometry in Computer Vision, Chapter 6.
  - And, miscellaneous paper & internet resource.

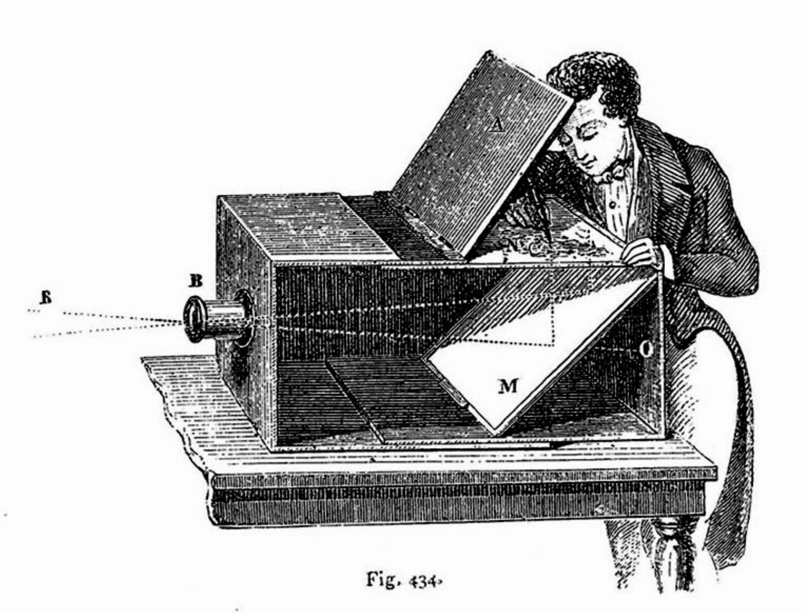




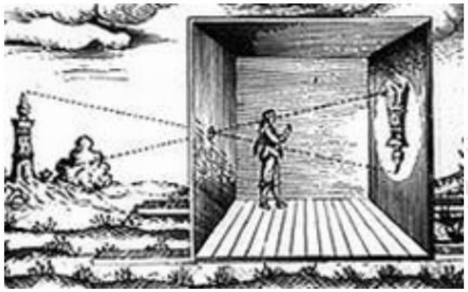
How an image formed?

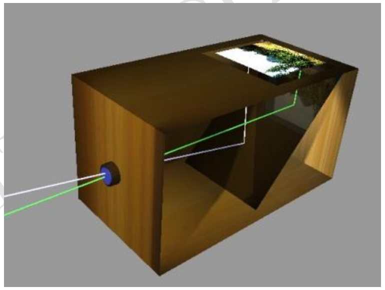
The prototype of modern cameras (Pinhole Camera) :

- In 1490, Leonardo Da Vinci gave clear descriptions of darkened chamber in his notebooks.
- However, in 1544, many of the first camera obscuras were large rooms like that illustrated by the Dutch scientist Reinerus Gemma-Frisius for use in observing a solar eclipse.



Camera Obscura, 1568

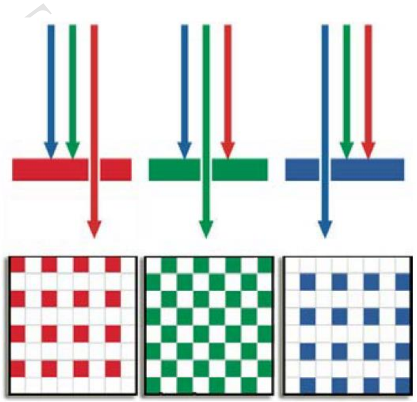


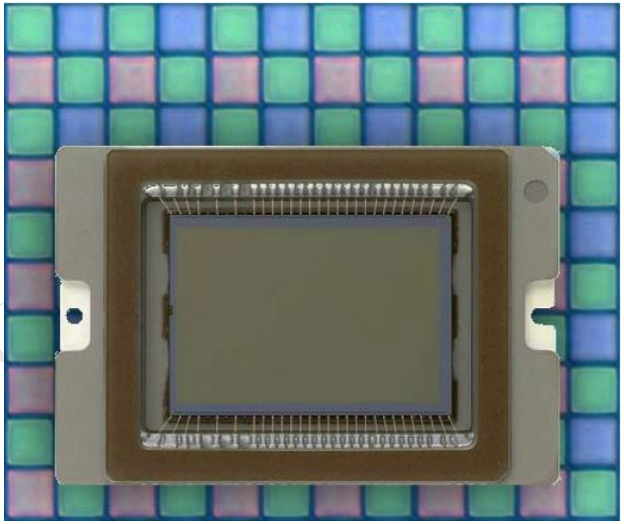


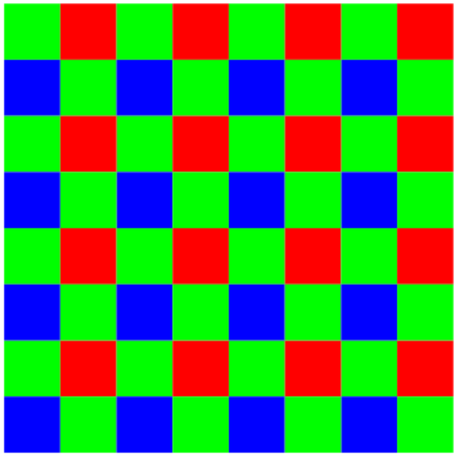
How an image formed?

Modern Digital Film:

- This figure shows general design for color CCD. NOTE: All Pixels are not created equal!
- In many applications, we consider "Grey" images only.

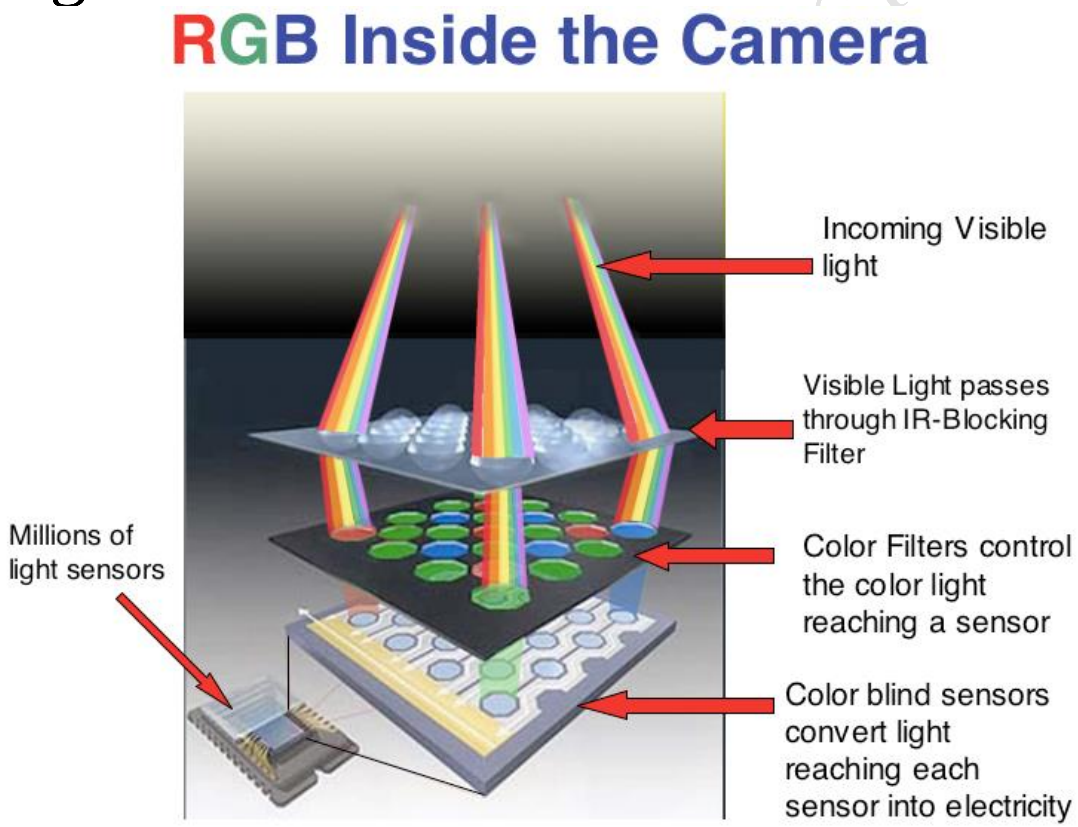






How an image formed?

- Inside digital film



http://www.photoaxe.com/wp-content/uploads/2007/04/camera-sensor.jpg

How an image formed?

How the "COLOR" image formed?

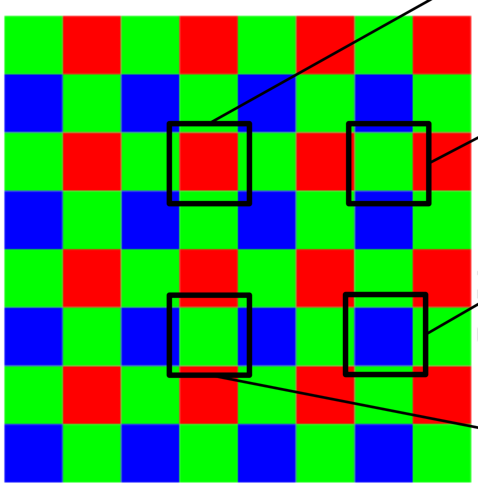
- Bitmap image uses 8 bit for R, G and B to store a true color pixel.
- A simple conversion from "grey CCD + Bayer pattern"
- 4 cases for all pixels:

red = red[x][y];
green = (green[x][y-1] + green[x-1][y] + green[x+1][y] + green[x][y+1]) / 4;
blue = (blue[x-1][y-1] + blue[x+1][y-1] + blue[x-1][y+1] + blue[x+1][y+1]) / 4;

red = (red[x-1][y] + red[x+1][y]) / 2;
green = green[x][y];
blue = (blue[x][y-1] + blue[x][y+1]) / 2;

red = (red[x-1][y-1] + red[x+1][y-1] + red[x-1][y+1] + red[x+1][y+1]) / 4;
green = (green[x][y-1] + green[x-1][y] + green[x+1][y] + green[x][y+1]) / 4;
blue = blue[x][y];

red = (red[x][y-1] + red[x][y+1]) / 2;
green = geen[x][y];
blue = (blue[x-1][y] + blue[x+1][y]) / 2;



8x8 Color CCD

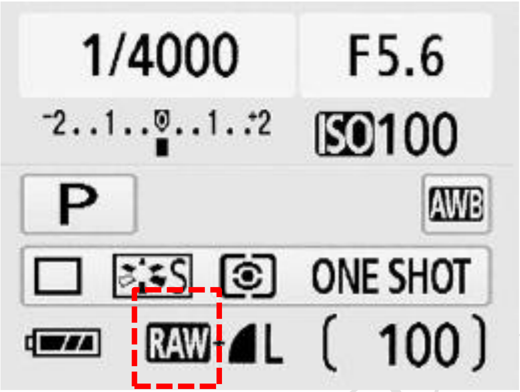
Note: This is one example. In general, CCD manufacturers will deal with "RAW data" with different algorithms to generate a final image (mostly in Jpg or Tif) for customers.

Further reading, http://www.peter-cockerell.net/Bayer/bayer.html

What does Color-bit stand for?

Color bit in image Raw data:

- Some CCD or Camera manufacturers would provide SDK for reading RAW data out.
- RAW data is the output from each of the original red, green and blue sensitive pixels of the image sensor, after being read out of the array by the array electronics and passing through an analog to digital converter.



Example: Canon DSLR

記錄系統		
記錄格式	相機檔案系統設計規格DCF 2.0	
影像類型	JPEG及RAW(14位元, Canon原創)	
RAW+JPEG同時記錄	有	
檔案大小	(1) 大/精細:	約1790萬像素 (518

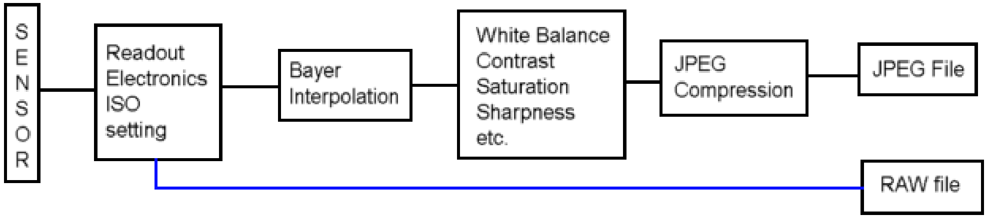
Raw data provides higher resolution than 8-bit on each channel (says R, G, B).

8 bit→256 grey levels

10 bit→ 1,024 grey levels

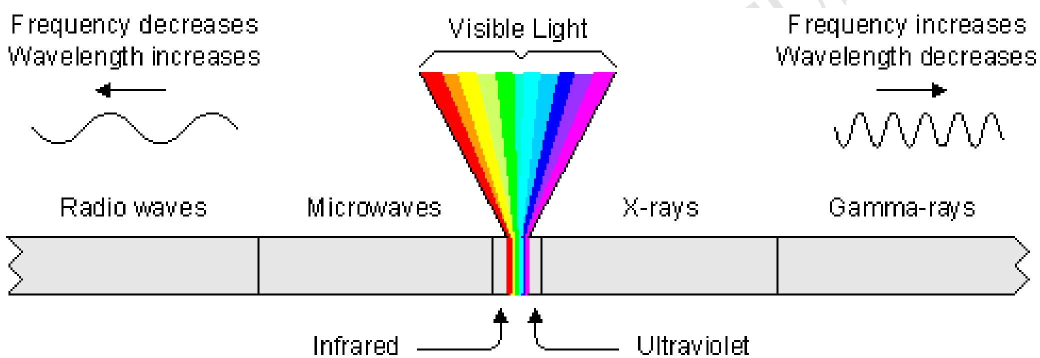
12 bit→ 4,096 grey levels

14 bit→ 16,384 grey levels

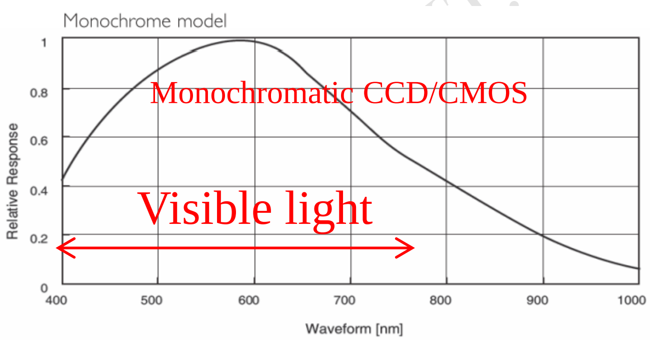


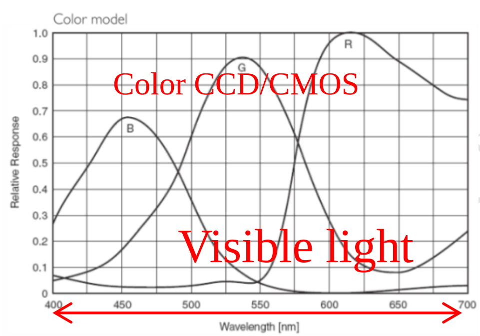
What does Color-bit stand for?

- Visible spectrum: 380nm~780nm



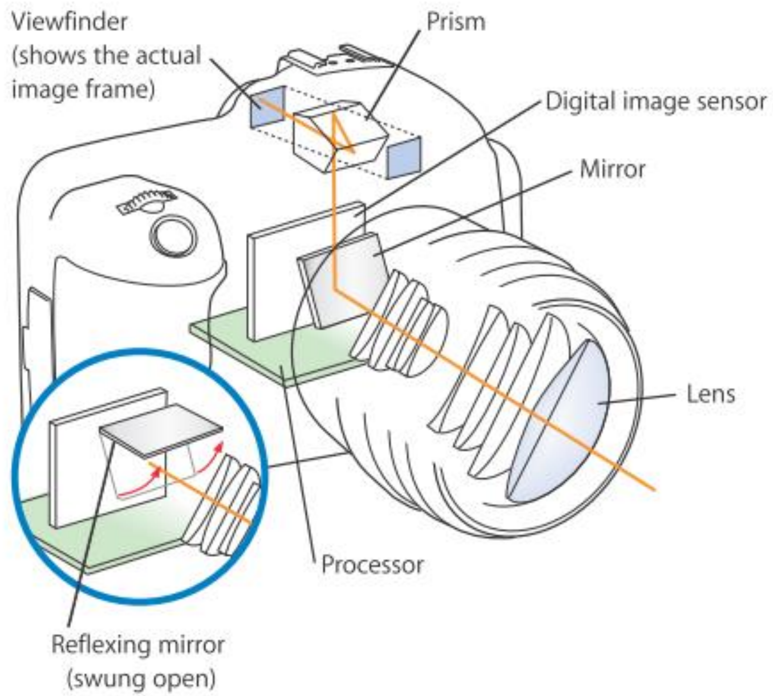
The visible portion of the electromagnetic spectrum

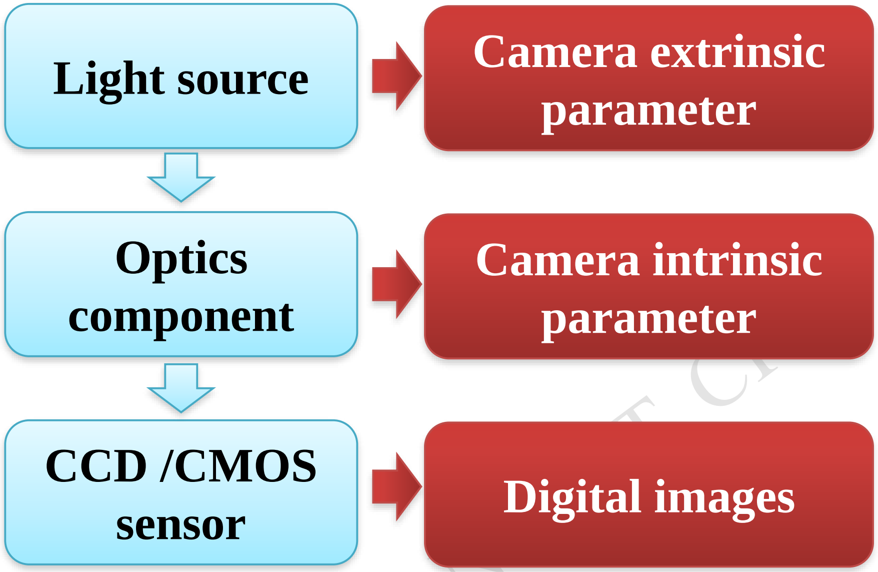




What does Color-bit stand for?

- DSLR: Digital Single Lens flex.





Picture from Aperture Digital Photography Fundamentals

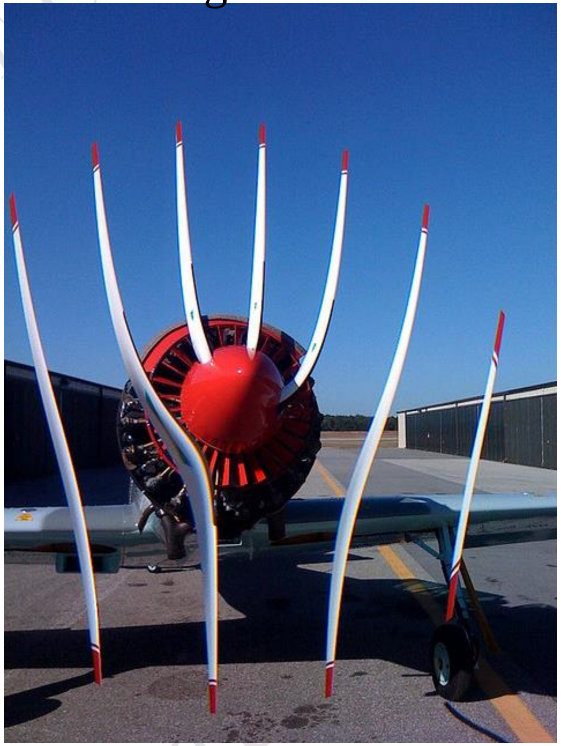
What is rolling-shutter effect?

Rolling shutter problem:

In computer vision field, all images are formed at the same instance, means, scenes in all images are assumed to be static at the moment of shooting pictures.

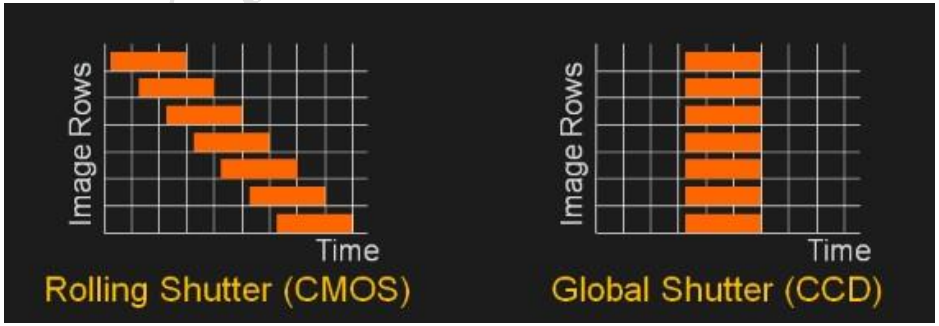
If you are using one camera with "rolling shutter", the distortions of all images caused by motion should be compensated as possible. In general case, increasing shutter speed of camera is the another way to suppress this effect.

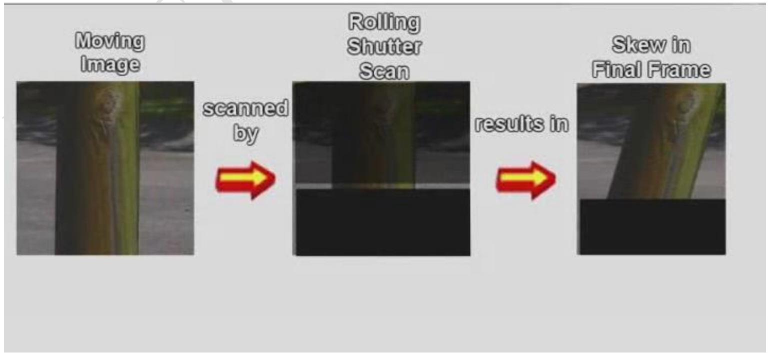
Rolling shutter issue



http://www.flickr.com/photos/sorenragsdale/3192314056/in/pool-1485036@N20

What is rolling-shutter effect?





http://www.cs.columbia.edu/CAVE/projects/crsp/images/crsp.png
http://www.stemmer-imaging.co.uk/images/glossary/Rolling-Shutter-I0.png
http://dvxuser.com/jason/CMOS-CCD/

Camera model

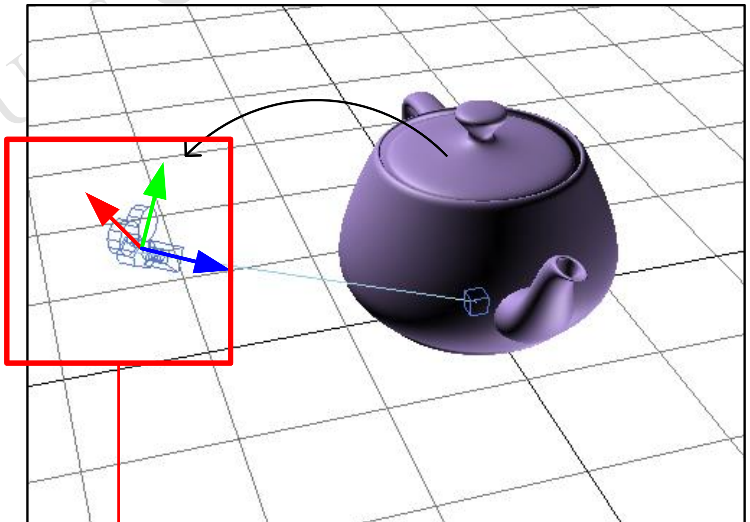
The mathematical description for the camera in 3D space is formed by "Extrinsic" and "Intrinsic" parameters:

1.**Extrinsic parameter** (外部參數)→defines the relation between camera and environment in Euclidean space. (a 3x4 matrix)→以3x4矩陣描述相機的位置(position)與擺置方向(orientation)。

2.**Intrinsic parameter** (內部參數)→defines how the light goes through lens (a 3x3 matrix), and finally induces the brightness on exact 2D coordinate of the image →以3x3矩陣描述光通過鏡頭投射到影像的座標位置。

Camera model

Description for extrinsic



After intrinsic operation

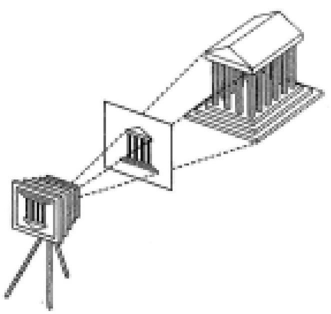
相機的位置描述式(矩陣形式)
=相機外部參數
(以相機的位置重新描述定義世界座標的座標點)
*不涉及觀看物體結果，只有涉及與物體相對關係

透過相機觀看
=相機內部參數
(觀看世界座標的座標點)
*涉及觀看物體

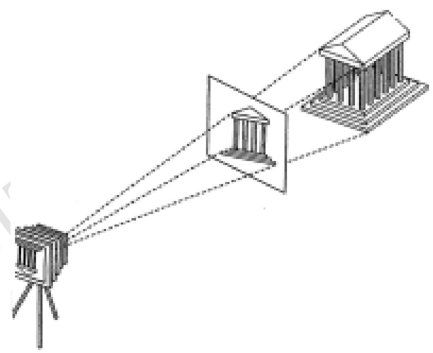
Camera model

- **General projection method for pinhole camera: "Perspective"**

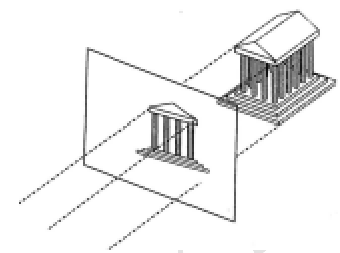
- This is a "Computer graphics" model, and similar to intrinsic parameter.



Perspective Projection
透視投影轉換運算



Weak-perspective Projection
透視投影轉換運算



Parallel Projection
平行投影轉換運算

Picture from book: Interactive Computer Graphics

Camera model

Requirement for forming a image (from geometrical viewpoint)

1. For Graphics model (or ideal pinhole)

- Extrinsic parameter
- Intrinsic parameter → perspective projection (in most cases, otherwise orthographical projection)

2. For real Cameras

- Extrinsic parameter
- Intrinsic parameter → perspective (with camera calibration matrix)
- Lens distortion distribution

homogenous 2D point on image

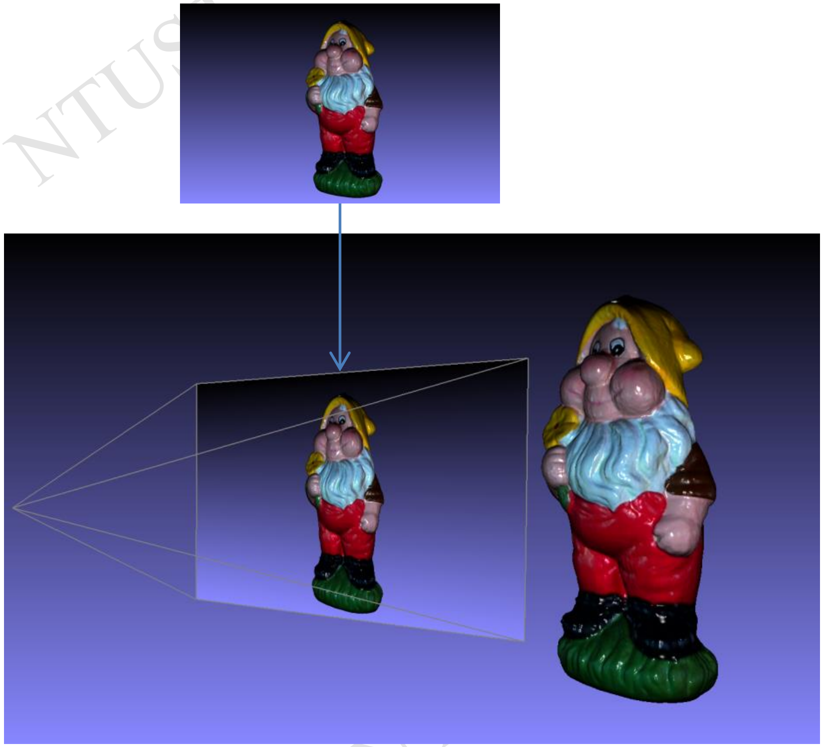
General form: $\mathbf{x} = \mathbf{K}[\mathbf{R} \mid \mathbf{t}]\mathbf{X}_{3D}$

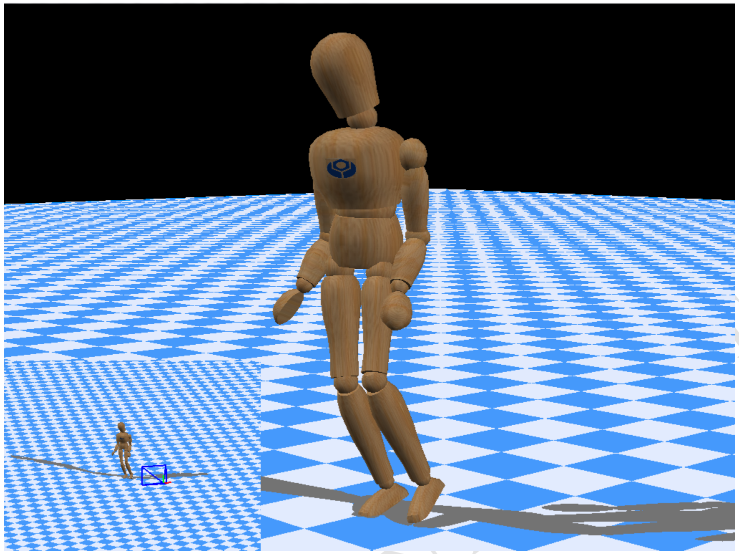
3D point in Euclidean space (says world coordinate)

Camera intrinsic parameter

Camera transformation in this Euclidean space (inverse to camera position and orientation)

Camera model: Example

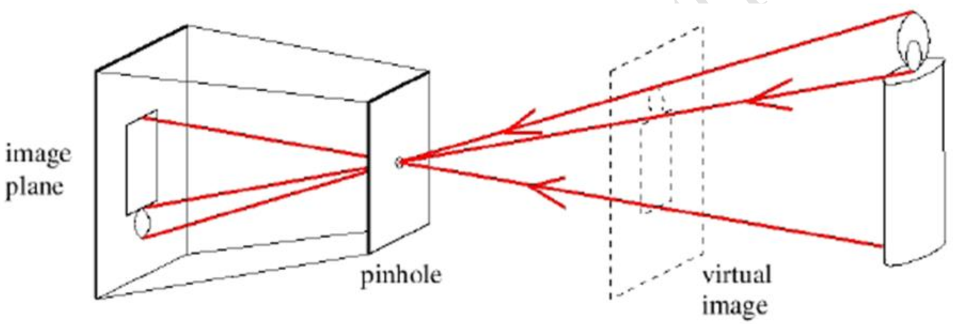


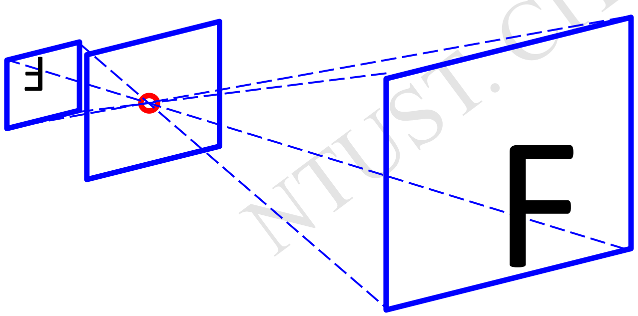


Camera model

Behaviors of the pinhole camera model

- The object image on the image plane will be upside down, but NOT "mirror".
- The pinhole perspective projection (also call central perspective) was proposed by Brunelleschi.

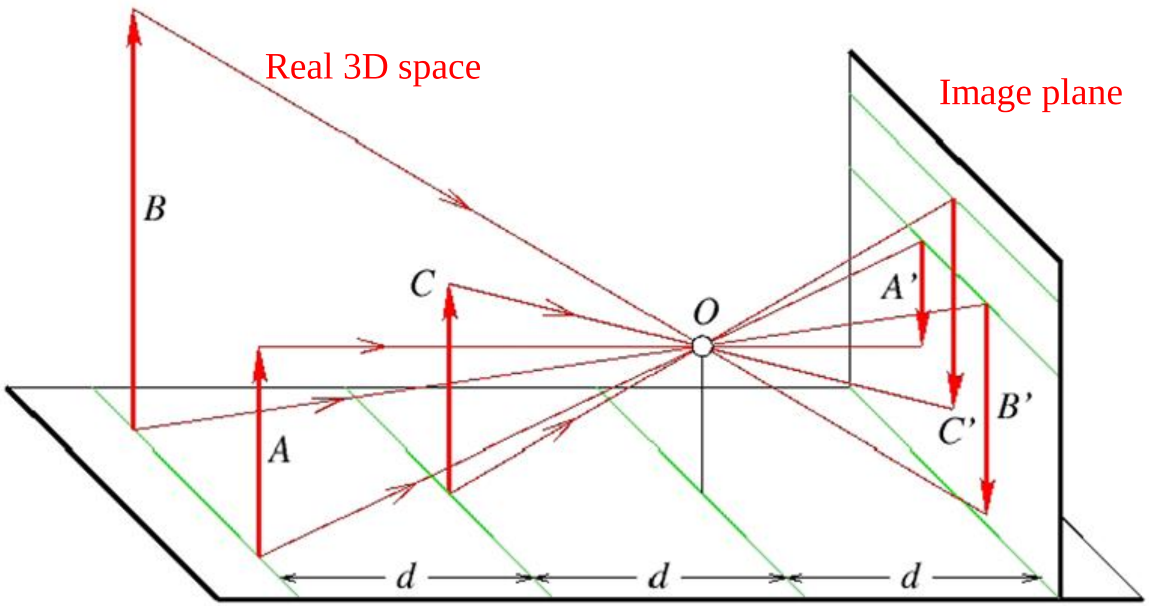




Camera model

Behaviors of the pinhole camera model –cont.

- Far objects appear smaller then close ones.



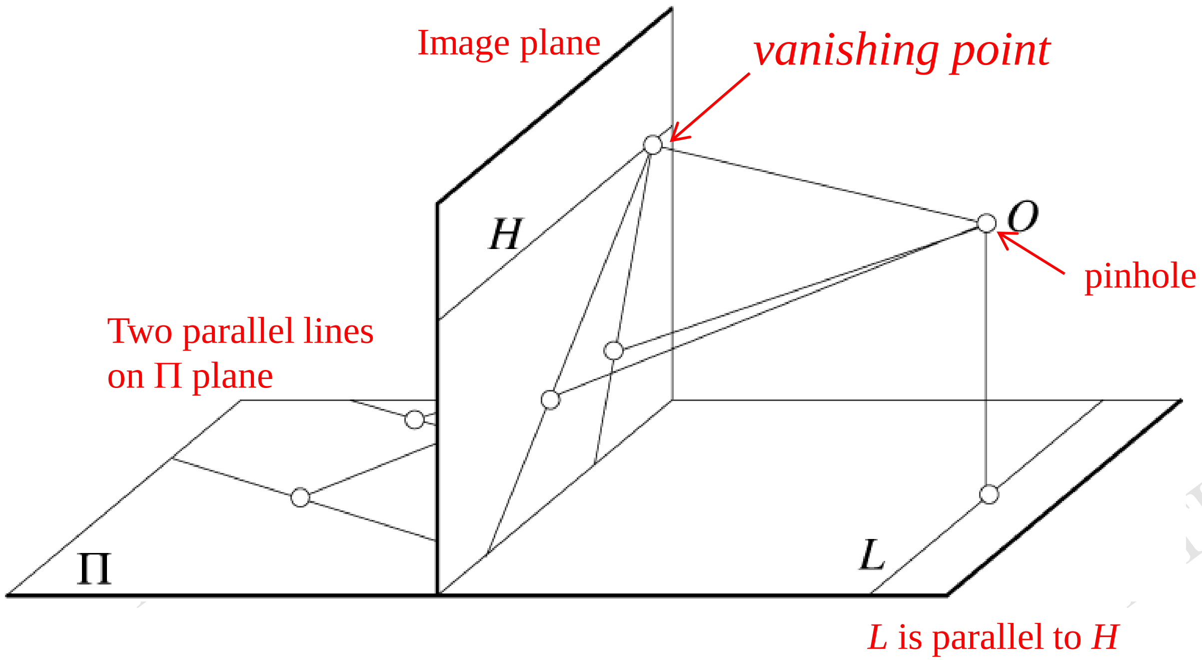
A and *C* are half the size of *B*
(in real 3D space)

C' & *B*' : same size
(in image plane)

Camera model

Behaviors of the pinhole camera model –cont.

- Distant objects are smaller : the vanishing point

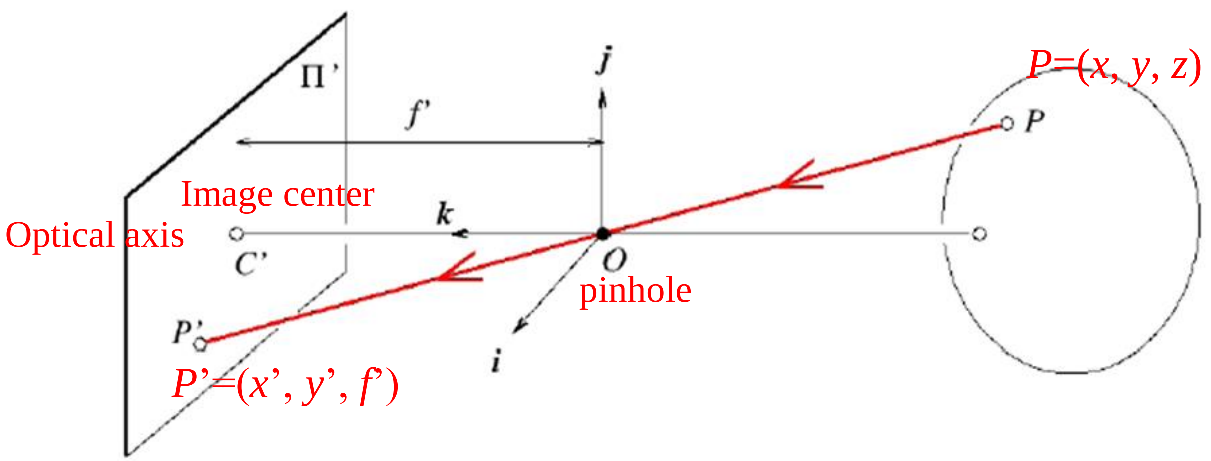


Camera model

Behaviors of the pinhole camera model –cont.

Π ' Plane is perpendicular to ***k*** axis.

- $P'=(x', y', f')$ → on image plane
- $P=(x, y, z)$ → in 3D space

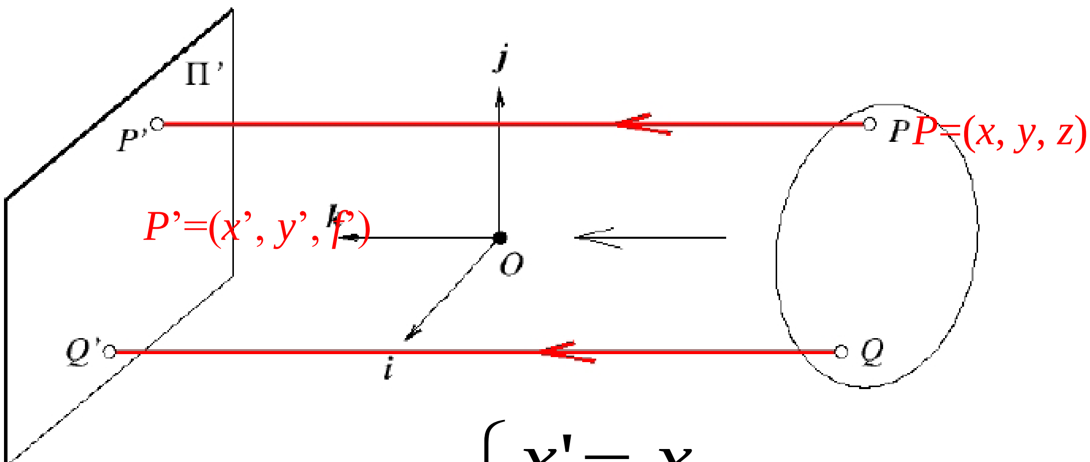


$$\begin{cases} x' = \lambda x \\ y' = \lambda y \\ f' = \lambda z \end{cases} \qquad x' = f'\frac{x}{z} \qquad y' = f'\frac{y}{z}$$

Camera model

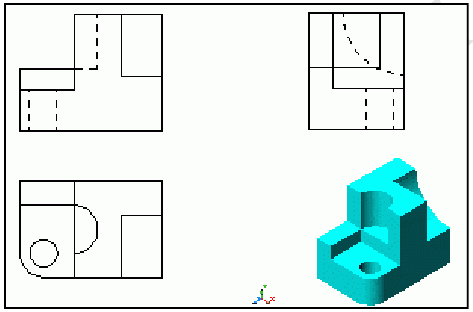
Another projection method :

- **orthographic** projection (so-called **parallel** projection), usually for engineering purposes



$$\begin{cases} x' = x \\ y' = y \end{cases}$$

Example: better visualization for dimension measurement

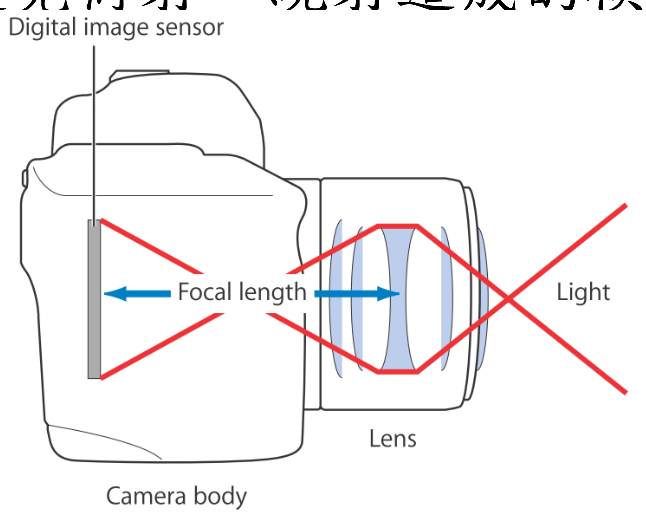


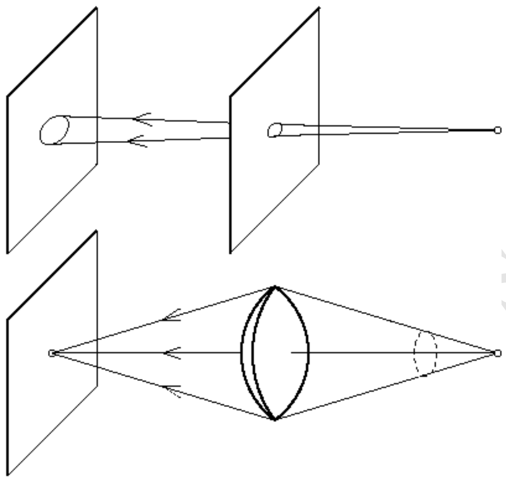
Camera model

Why cameras with lenses？ Two Reasons:

1.To gather light since a single ray of light would otherwise reach each point in the image plane. (增加進光亮、聚焦)

2.To keep the picture in sharp focus.→ to avoid diffraction effect. (避免衍射、繞射造成的模糊)



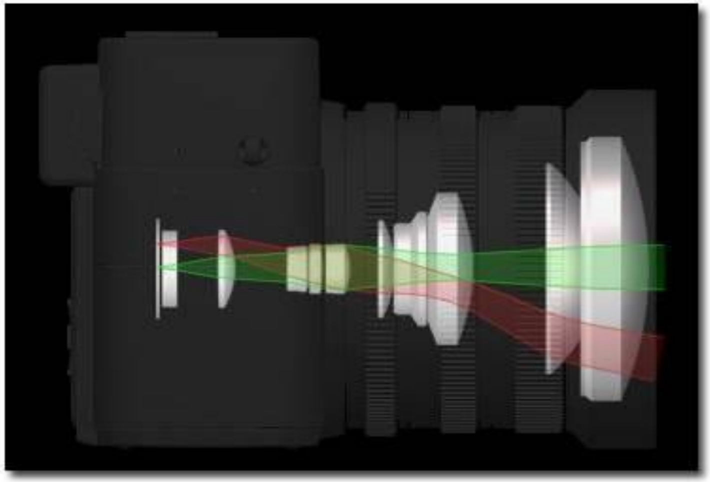


Picture from Aperture Digital Photography Fundamentals

Camera model

Lens systems in a real-camera

- A good camera lens may contain 15 elements and cost a thousand dollars
- The best modern lenses may contain aspherical elements (非球面元件)



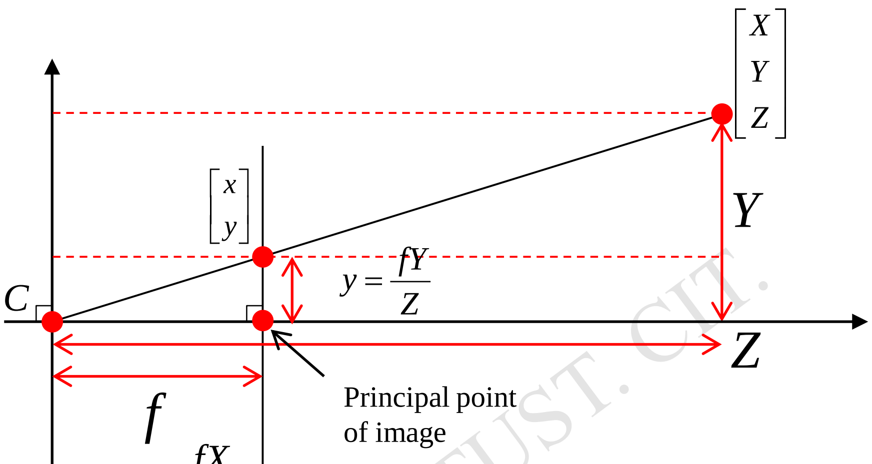
In modern computer vision textbook, all of the lens behaviors are described as a 3x3 matrix (intrinsic parameter for mapping 3D to 2D) and a polynomial function for lens distortion.

Picture from David Lowe

Camera model

Ideal case (mathematic model)

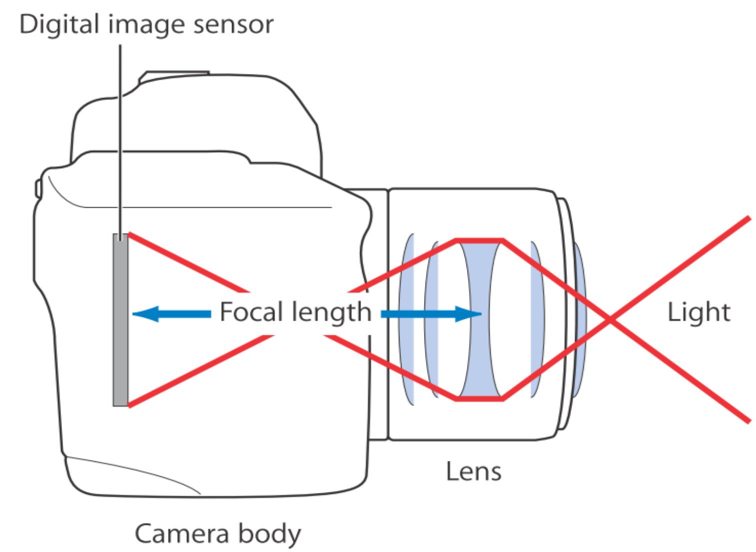
- **Intrinsic parameter** governs the geometry for the ideal camera model, i.e. mathematic eqs.



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

$$y = \frac{fY}{Z}$$

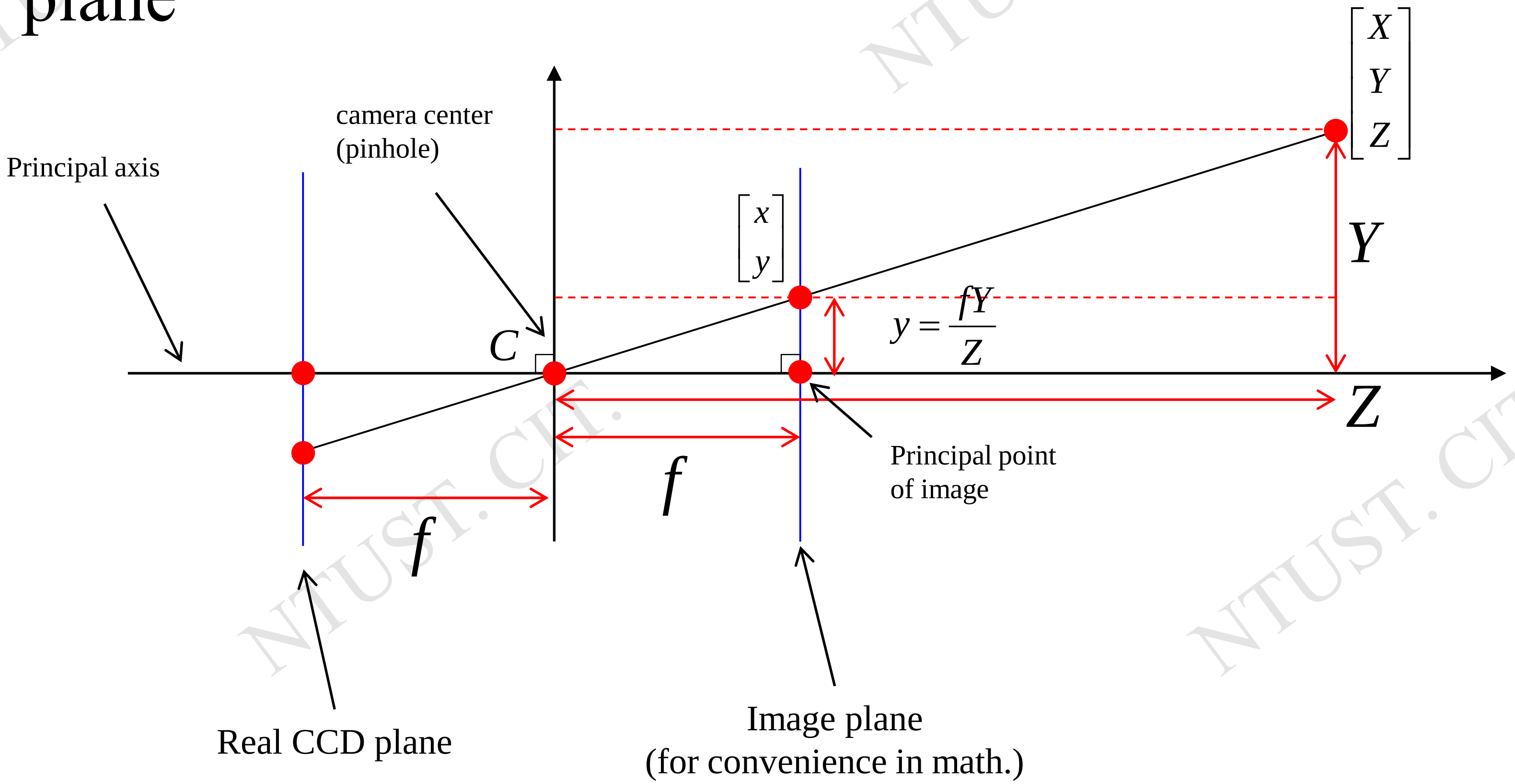
$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$



Note: lens and projection are rewritten as one 3x3 matrix.

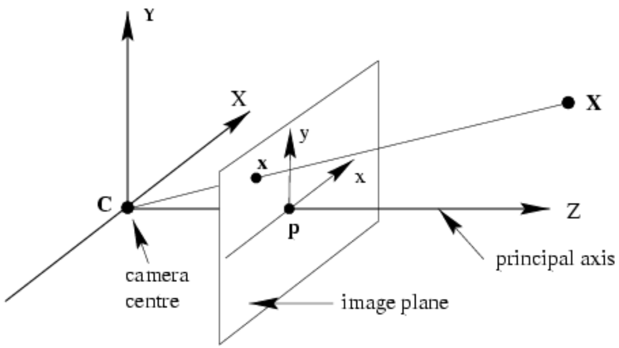
Camera model

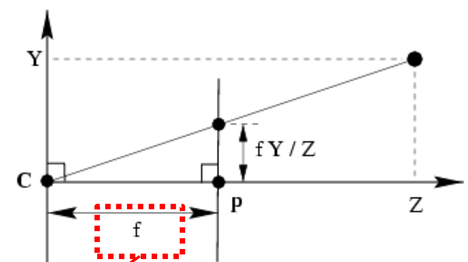
- Between “real CCD” and “mathematical image-plane ”



Camera model

- Intrinsic parameter





$$\begin{pmatrix}X\\Y\\Z\\1\end{pmatrix}\mapsto\begin{pmatrix}fX\\fY\\Z\end{pmatrix}=\begin{bmatrix}f&&&0\\&f&&0\\&&1&0\end{bmatrix}\begin{pmatrix}X\\Y\\Z\\1\end{pmatrix}$$

homogenous 2D point on image plane.

camera projection matrix (one kind of intrinsic parameter)

3D points (relative to camera coordinate)

Camera model

- Extrinsic parameter

$$\begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \Rightarrow \boxed{\begin{pmatrix} fX \\ fY \\ Z \end{pmatrix}} = \begin{bmatrix} f & & \\ & f & \\ & & 1 \end{bmatrix} \underbrace{\boxed{\begin{bmatrix} 1 & & & 0 \\ & 1 & & 0 \\ & & 1 & 0 \end{bmatrix}}}_{[\mathrm{I}\,|\,0]} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

homogenous 2D point on image plane.

$$\mathbf{x} = diag(f, f, 1)[\mathrm{I}\,|\,0]\boxed{\mathbf{X}_{\mathrm{cam}}}$$

3D points relative to Camera coordinates

Camera model

In practice, we hope to get 2D points as the coordinates on one image, then, the image should be shifted.

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \mapsto \begin{pmatrix} fX/Z + p_x \\ fY/Z + p_y \\ 1 \end{pmatrix}$$

image center
(or called principal point)

rewrite as:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{bmatrix} f & & p_x & 0 \\ & f & p_y & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

In simple:

$$\mathbf{x} = \mathbf{K}[\mathrm{I} \mid 0]\mathbf{X}_{\mathrm{cam}}$$

here, $\mathbf{K} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}$

3D points relative to
Camera coordinates

Camera model

Intrinsic parameter (short summary)

$$\mathbf{x} = \mathbf{K}[\mathrm{I}\,|\,0]\mathbf{X}_{\mathrm{cam}}$$

K denotes the intrinsic parameter of the camera.

Perspective projection
(central projection, with offset)

$$\mathbf{K} = \begin{bmatrix} f & \boxed{0} & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}$$

General case for REAL camera

Perspective projection
(Finite projective camera)

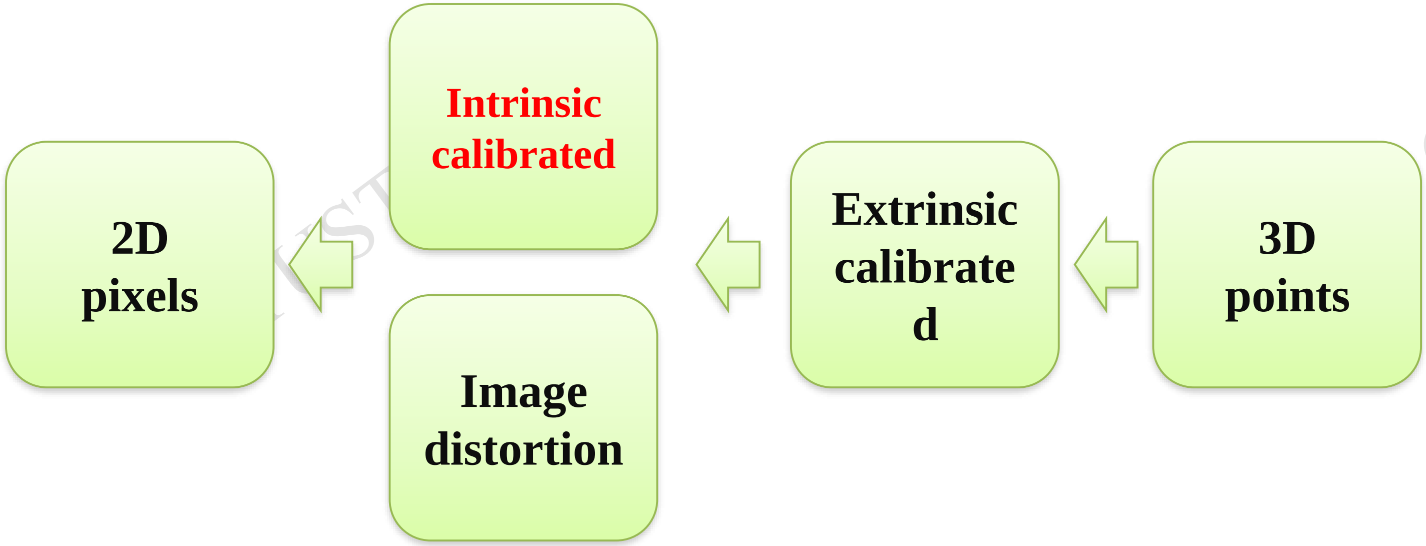
$$\mathbf{K} = \begin{bmatrix} f_x & \boxed{\gamma} & x_c \\ 0 & f_y & y_c \\ 0 & 0 & 1 \end{bmatrix}$$

f_x **:focal length in** x **direction**
f_y **:focal length in** y **direction**
γ **: skew**
(x_c, y_c) **: principal point**

Camera model

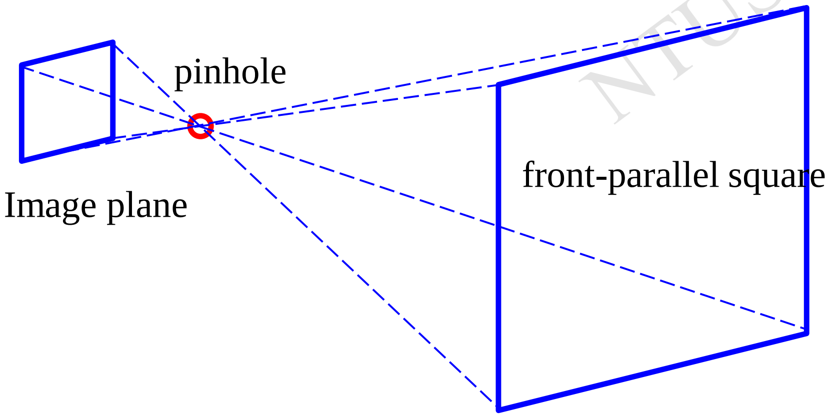
A application scenario (3D projection):

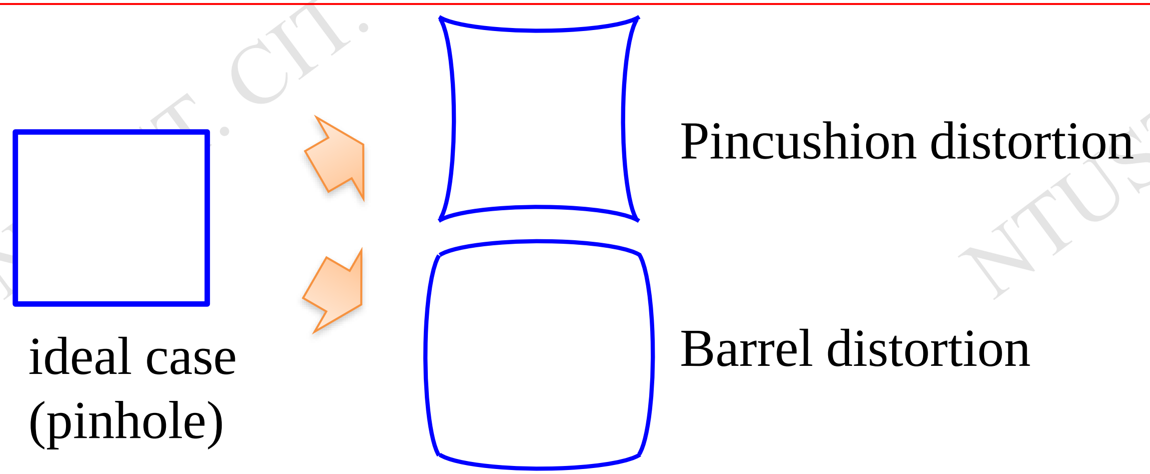
- Input (known): 3D point coordinates, Camera extrinsic & intrinsic parameter
- Output (unknown) : to determine 2D coordinates of after projecting 3D points on to one image.



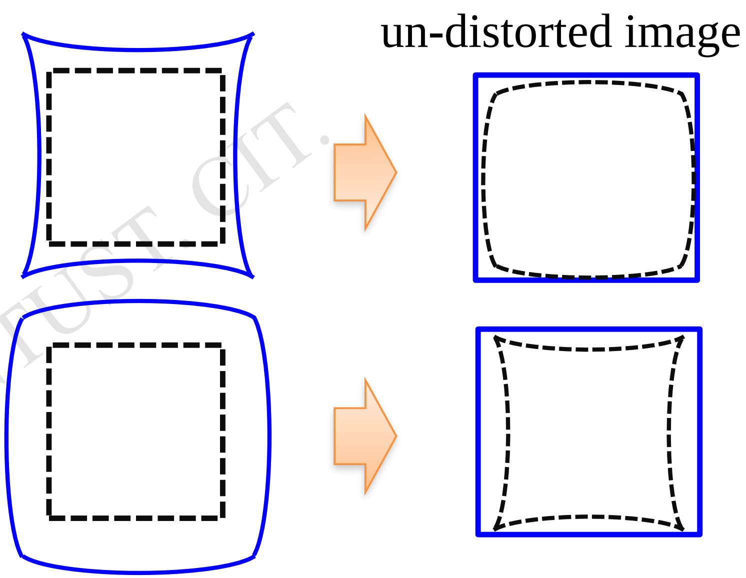
Camera model

- What's the difference between real & ideal case?





Camera model



- How the image un-distorted?

$$\mathbf{x}_s = \begin{bmatrix} X/Z \\ Y/Z \end{bmatrix} = \begin{bmatrix} x_s \\ y_s \end{bmatrix}$$

normalized 2D points
on a "virtual" image (1 unit far from camera center)

3D points relative to
camera coordinate

define

$$r^2 = x_s^{\ 2} + y_s^{\ 2}$$

Detail at Camera Calibration Toolbox for Matlab, strongly recommended!!!
http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/parameters.html

Camera model

- How the image un-distorted—cont.?

$$\mathbf{x}_d = \boxed{\begin{bmatrix} x_d \\ y_d \end{bmatrix}} = (1 + k_0 r^2 + k_1 r^4 + k_4 r^6) \begin{bmatrix} x_s \\ y_s \end{bmatrix} + \begin{bmatrix} 2k_2 x_s y_s + k_3 (r^2 + 2{x_s}^2) \\ k_2 (r^2 + 2{y_s}^2) + 2k_3 x_s y_s \end{bmatrix}$$

After un-distortion, we get new 2D coordinate → which is normalized and close to "linear" perspective model.

$$\mathbf{x}_p = \mathbf{K}\mathbf{x}_d = \begin{bmatrix} f_x & \gamma & x_c \\ 0 & f_y & y_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix}$$

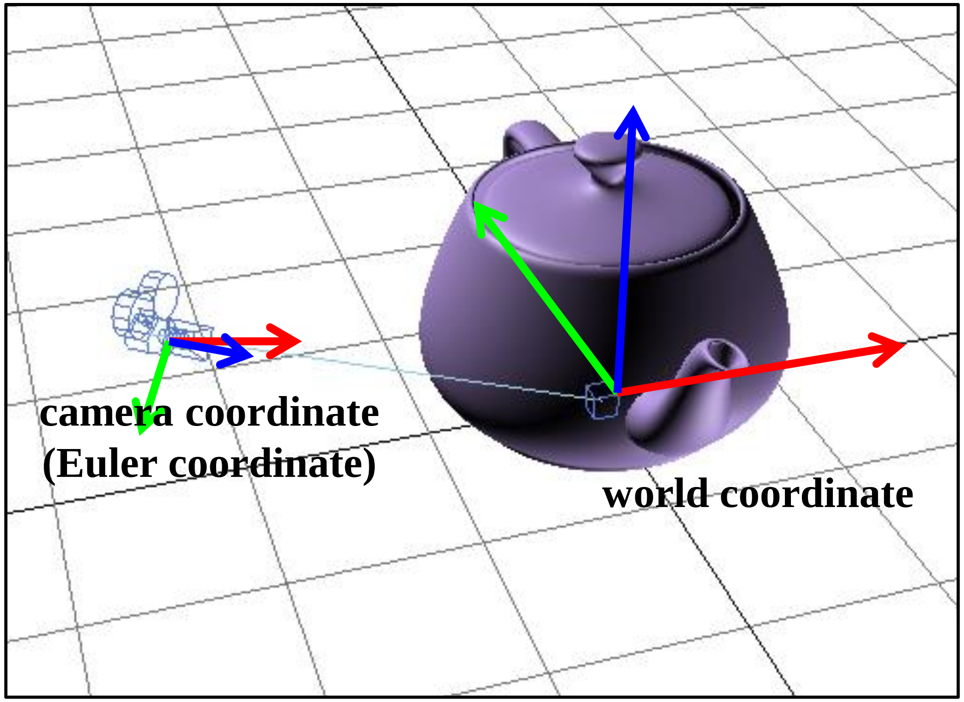
Detail at Camera Calibration Toolbox for Matlab, strongly recommended!!!
http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/parameters.html

Camera model

General description for extrinsic parameter:

To transfer the 3D points in world coordinate into another 3D points in camera coordinate.

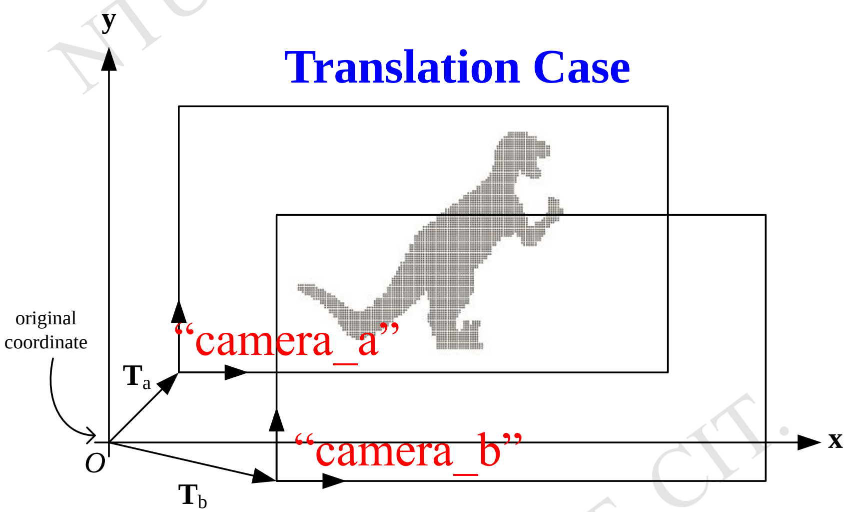
But, how to transform between them?



Camera model

For example, **Extrinsic** parameter in 2D:

Translation Case

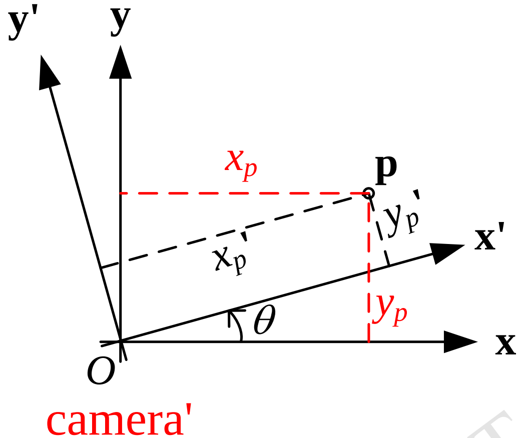


Data in camera_a

$$\mathbf{X}_a = \mathbf{T}_a^{-1}\mathbf{X}_{world}$$

or
$$\begin{bmatrix} x_a \\ y_a \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & T_{ax} \\ 0 & 1 & T_{ay} \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} x_{world} \\ y_{world} \\ 1 \end{bmatrix}$$

Rotation Case



Data in camera'

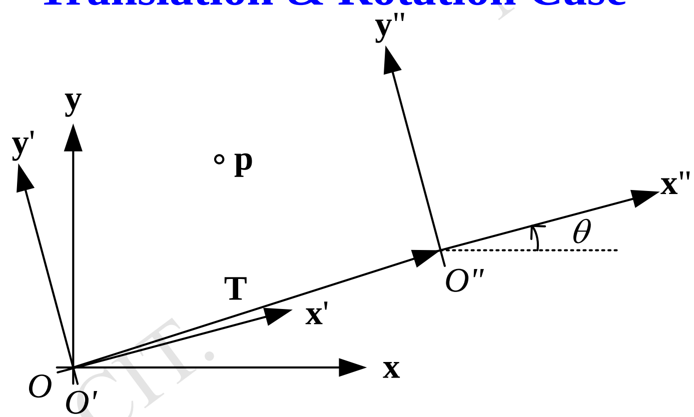
$$\mathbf{X}' = \mathbf{R}_\theta^{-1}\mathbf{X}_{world}$$

or
$$\begin{bmatrix} x_p' \\ y_p' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix}$$

Camera model

For example : **Extrinsic** parameter in 2D (mixed transformation)

Translation & Rotation Case

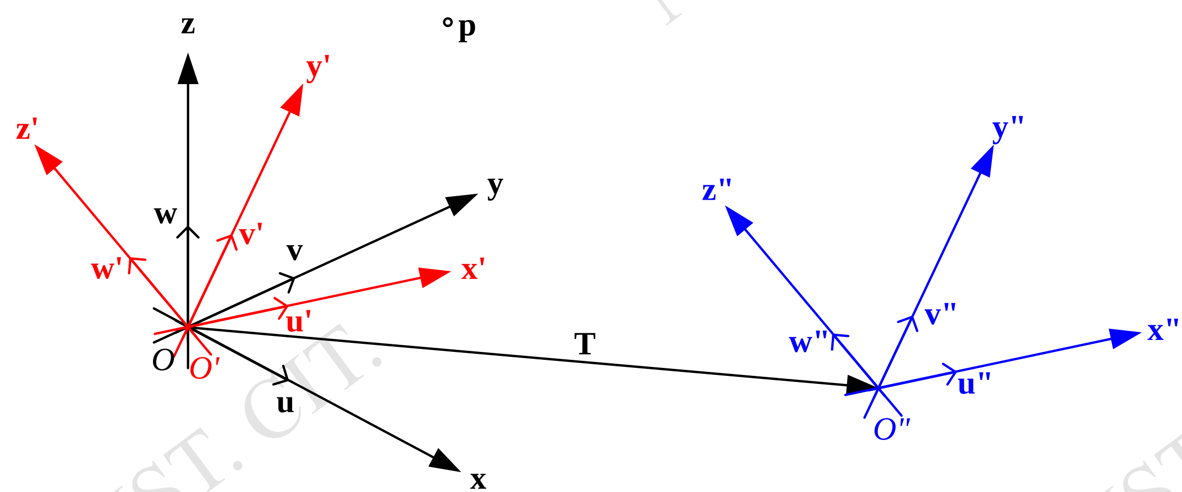


$$\mathbf{X}''=\begin{bmatrix} x_p'' \\ y_p'' \\ 1 \end{bmatrix}=\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1}\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}^{-1}\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix}=\begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}^{-1}\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix}$$

summary: $\mathbf{X}''=\begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}^{-1}\mathbf{X}_{world}$

Camera model

For example : **Extrinsic** parameter in 3D (mixed transformation)



$$\mathbf{X}''=\begin{bmatrix} u_x' & v_x' & w_x' & 0 \\ u_y' & v_y' & w_y' & 0 \\ u_z' & v_z' & w_z' & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \cdot \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \cdot \mathbf{X}_{world} = \begin{bmatrix} u_x' & v_x' & w_x' & t_x \\ u_y' & v_y' & w_y' & t_y \\ u_z' & v_z' & w_z' & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \cdot \mathbf{X}_{world}$$

Note: Here, u''=u', v''=v' and w''=w', but they indicate different coordinates (says start from *O*' or *O*'')

Camera model

Summary remark:

3D point relative to **Camera Coordinate**

3D point relative to **World Coordinate**

extrinsic parameter→ $$\mathbf{X}_{cam} = \begin{bmatrix} \mathbf{R} \mid \mathbf{t} \\ 0 \quad 1 \end{bmatrix} \mathbf{X}_{world}$$

(hint: transfer world coordinate to camera coordinate, one 3D space → another 3D space)

intrinsic parameter→ $$\mathbf{x}_{img} = \mathbf{K}[\mathrm{I} \mid 0]\mathbf{X}_{cam}$$

(hint: mapping 3D points to 2D image, one 3D space relative to camera → one 2D space)

Camera model

Summary remark—cont.:

homogenous 2D point on image (unit: pixel)

General form: $\mathbf{x}_{\text{img}} = \mathbf{K}[\mathbf{R} \,|\, \mathbf{t}]\mathbf{X}_{\text{world}}$

3D point in Euclidean space (says world coordinate)

Camera intrinsic parameter

Camera transformation in this Euclidean space, says extrinsic parameter (inverse to camera position and orientation)

Camera model

Comparison the coordinate transformation (extrinsic parameter) between "Computer Graphics" and "Computer Vision"

note: this is an **inverse** operator

Computer graphics textbook says:

$$\mathbf{X}_{\mathrm{cam}} = \underbrace{\begin{bmatrix} u_x' & v_x' & w_x' & t_x \\ u_y' & v_y' & w_y' & t_y \\ u_z' & v_z' & w_z' & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1}}_{} \cdot \mathbf{X}_{\mathrm{world}}$$

4x4 matrix
(4th row is dummy)

Computer vision textbook says:

3x4 matrix

$$\mathbf{X}_{\mathrm{cam}} = \begin{bmatrix} \boxed{\mathbf{R} \mid \mathbf{t}} \\ 0 \quad 1 \end{bmatrix} \mathbf{X}_{\mathrm{world}}$$

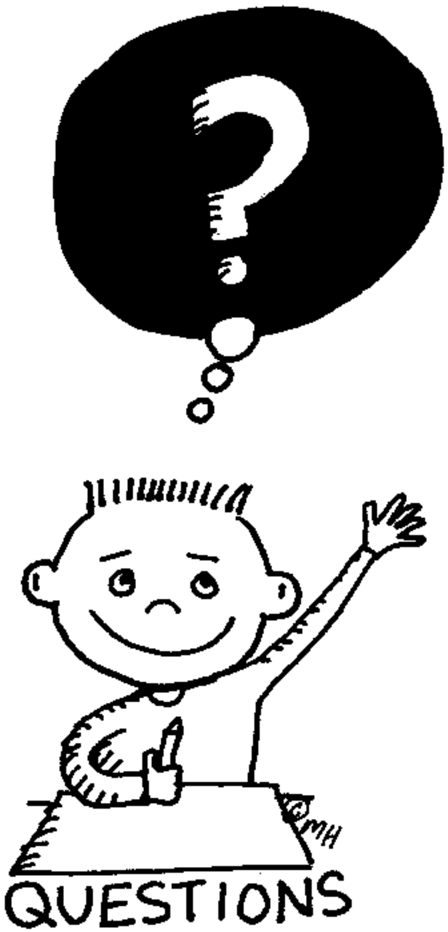


Figure from : http://www.gabankruptcylawyersnetwork.com/files/2014/08/question.gif