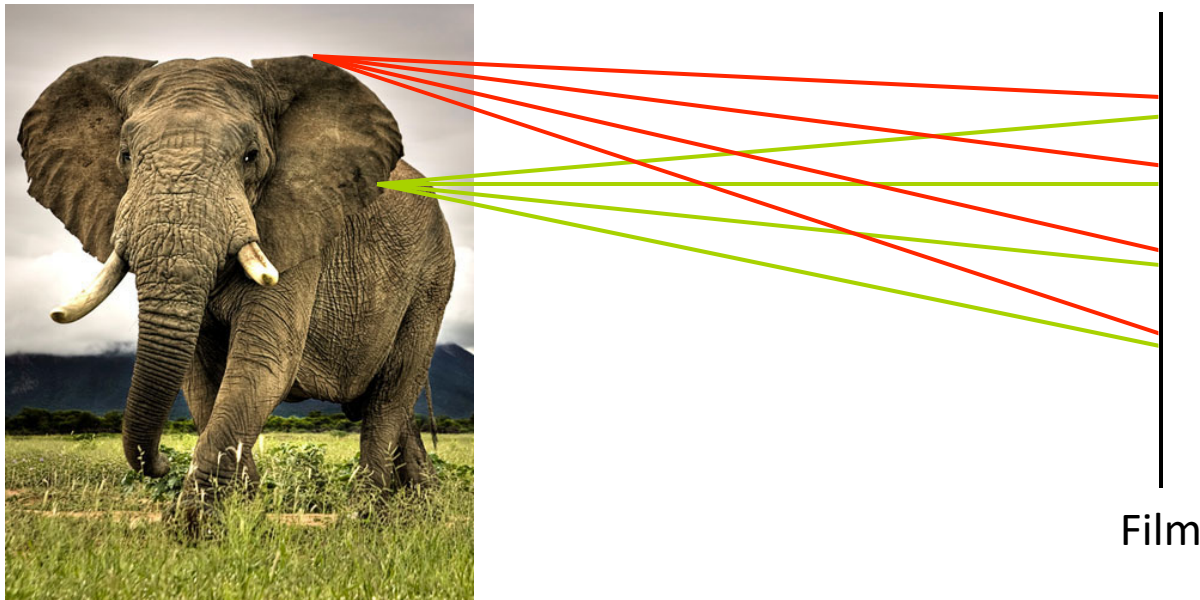

Lecture 3:

Projective geometry and camera calibration

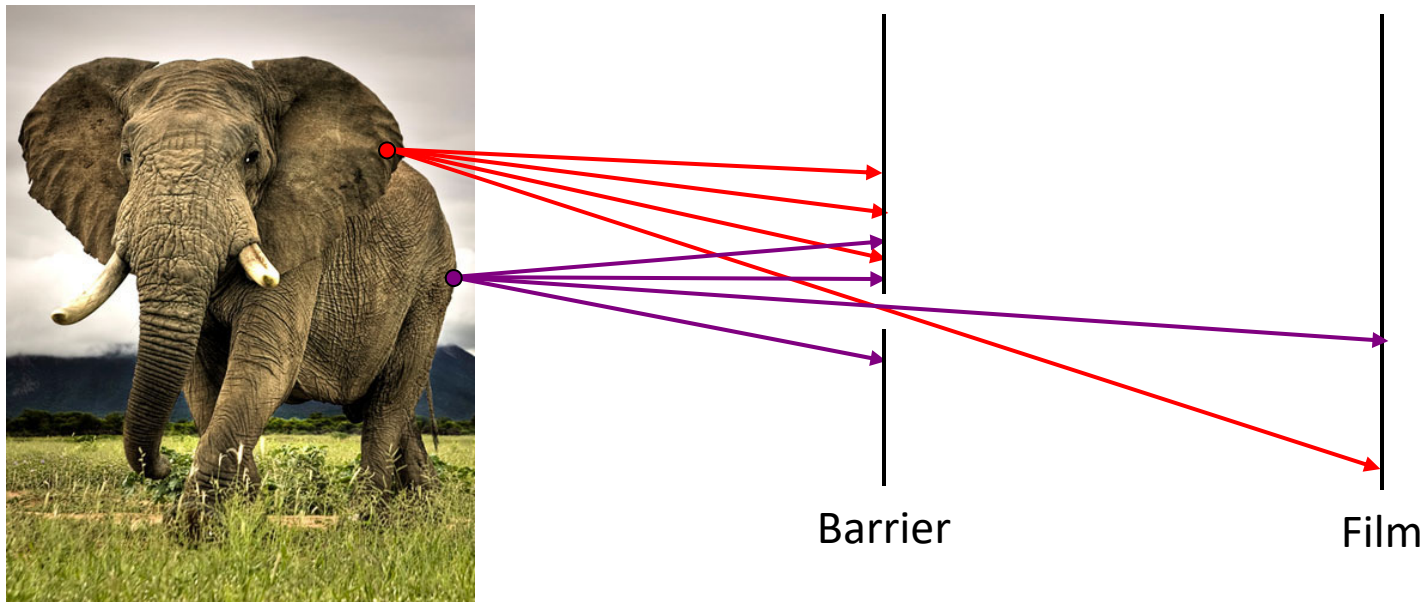
Dr. Stephen Czarnuch

Let's design a camera



- Put a piece of film in front of an object
- Do we get a reasonable image?

Solution



- We introduce pinhole camera: add a barrier to block off most of the rays
 - This reduces blurring
 - The opening known as the aperture
- The image gets inverted on the film

Some history

Camera Obscura: the pre-camera

- Known during classical period in China and Greece (e.g. Mo-Ti, China, 470BC to 390BC)

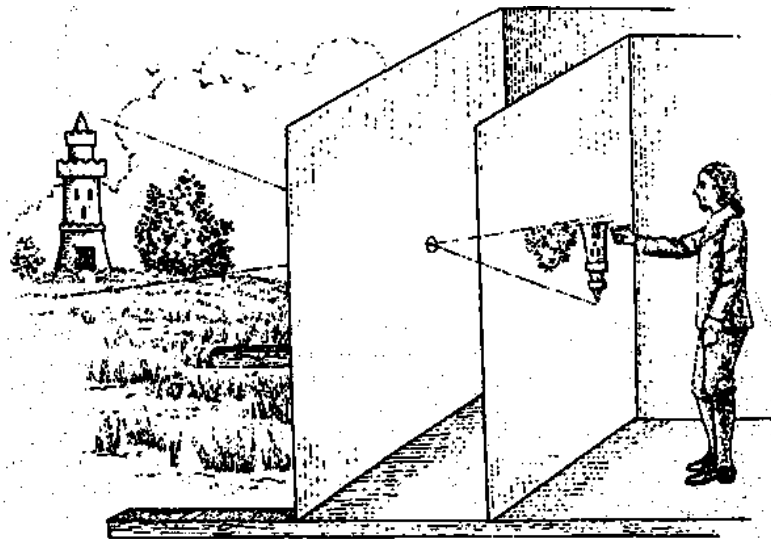


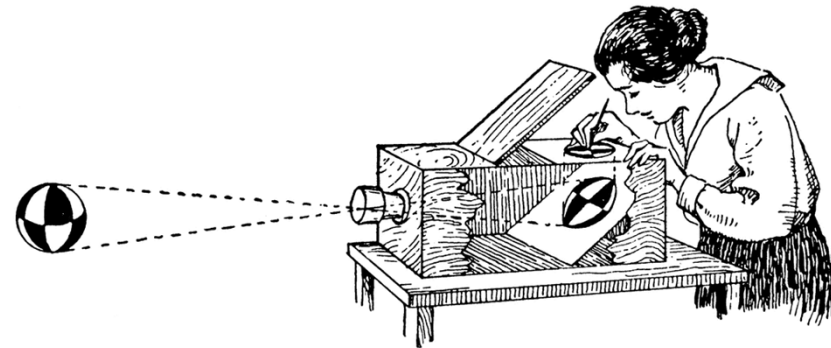
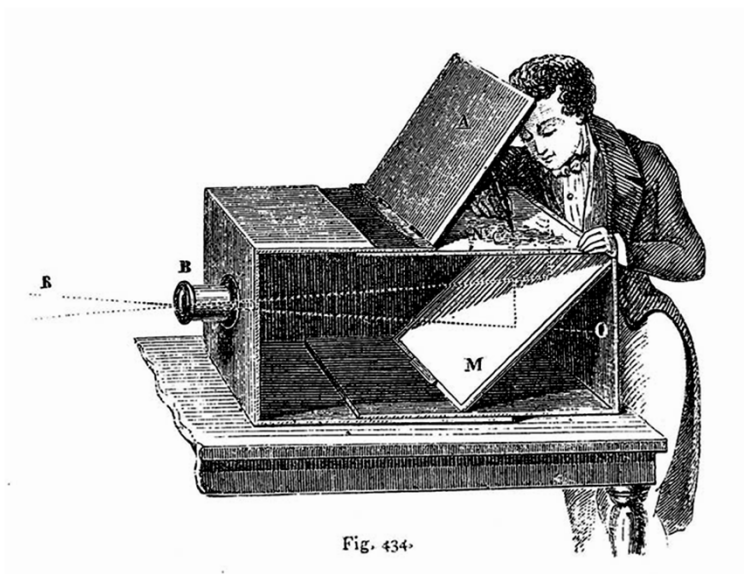
Illustration of Camera Obscura



Freestanding camera obscura at UNC
Chapel Hill



Camera Obscura used for tracing



Lens Based Camera Obscura, 1568

First photograph

- Oldest surviving photograph



By: Joseph Niepce, 1826



Stored at UT Austin

- The image depicts the view from an upstairs window at Niépce's estate, Le Gras, in the Burgundy region of France

Camera and world geometry

- Once we have the 2D image, we think about the relationship of the 2D image to the 3D real world
- Some questions that arises:
 - Which is taller: The building or the bridge?
 - How tall is the bridge?
 - Which is closer: The building or the tree?
 - Are the train rails parallel?

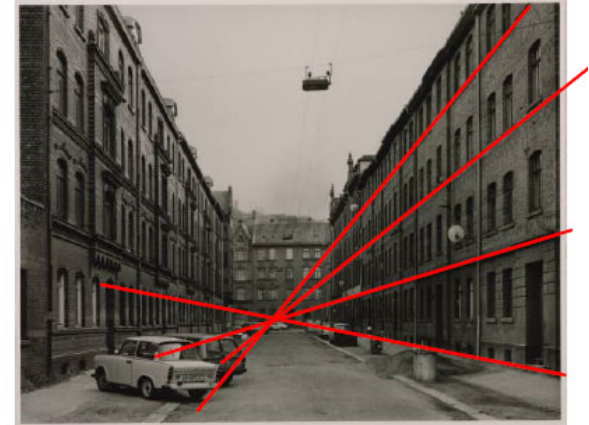


Projection can be tricky...



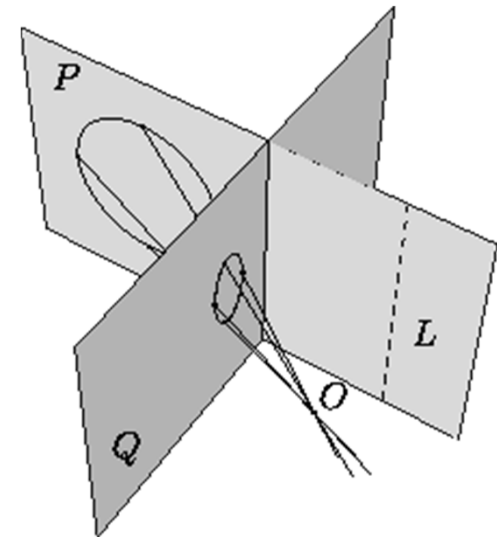
Projective geometry

- A projective transformation (homography):
 - A transformation that maps lines from 3D space to lines of 2D space
 - Can be represented by an invertible 3×3 matrix (isomorphism)
 - Does not necessarily preserve parallelism
- Parallel lines:
 - Meet at infinity in 3D space
 - Do not necessarily stay parallel after projected to 2D space



Perspective transformation

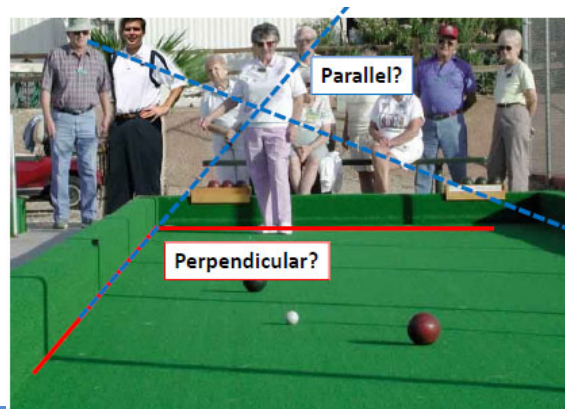
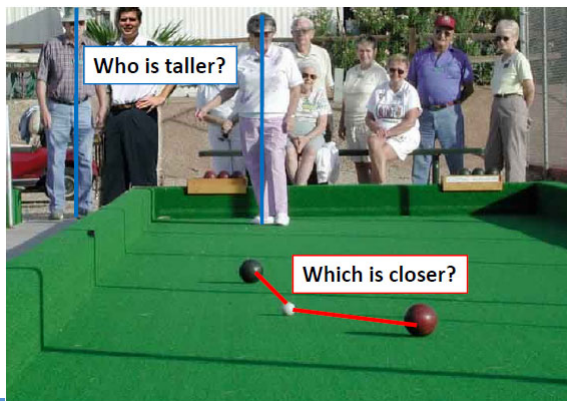
- A common example of a projective transformation is given by a **perspective transformation**
- All projected lines go through a common center called the optical center (O)



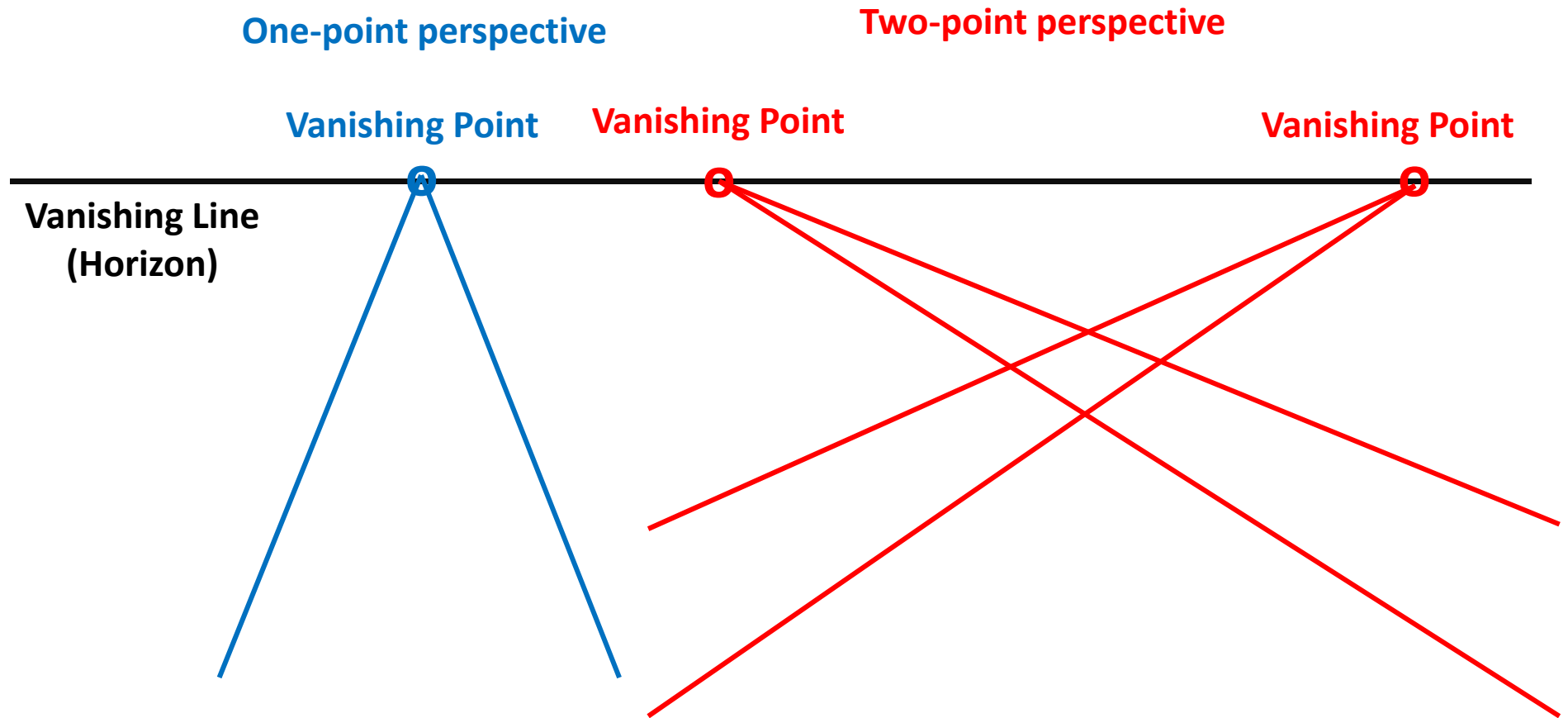
<http://www.geom.uiuc.edu/docs/reference/CRC-formulas/node16.html>

Projective geometry properties

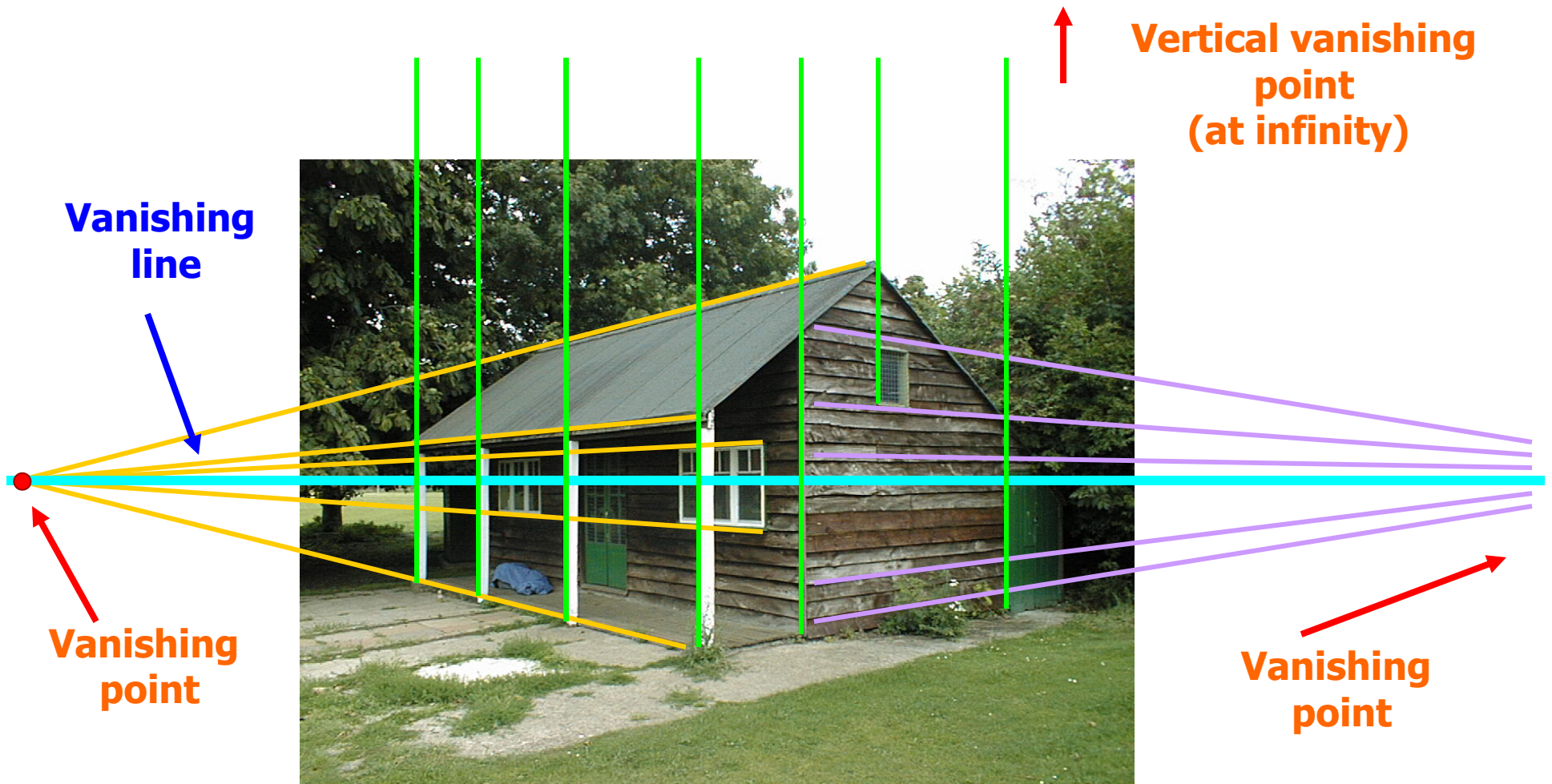
- What are not preserved:
 - Length
 - Angles
 - Many parallel lines in the world intersect in the image at a **vanishing point**
- What is preserved:
 - Straight lines are still straight



Vanishing points and lines



Vanishing points and lines



Focal length of a camera

Think of focal length as *Zooming*



24mm



50mm



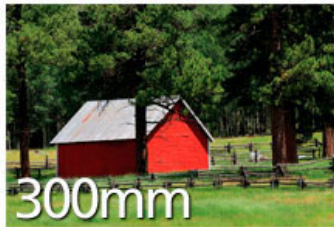
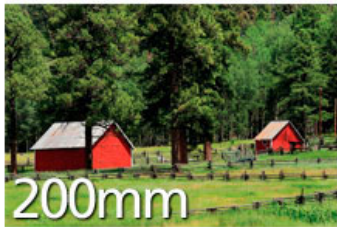
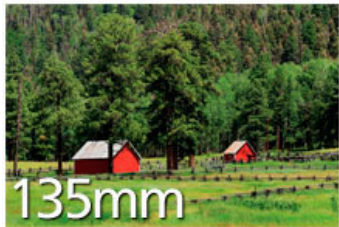
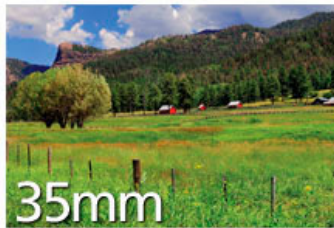
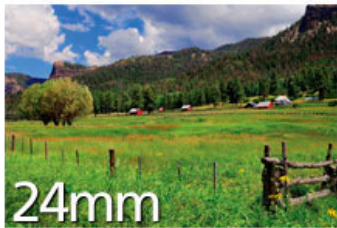
200mm



800mm



Focal length of a camera cont..

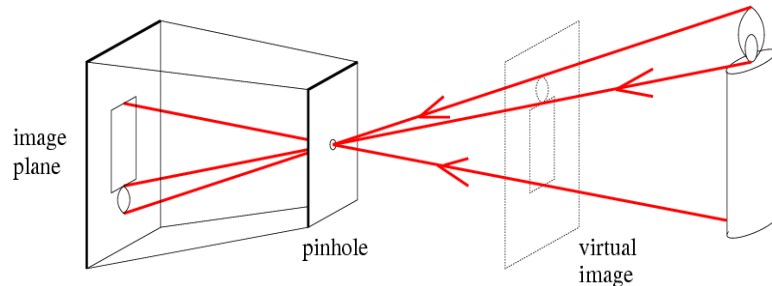


Focal length tells about the *angle of view* and the *magnification*

Longer focal length:
Narrower the angle of view
Higher the magnification

Shorter focal length:
Wider angle of view
Lower the magnification

Camera parameters and the perspective projection



Given point $P(X,Y,Z)$ in the real world.

Camera parameters:

f = focal length (m)

k, l = pixels/m

The pixel coordinates (point p) on the image plane are then (with no skew):

$$(x, y) = \left(kf \frac{X}{Z}, lf \frac{Y}{Z} \right)$$

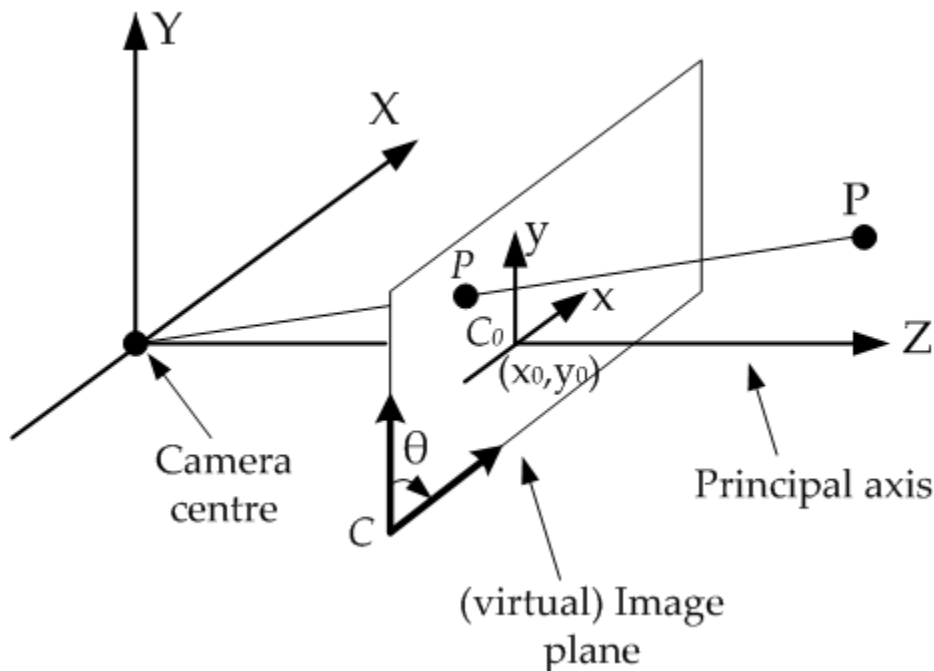
If the image plane is centred on the principal axis at (x_0, y_0) then:

$$(x, y) = \left(kf \frac{X}{Z} + x_0, lf \frac{Y}{Z} + y_0 \right)$$

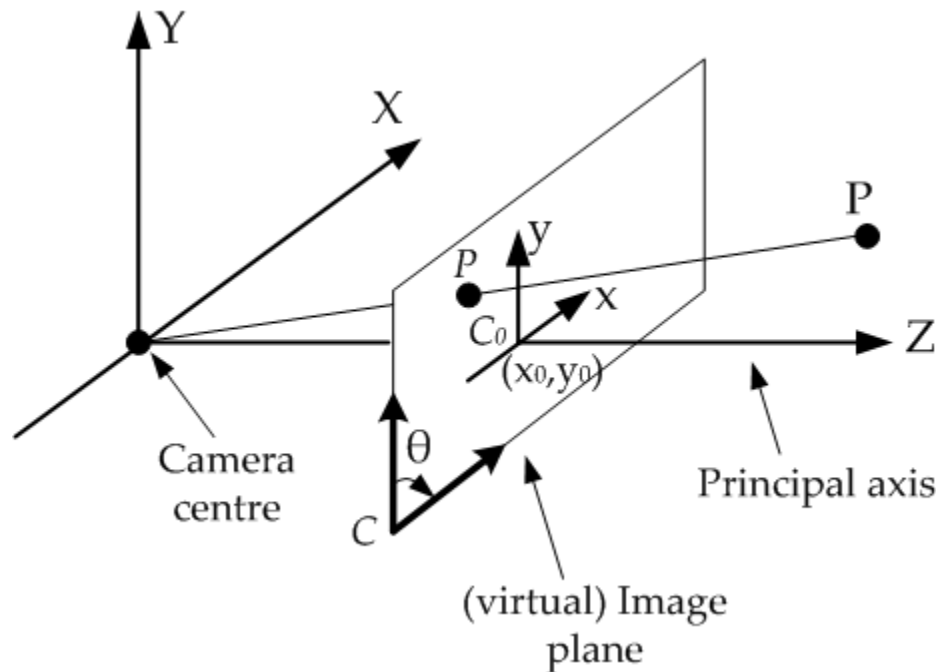
Note:

C_0 = principal point/optical centre

C = origin of the image coordinate system



Camera parameters (intrinsic)



$$x = \alpha \frac{x}{z} - \alpha \cot \theta \frac{y}{z} + x_0$$

$$y = \frac{\beta}{\sin \theta} \frac{y}{z} + y_0$$

$\cot \theta$ – skew parameter

Represented with matrices:

$$p = \frac{1}{z} MP = \frac{1}{z} [K \ 0] P$$

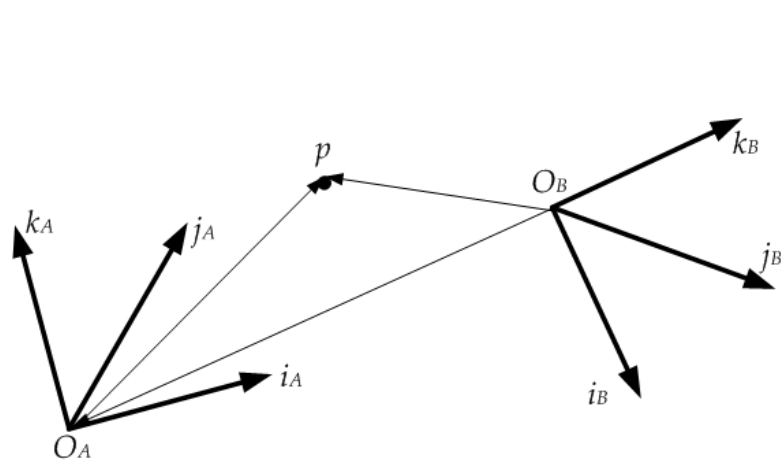
$$K = \begin{bmatrix} \alpha & -\alpha \cot \theta & x_0 \\ 0 & \frac{\beta}{\sin \theta} & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

P = coordinate of a point in the camera coordinate system (real world)

p = coordinate of a point in the image plane with origin C

$\alpha, \beta, \theta, x_0, y_0$ – intrinsic parameters

Rigid transformations



Rotation

Translation

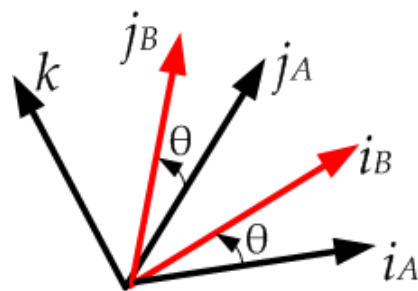
$${}^B p = {}^B R ({}^A p) + {}^B o_A$$

${}^B p$ – Coordinate vector of point p in the frame (B)

${}^B R$ – Rotation matrix describing the frame (A) in the coordinate system (B)

$${}^B R \stackrel{\text{def}}{=} \begin{bmatrix} i_A \cdot i_B & j_A \cdot i_B & k_A \cdot i_B \\ i_A \cdot j_B & j_A \cdot j_B & k_A \cdot j_B \\ i_A \cdot k_B & j_A \cdot k_B & k_A \cdot k_B \end{bmatrix}$$

Example: Suppose $k_A = k_B = k$ and angle between i_A and i_B is θ . Find the rotation matrix ${}^B R$



$${}^B R = \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Camera parameters (extrinsic)

- The camera frame (C) is distinct from the world frame (W)
- Consider point P :

$${}^c_P = \frac{1}{z} K [R \quad t] {}^w_P$$

where:

K – intrinsic parameter matrix

$[R \quad t]$ – extrinsic parameter matrix

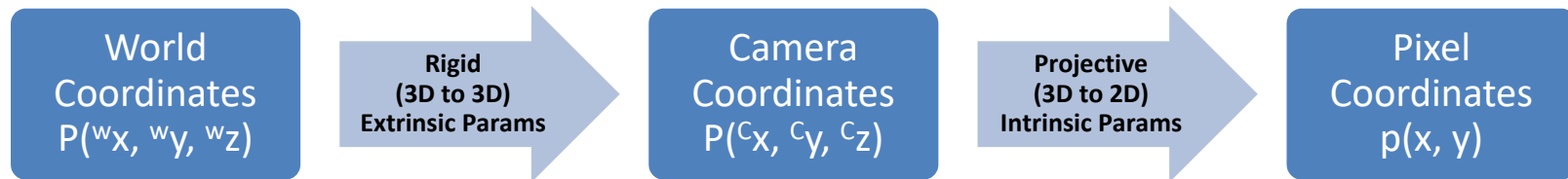
R – rotation matrix

t – translation matrix

${}^w_P = ({}^w_x, {}^w_y, {}^w_z, 1)^T$ – coordinate vector of P in world frame

${}^c_P = (x, y, 1)^T$ – coordinate of P in the image frame

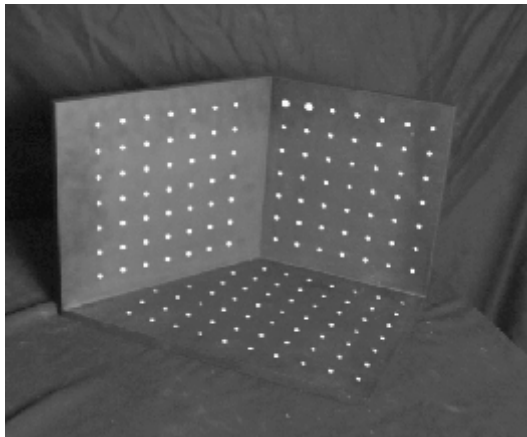
Camera parameters



- Extrinsic parameters:
 - Represent a rigid transformation from **3-D world coordinate system** to **3-D camera's coordinate system**.
- Intrinsic parameters:
 - Represent a projective transformation from the **3-D camera's coordinates** into **the 2-D image coordinates**.
- Overall 11 parameters representing 5 intrinsic and 6 extrinsic (3 angles defining R and 3 coordinates of T) parameters

Camera calibration approaches

- Camera calculates the 11 intrinsic and extrinsic parameters



- Consider the general form:

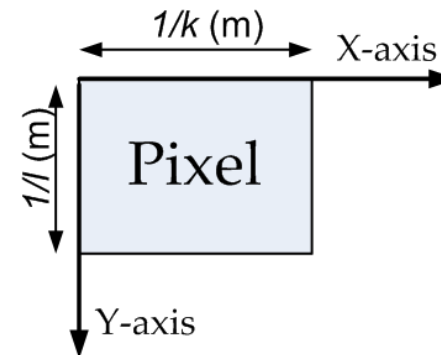
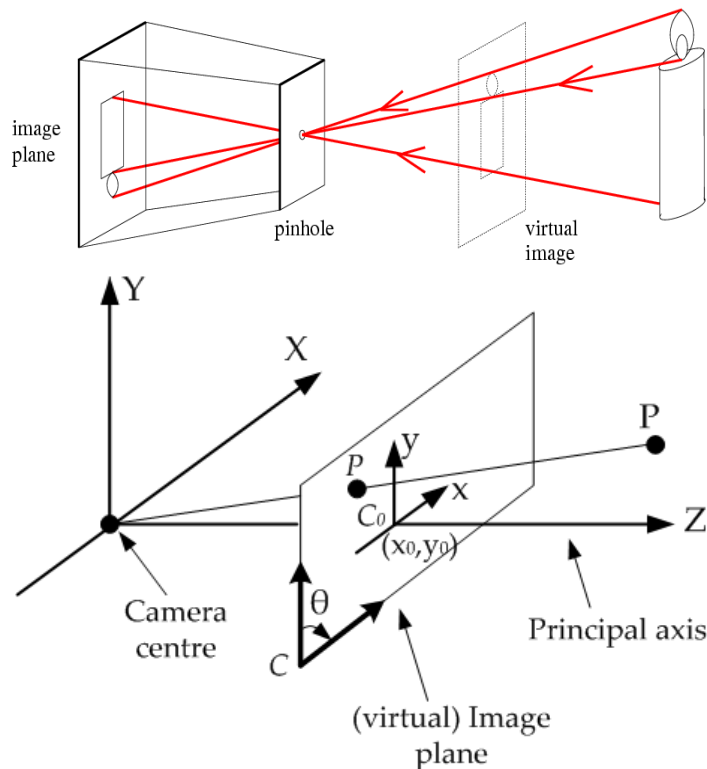
$$wp = K[R \quad |t]P = MP$$

w – scale factor

- Directly estimates 11 unknowns in the **M** matrix using **known** 3D points (X_i, Y_i, Z_i) and **measured** pixel coordinates (x_i, y_i)
- Take a photograph of the target and find (by hand or automatically) the pixel coordinates of the known points on the checkerboard.
- A set of constraints relating the pixel positions (x_i, y_i) and the scene positions (X_i, Y_i, Z_i) can be found.

Understanding some concepts in camera calibration

Expressing in pixels



How many pixels for 1m in X-axis? k

How many pixels for 1m in Y-axis? l

f – focal length

$$x = f \frac{x}{Z} \text{ and } y = f \frac{y}{Z} \text{ (both in meters wrt } C_0)$$

$$x = kf \frac{x}{Z} \text{ and } y = lf \frac{y}{Z} \text{ (both in pixels wrt } C_0)$$

$$x \text{ (in pixels wrt } C) = \overset{\alpha}{kf} \frac{x}{Z} + x_0$$

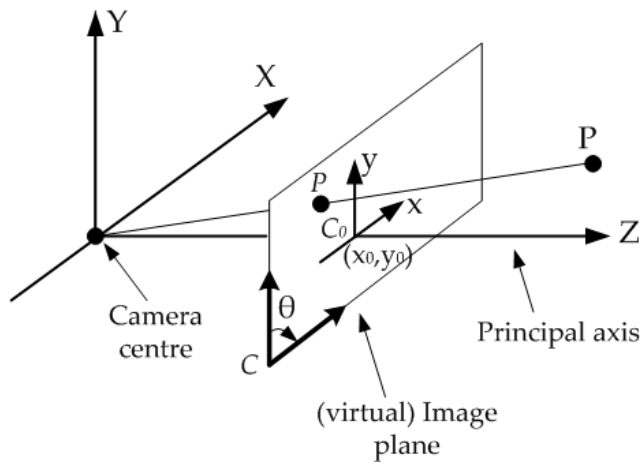
$$y \text{ (in pixels wrt } C) = \overset{\beta}{lf} \frac{y}{Z} + y_0$$

C_0 – principal point, optical center

C – origin of the retinal coordinate system (image frame)

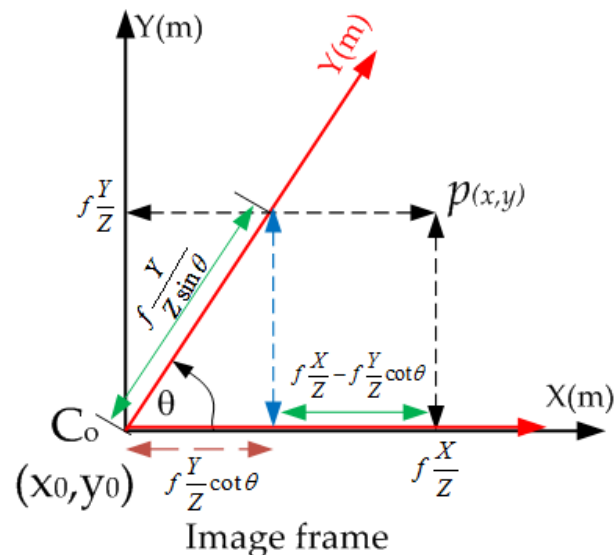
(x_0, y_0) – coordinates of the principal point wrt C (in pixels)

Skew parameter, $\cot \theta$



$$\begin{cases} x = \frac{\alpha}{kf} \frac{X}{Z} + x_0 \\ y = \frac{\beta}{lf} \frac{Y}{Z} + y_0 \end{cases} \text{ in pixels wrt } C, \text{ when } \theta = \frac{\pi}{2}$$

$$\begin{cases} x = \alpha \frac{X}{Z} - \alpha \cot \theta \frac{Y}{Z} + x_0 \\ y = \frac{\beta}{\sin \theta} \frac{Y}{Z} + y_0 \end{cases} \text{ in pixels wrt } C, \text{ when } \theta < \frac{\pi}{2}$$



C – Origin of the ratinal coordinate system (image frame)

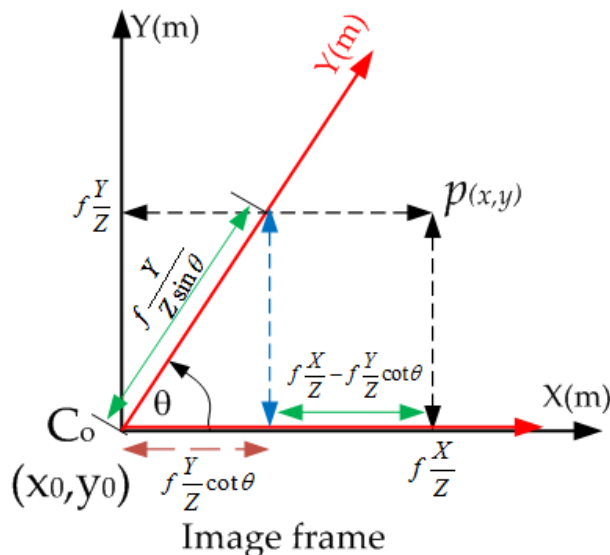
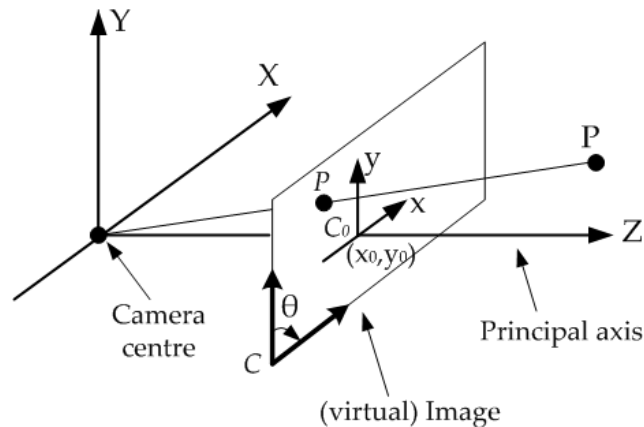
C_0 – Principal point, optical center

(x_0, y_0) – Coordinates of the principal point wrt C (in pixels)

(x, y) – Coordinates of the point p in image frame wrt C (in pixels)

θ – Skew due to manufacturing errors

Developing the intrinsic parameter matrix



Let $p_{img} = (x, y, 1)^T$ and $P_{cam} = (x, y, z, 1)^T$

$$x = \alpha \frac{x}{z} - \alpha \cot \theta \frac{y}{z} + x_0$$

$$y = \frac{\beta}{\sin \theta} \frac{y}{z} + y_0$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \frac{1}{z} \begin{bmatrix} \alpha & -\alpha \cot \theta & x_0 & 0 \\ 0 & \frac{\beta}{\sin \theta} & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}_{3 \times 4} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

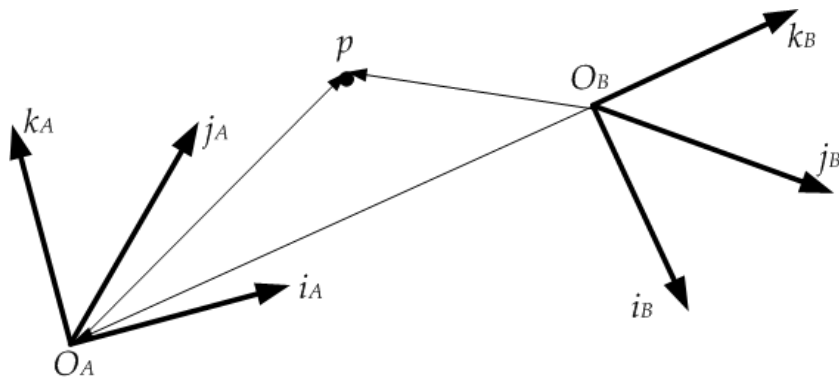
$$p_{img} = \frac{1}{z} K_{3 \times 4} P_{cam}$$

$K_{3 \times 4}$ – Intrinsic parameter matrix

$P_{cam} = (x, y, z, 1)^T$ – coordinate vector of P in the camera coordinate system (expressed in meters)

$p_{img} = (x, y, 1)^T$ – coordinate vector of p in the image frame (expressed in pixels)

Extrinsic parameter matrix mapping world coordinates to camera coordinates (3D)



Rotation Translation

$${}^B P = {}^B_A R ({}^A P) + \overrightarrow{O_B O_A}$$

${}^A P$ – point P expressed in the frame A
 ${}^B P$ – point P expressed in the frame B
 ${}^B_A R$ – Rotation matrix used to rotate frame A to B

$$P_{cam\ 3D} = {}^C_W R(P_{world\ 3D}) + \overrightarrow{O_{cam} O_{world}}$$

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_{cam\ 3D} = \begin{bmatrix} [{}^C_W R]_{3 \times 3} & [t_x\ t_y\ t_z]^T \\ [0]_{1 \times 3} & 1 \end{bmatrix}_{4 \times 4} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_{world\ 3D}$$

$$\Rightarrow P_{cam\ 3D} = [R\ t]_{4 \times 4} P_{world\ 3D}$$

R – Rotation matrix

$t = [t_x\ t_y\ t_z]^T$ – Translation vector

We know that: $p_{img\ 2D} = \frac{1}{z} K_{3 \times 4} P_{cam\ 3D}$

$$\Rightarrow p_{img\ 2D} = \frac{1}{z} K_{3 \times 4} [R\ t]_{4 \times 4} P_{world\ 3D}$$

$$\Rightarrow \text{Let } K_{3 \times 4} [R\ t]_{4 \times 4} = M_{3 \times 4}$$

The projection matrix M combining intrinsic and extrinsic parameters



$$M_{3 \times 4} = \begin{bmatrix} \alpha r_1 - \alpha \cot \theta r_2 + x_0 r_3 & \alpha t_x - \alpha \cot \theta t_y + x_0 t_z \\ \frac{\beta}{\sin \theta} r_2 + y_0 r_3 & \frac{\beta}{\sin \theta} t_y + y_0 t_z \\ r_3 & t_z \end{bmatrix}$$

where

r_1, r_2, r_3 – Three rows of the rotation matrix ${}^C_W R$

$t = [t_x \ t_y \ t_z]$ – Translation vector

Back to camera calibration...

Camera calibration cont..

$$w \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$x_i = \frac{m_{11}X_i + m_{12}Y_i + m_{13}Z_i + m_{14}}{m_{31}X_i + m_{32}Y_i + m_{33}Z_i + m_{34}}$$

$$y_i = \frac{m_{21}X_i + m_{22}Y_i + m_{23}Z_i + m_{24}}{m_{31}X_i + m_{32}Y_i + m_{33}Z_i + m_{34}}$$

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -x_1X_1 & -x_1Y_1 & -x_1Z_1 & -x_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -y_1X_1 & -y_1Y_1 & -y_1Z_1 & -y_1 \\ \vdots & & & & & & & & & & & \\ X_N & Y_N & Z_N & 1 & 0 & 0 & 0 & 0 & -x_NX_N & -x_NY_N & -x_NZ_N & -x_N \\ 0 & 0 & 0 & 0 & X_N & Y_N & Z_N & 1 & -y_NX_N & -y_NY_N & -y_NZ_N & -y_N \end{bmatrix} \begin{bmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ \vdots \\ m_{31} \\ m_{32} \\ m_{33} \\ m_{34} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Camera calibration cont..

- M has 12 parameters:
 - Need 12 data points to solve.
 - However, data has some measurement errors and projection model is only an approximation.
 - Solution
 - Use more data points and find the solution using least squares method
 - Assume the over determined system $\mathbf{Ax}=\mathbf{b}$.
 - Find the vector x that minimizes the mean-squared error measure E such that:
- $$E = \sum_{i=1}^n (a_{i1}x_1 + \dots + a_{in}x_n - b_i)^2 = \|\mathbf{Ax} - \mathbf{b}\|^2$$
- Least squares methods-techniques used to minimize the mean squared error

Normal equations

Consider $E = e \cdot e$, with $e = Ax - b$

To find the vector x which minimizes E :

$$\frac{\partial E}{\partial x_i} = 2e \cdot \frac{\partial e}{\partial x_i} = 0, \forall i \in (1, \dots, n)$$

$$\frac{\partial e}{\partial x_i} = a_i, \text{ where } a_i = (a_{1i}, \dots, a_{mi})^T, \forall i \in (1, \dots, n)$$

$$\Rightarrow a_i^T (Ax - b) = 0$$

Minimum occurs when all directional derivatives are zero

$$\Rightarrow \begin{bmatrix} a_1^T \\ \vdots \\ a_n^T \end{bmatrix} [Ax - b] \Rightarrow \overbrace{A^T Ax = A^T b}^{\text{Normal equation}}$$

$$x = A^t b, \text{ where } A^t = \left[(A^T A)^{-1} A^T \right] = \text{Moore-Penrose pseudoinverse of } A$$

Normal equation:

$$\text{Let } r = b - Ax$$

$$A^T(r) = 0$$

$\Rightarrow r$ is normal to every
vector in the span of A

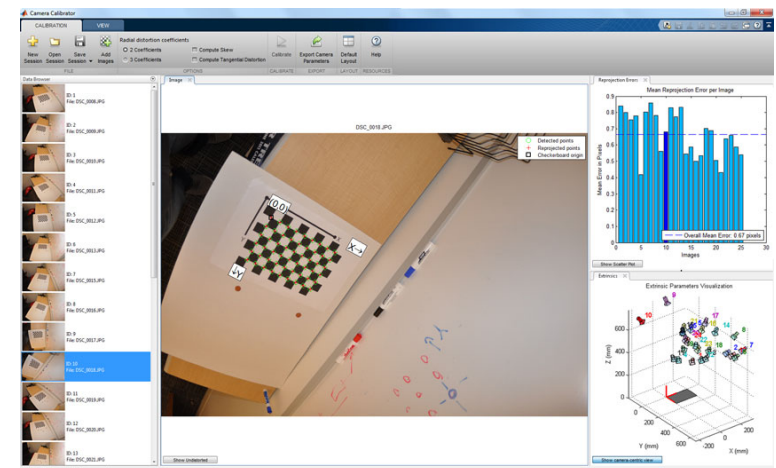
Approaches to camera calibration

- Least squares methods
 - Linear
 - Nonlinear
 - Who invented Least squares method?
 - Carl Friedrich Gauss (1777–1855) claimed to have used it since 1795
 - First published by Adrien-Marie Legendre first published it in 1804
- Nonlinear optimization methods
 - Iterative process
 - More accurate



Camera calibration with Matlab

- Matlab Computer Vision System Toolbox provides functions for the camera calibration workflow. Such as:
 - Automatic detection and location of checkerboard calibration pattern including corner detection
 - Estimation of all intrinsic and extrinsic parameters including axis skew
 - Calculation of radial and tangential lens distortion coefficients
 - Correction of optical distortion
 - Support for single camera and stereo calibration
- The Camera Calibrator app is used to:
 - select and filter calibration images
 - choose radial distortion coefficients
 - view reprojection errors
 - visualize extrinsic parameters, and
 - export camera calibration parameters



Camera distortion

- We used a pinhole camera model
 - No lens
- To represent a real camera the camera model needs to include radial and tangential lens distortions

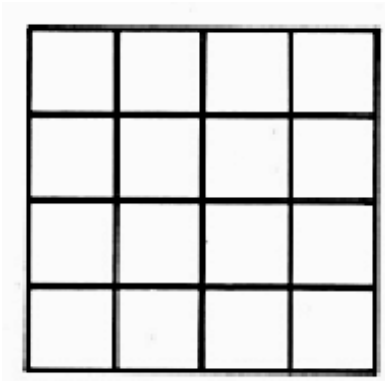
Radial distortion

- Caused by imperfect lenses
- Most noticeable deviations when rays pass through the edge of the lens
- Barrel Distortion, pin cushion distortion

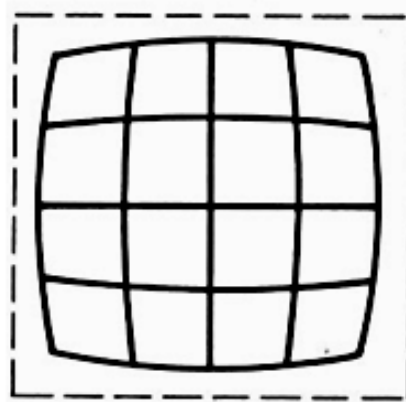
Tangential distortion

- When the lens and the sensor array (image plane) are not parallel
- The effects of tangential distortion may be neglected compared to the radial (barrel) distortion

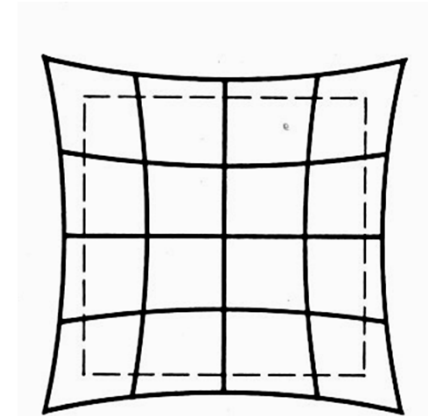
Radial distortion



No distortion



Barrel distortion



pin cushion distortion

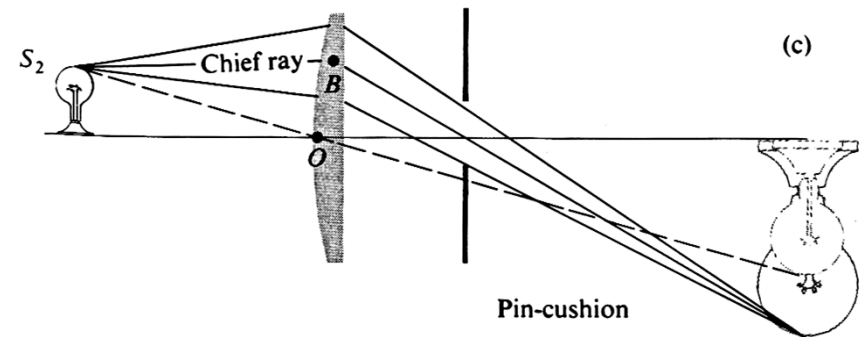
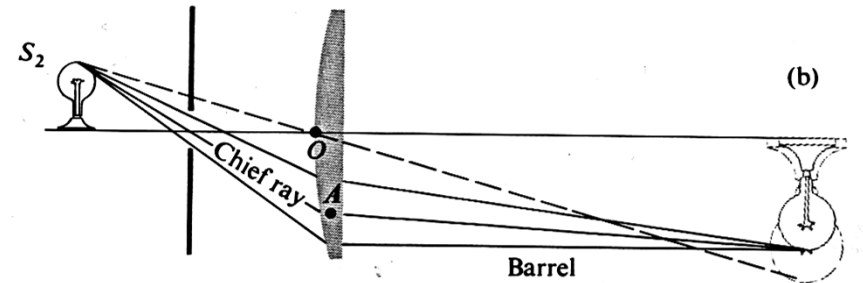
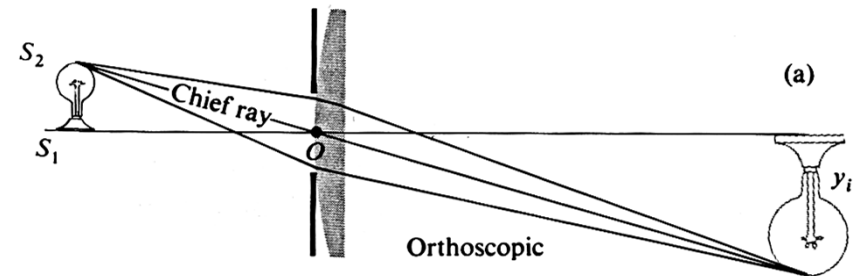


Barrel distortion

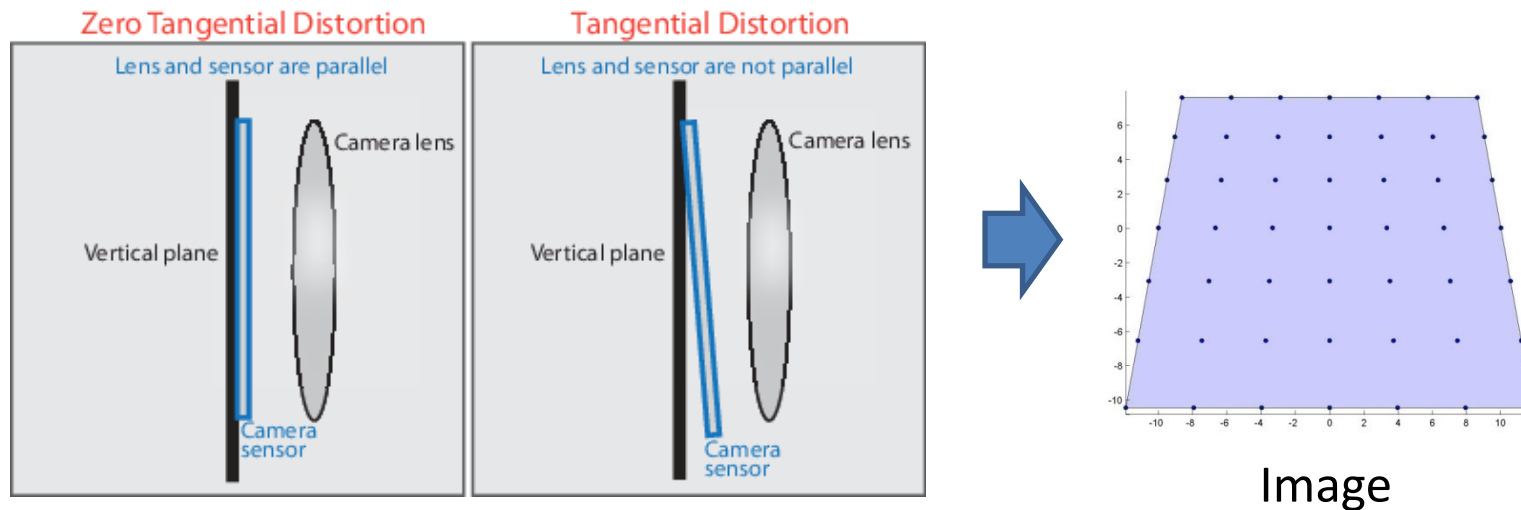


Pin cushion distortion

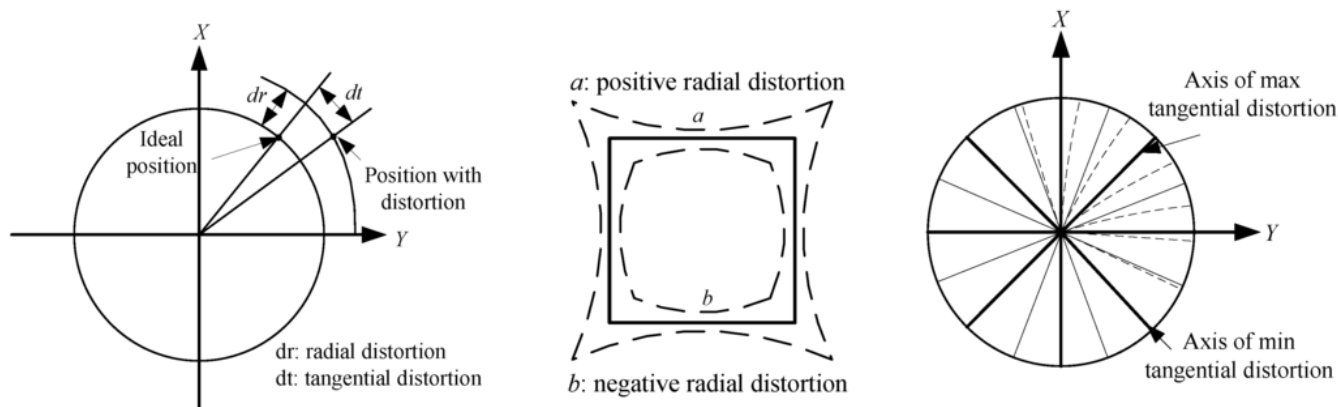
Radial distortion cont..



Tangential distortion

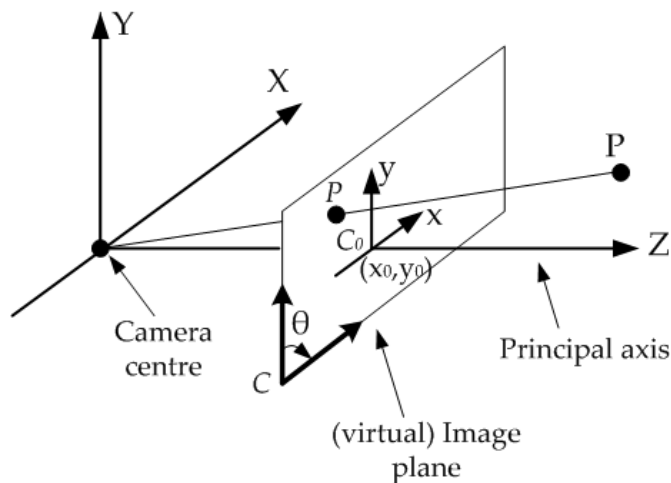


Source: Matlab



Source: Wang, Geng and Jin, *Sensors*, 15(12) 2015, doi:[10.3390/s151229863](https://doi.org/10.3390/s151229863).

Calculating the total distortion



$$x = kf \frac{X}{Z}, y = lf \frac{Y}{Z}$$

$$\frac{1}{k} \times \frac{1}{l} - \text{pixel dimensions}$$

Let $kf = lf = f_{px}$: focal length in pixels

$$x = f_{px} \frac{X}{Z} + x_0 : \text{in pixels}$$

$$y = f_{px} \frac{Y}{Z} + y_0 : \text{in pixels}$$

Let (\hat{x}, \hat{y}) be the normalized image coordinates in pixels calculated by translating to the optical center and dividing by the focal length (in pixels):

$$\hat{x} = \frac{X}{Z}, \hat{y} = \frac{Y}{Z}, \text{ Let } r^2 = \hat{x}^2 + \hat{y}^2$$

$$\text{Let } \delta x_{rd} = \hat{x}(k_1 r^2 + k_2 r^4), \delta y_{rd} = \hat{y}(k_1 r^2 + k_2 r^4)$$

δx_{rd} – radial distortion in x, δy_{rd} – radial distortion in y

k_1, k_2 : Radial distortion coefficients of the lens

$$\delta x_{td} = (2p_1 \hat{x}\hat{y} + p_2(r^2 + 2\hat{x}^2)), \delta y_{td} = (2p_2 \hat{x}\hat{y} + p_1(r^2 + 2\hat{y}^2))$$

δx_{td} – tangential distortion in x, δy_{td} – tangential distortion in y

p_1, p_2 : Tangential distortion coefficients of the lens

Putting all together:

$$x_d = \hat{x} + \delta x_{rd} + \delta x_{td}, y_d = \hat{y} + \delta y_{rd} + \delta y_{td}$$

(x_d, y_d) – Distorted pixel coordinates

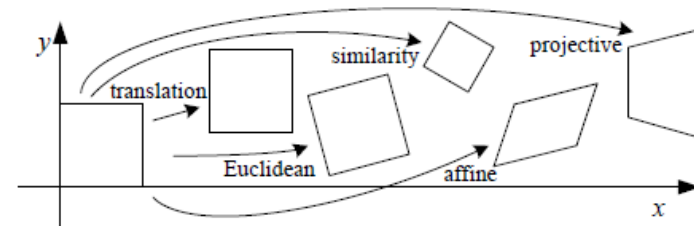
Applying focal length and translating the image center:

$$x = f_{px} x_d + x_0, y = f_{px} y_d + y_0 \leftarrow \text{To model lens distortion}$$

Geometric Transformation

Geometric transformations

- Modify the spatial relationship between pixels in an image
- Also called *rubber-sheet* transformations
- Consists of two operations:
 1. Spatial transformation of coordinates
 2. Intensity interpolation: assigning intensity values to spatially transformed pixels
- Applications such as **image registration** (aligning two or more images of the same scene)



Basic set of 2D planar transformations

Spatial transformations and image registration

Spatial transformations modify coordinates of an image according to a pre-defined transformation matrix **T**

Affine transformations

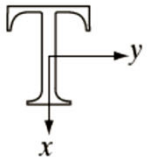
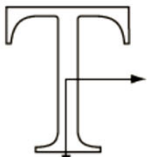

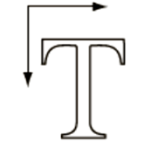


- Affine transform includes: translation, rotation, scaling and shear
- A commonly used class spatial coordinate transformation

The general form of affine transform matrix is as follows:

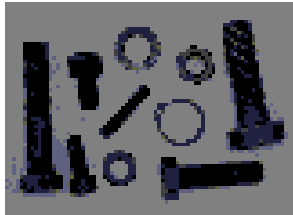
$$[x' \ y' \ 1]_{1 \times 3} = [x \ y \ 1]_{1 \times 3} T_{3 \times 3} = [x \ y \ 1]_{1 \times 3} \begin{bmatrix} t_{11} & t_{12} & 0 \\ t_{21} & t_{22} & 0 \\ t_{31} & t_{32} & 1 \end{bmatrix}_{3 \times 3}$$

(x, y) – Pixel coordinates of the original image

(x', y') – Pixel coordinates of the transformed image

| Transformation Name | Affine Matrix, T | Coordinate Equations | Example |
|---------------------|--------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| Identity | $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ | $x = v$ $y = w$ |  |
| Scaling | $\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$ | $x = c_x v$ $y = c_y w$ |  |
| Rotation | $\begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$ | $x = v \cos \theta - w \sin \theta$ $y = v \sin \theta + w \cos \theta$ |  |
| Translation | $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$ | $x = v + t_x$ $y = w + t_y$ |  |
| Shear (vertical) | $\begin{bmatrix} 1 & 0 & 0 \\ s_v & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ | $x = v + s_v w$ $y = w$ |  |
| Shear (horizontal) | $\begin{bmatrix} 1 & s_h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ | $x = v$ $y = s_h v + w$ |  |

Source: Figure 2.3, Gonzalez, R. C., & Woods, R. E. (2017). *Digital Image Processing (4 ed.)*. New York, NY: Pearson.



Original Image

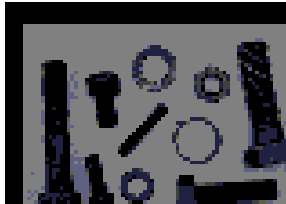


Image translation
Imtranslate(i,[tx ty])

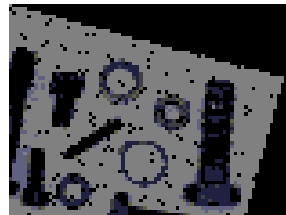


Image rotation
Imrotate(i,angle)

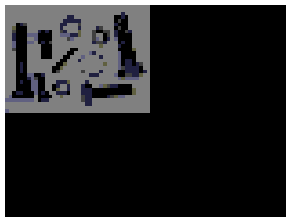
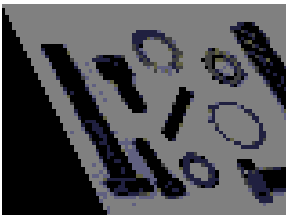


Image scale
Imscale(i,scale)



Shear vertical



Shear horizontal

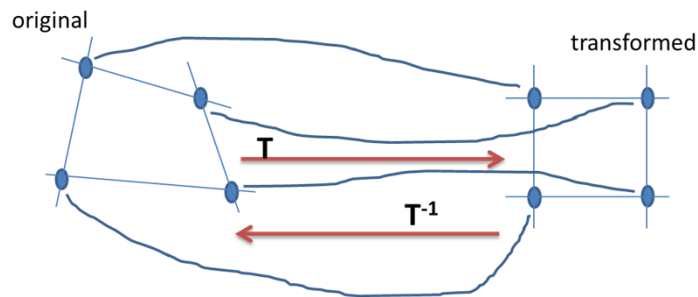
```
function affine_transforms
clear;
T=[1 0 0; 0 1 0; 100 100 1]; % translation matrix
theta=pi/12;
R=[cos(theta) sin(theta) 0; -sin(theta) cos(theta) 0; 0 0 1]; % Rotation matrix
S=[0.5 0 0; 0 0.5 0; 0 0 1]; % scaling matrix
Sh_v=[1 0 0; 0.5 1 0; 0 0 1]; % shear vertical matrix
Sh_h=[1 0.5 0; 0 1 0; 0 0 1]; %shear horizontal matrix
% Add transform matrices and make 3D matrix
Affine=zeros(3,3,5);
Affine(:,:,1)=T; Affine(:,:,2)=R; Affine(:,:,3)=S;
Affine(:,:,4)=Sh_v; Affine(:,:,5)=Sh_h;
```

```
gray_im=imread('Fasteners_1_gray.png'); % Read the grayscale image
H=size(gray_im,1); % Read the height of the image
W=size(gray_im,2); % Read the width of the image
subplot(6,1,1);
imshow(gray_im); % Show original image
```

```
for j=1:5,
    Test_mat=Affine(:,:,j);
    T_image=zeros(H,W); % a null matrix to hold translated image
    for iy=1:H,
        for ix=1:W,
            temp=uint16([ix iy 1]*Test_mat);
            if((temp(1)>0)&&(temp(2)>0)&&(temp(1)<W+1)&&(temp(2)<H+1))
                T_image(temp(2),temp(1))=gray_im(iy,ix);
            end
        end
    end
    subplot(6,1,(j+1))
    imshow(T_image,[0 255]); % Scale and show each image
end
end
```

Example application: Image registration

- Used to align two or more images of the same scene



Examples:

- Align two or more images taken at the same time but from different imaging systems
 - Align satellite images of a given location taken days apart
- In image registration, we have available the input and output images, but the transformation function T is unknown and needs to be estimated
 - One approach is called control points method, where the correspondence of few control points (tie points) of input and reference images are known



Example code from Matlab

```
I1 = imread('westconcordaerial.png'); % input image
I2 = imread('westconcordorthophoto.png'); % reference image
cpselect(I1,I2); % Control point selection tool
tform = cp2tform(input_points, base_points, 'affine'); %
infer spatial transformation from control point pairs
registered=imtransform(I1,tform,'bilinear'); % applying 2D
spatial transform to image, The form of interpolation is
bilinear
Figure('registered'),
imshow(registered,[]);
imshowpair(registered,I2,'blend'); % compare the difference
between images
```



+



Additional reference for camera calibration:

David A. Forsyth , Jean Ponce, Computer Vision: A Modern Approach, Prentice Hall, 2003

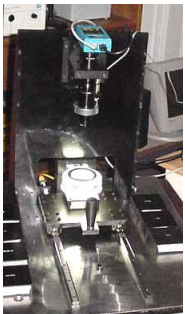
Lecture 3 cont..:

Recap camera calibration, camera distortion and geometric transformation

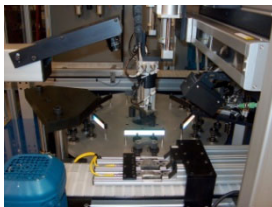
Dr. Stephen Czarnuch

Recap: Camera calibration

- Camera calibration is the estimation of the above **intrinsic** (5) and **extrinsic** (6) parameters of the camera. These parameters are used to correct lens distortion, determine the location of the camera in 3D space, or measure the size of an object in world's units
- Applications of camera calibration:
 - In machine vision systems to detect and measure objects
 - Robotics and navigation systems
 - 3-D scene recognition



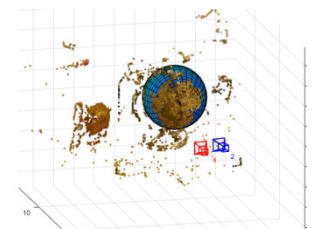
Machine vision system
measuring parts



Mobile robotic navigation and mapping



Estimate 3D structure



Recap: Camera calibration cont..

- Intrinsic parameters
 - Parameters of the lens and image sensor
 - Focal length in pixels, optical center in pixels, skew coefficient
- Extrinsic parameters
 - Parameters of the rigid transformation from 3D world coordinates to 3D camera coordinates

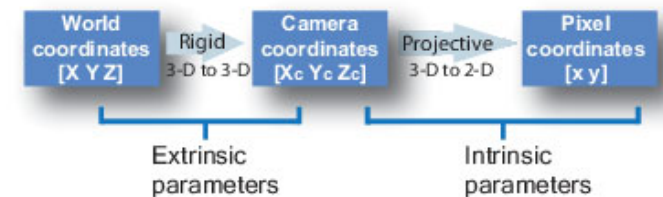
$$K \stackrel{\text{def}}{=} \begin{bmatrix} \alpha & -\alpha \cot \theta & x_0 \\ 0 & \frac{\beta}{\sin \theta} & y_0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{with } \alpha = kf, \beta = lf, \text{optical center} = (x_0, y_0) \text{ and skew coefficient} = \theta$$

- 3 angles defining the rotation matrix and 3-axis translation vector

$\begin{bmatrix} R & t \end{bmatrix}$ – extrinsic parameter matrix

$R = {}^C_w R$ – rotation matrix

$t = [x_t, y_t, z_t]^T = {}^C O_w$ – translation vector



Recap: Camera calibration cont..

- With matrix multiplication using both intrinsic and extrinsic parameters, a point P in world coordinates can be mapped to point p in image frame (pixels)

$$p_{ic} = \mathbf{K} \begin{bmatrix} \mathbf{R} | \mathbf{t} \end{bmatrix} P_{wc} = \mathbf{M}_{3 \times 4} P_{wc}$$

wc – world coordinates, ic – coordinates of the image frame (in pixels)

[M] has 12 parameters: needs 12 data points. However, data has some measurement errors and projection mode is only an approximation.

Solution: Use more data points ($N \gg 6$) and find the solution using the **least squares method**

Assume the over determined system $Ax = b$ and find the vector x that minimizes the mean squared error:

$$\Rightarrow A^T Ax = A^T b \rightarrow x = A^* b, \text{ where } A^* = \left[(A^T A)^{-1} A^T \right]$$

