
MAKE POVERTY HISTORY
JULIAN

Lecture 5
坐标变换与视觉测量

给我一个摄像头，我可以用它来丈量天下

七月在线 金老师

2016年9月24日

角点检测

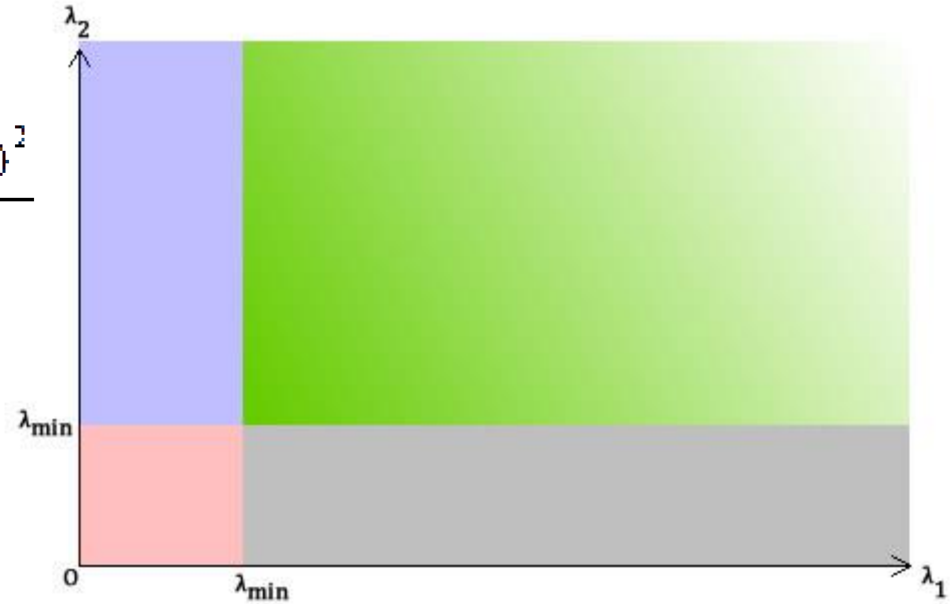
$$R = \det(M) - \alpha\,\mathrm{trace}(M)^2 = \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$$

Nobel, 1988

$$cim = \frac{I_x^2 I_y^2 - (I_x I_y)^2}{I_x^2 + I_y^2}$$

Shi-Tomasi, 2000

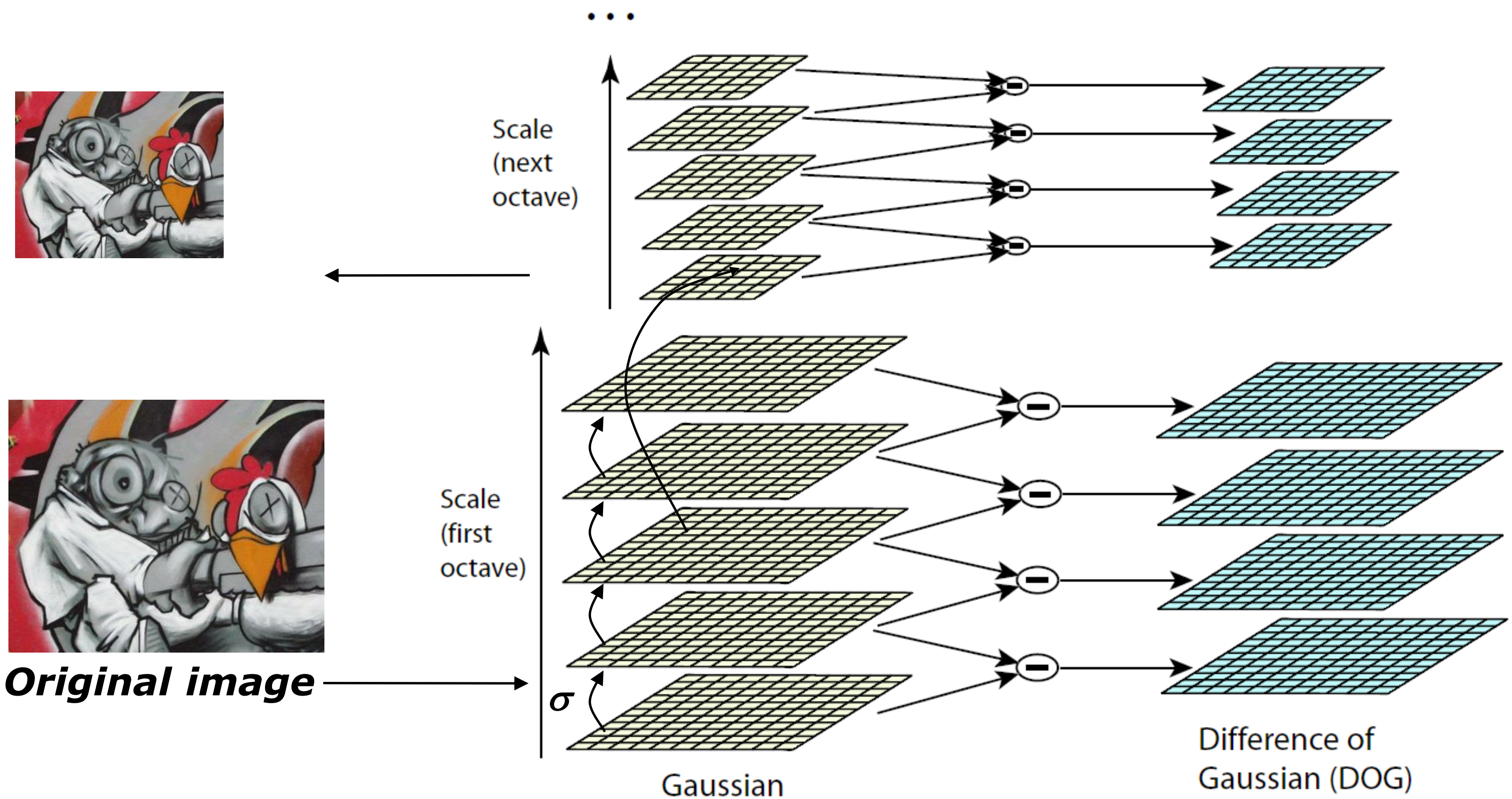
$$R = \min(\lambda_1 \lambda_2)$$



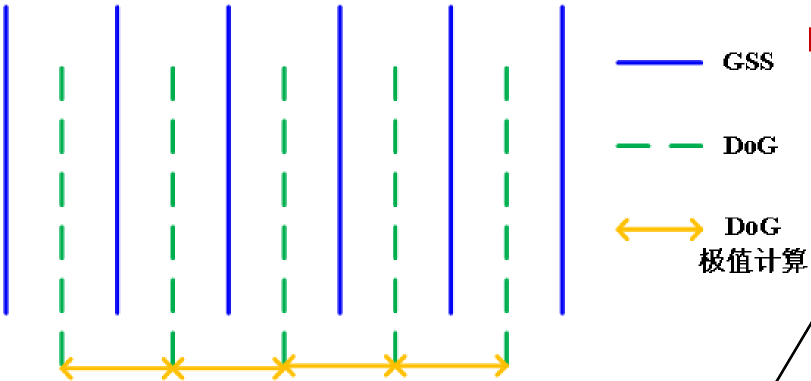

```
cv::goodFeaturesToTrack(image,corners,
    500,   // maximum number of corners to be returned
    0.01,   // quality level
    10);   // minimum allowed distance between points
```

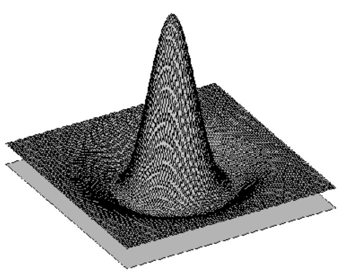
DoG – Efficient Computation

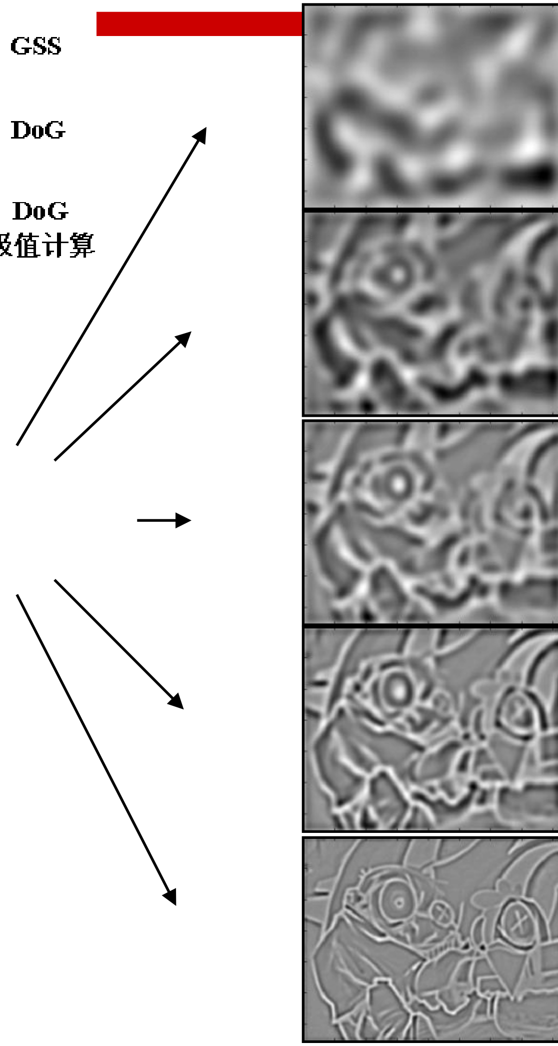
- Computation in Gaussian scale pyramid

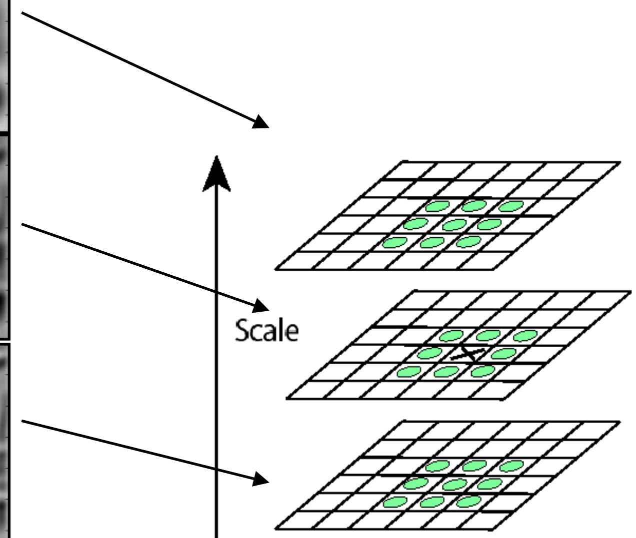


Find local maxima in position-scale space of **Difference-of-Gaussian**

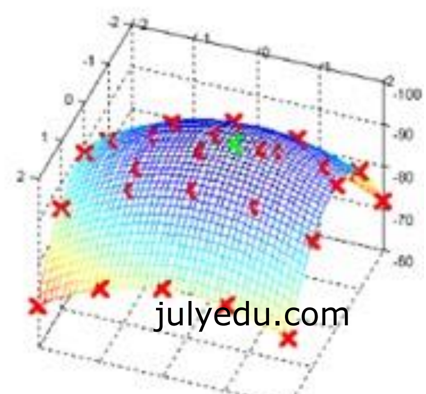






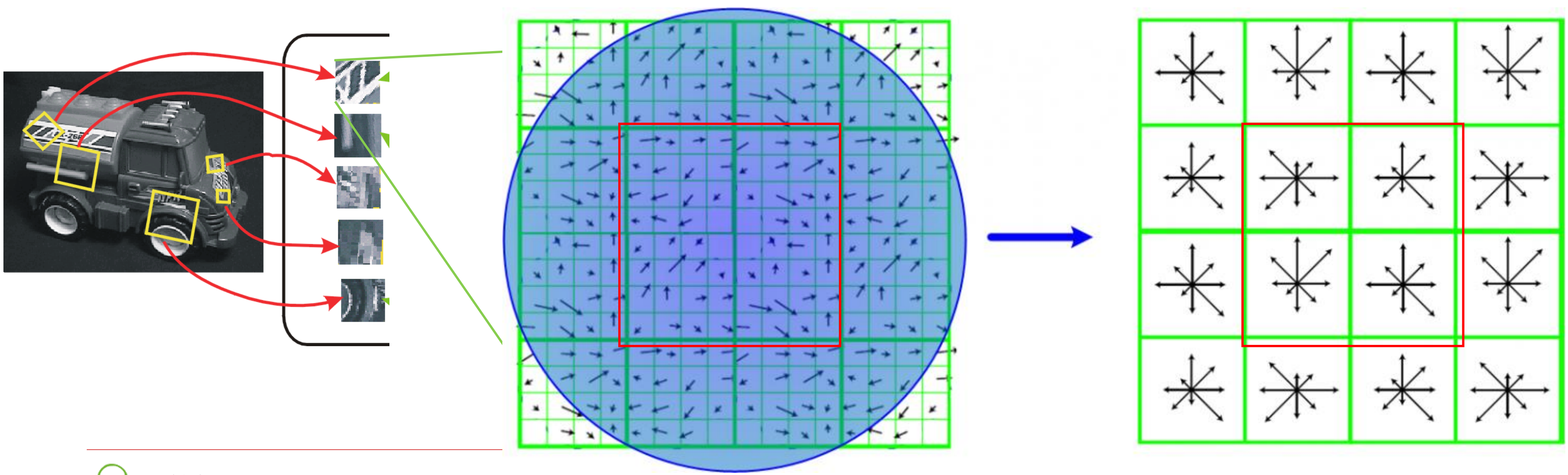


⇒ List of *(x, y, s)*



SIFT vector formation

- 4x4 array of gradient orientation histogram weighted by magnitude
- 8 orientations x 4x4 array = 128 dimensions
- Motivation: some sensitivity to spatial layout, but not too much.

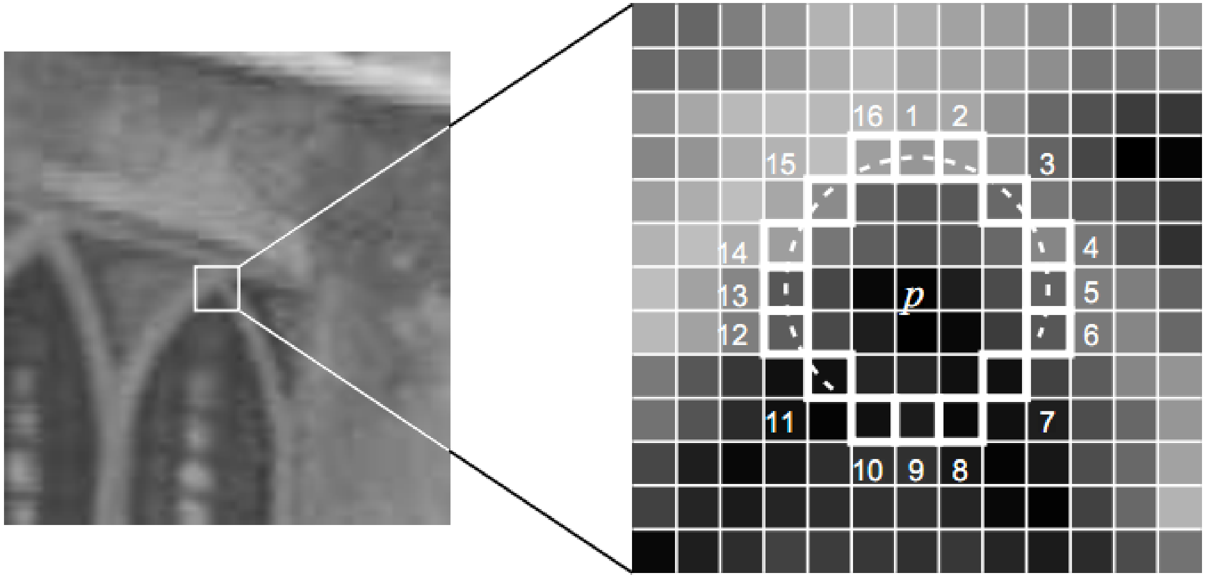


SURF

SURF Speeded Up Robust Features，号称是SIFT算法的增强版，SURF算法的计算量小，运算速度快，提取的特征点几乎与SIFT相同，由Bay 2006年提出。

	SIFT	SURF
特征点检测	用不同尺度的图片与高斯函数做卷积	用不同大小的box filter与原始图像(integral image)做卷积，易于并行
方向	特征点邻接矩形区域内，利用梯度直方图计算	特征点邻接圆域内，计算x、y方向上的Haar小波响应
描述符生成	16*16(单位为sample array)区域划分为4*4(或2*2)的子区域，每个子域计算8bin直方图	20*20(单位为sigma)区域划分为4*4子域，每个子域计算5*5个采样点的Haar小波响应，记录$\sum dx$，$\sum dy$，$\sum \lvert dx \rvert$，$\sum \lvert dy \rvert$。

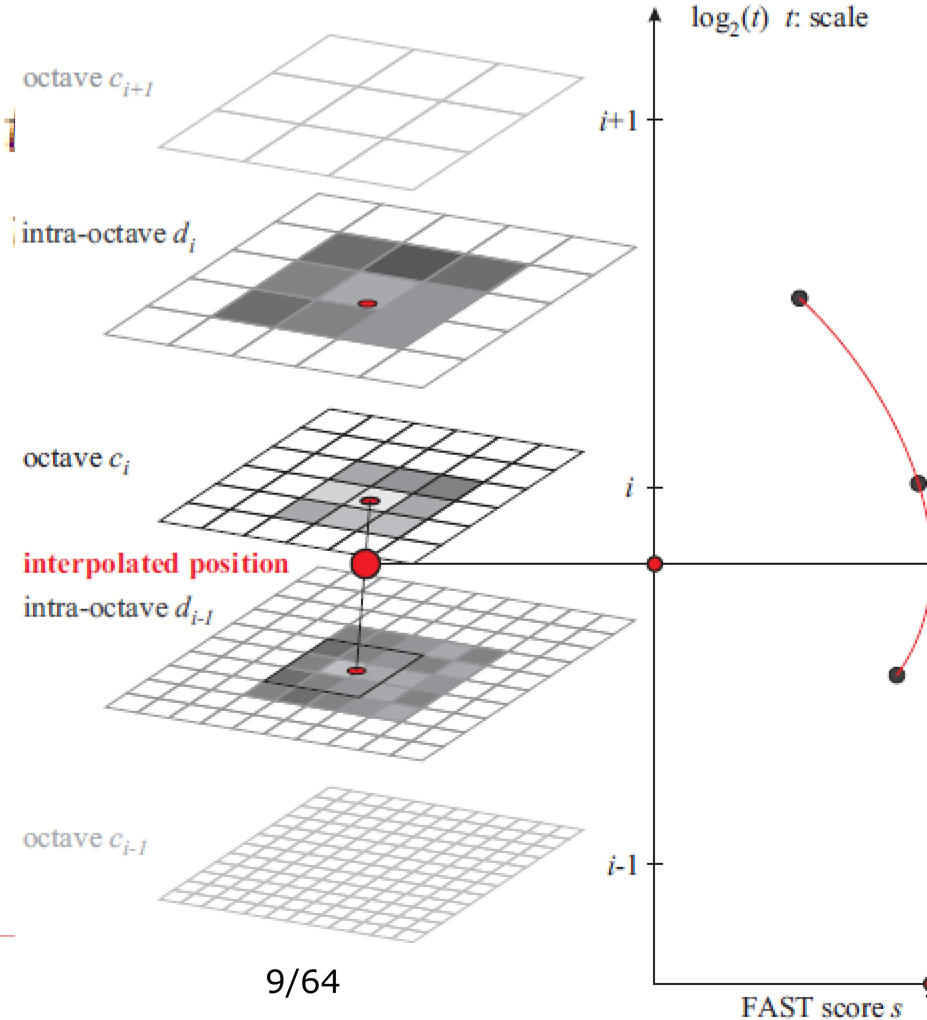
FAST Features from accelerated segment test



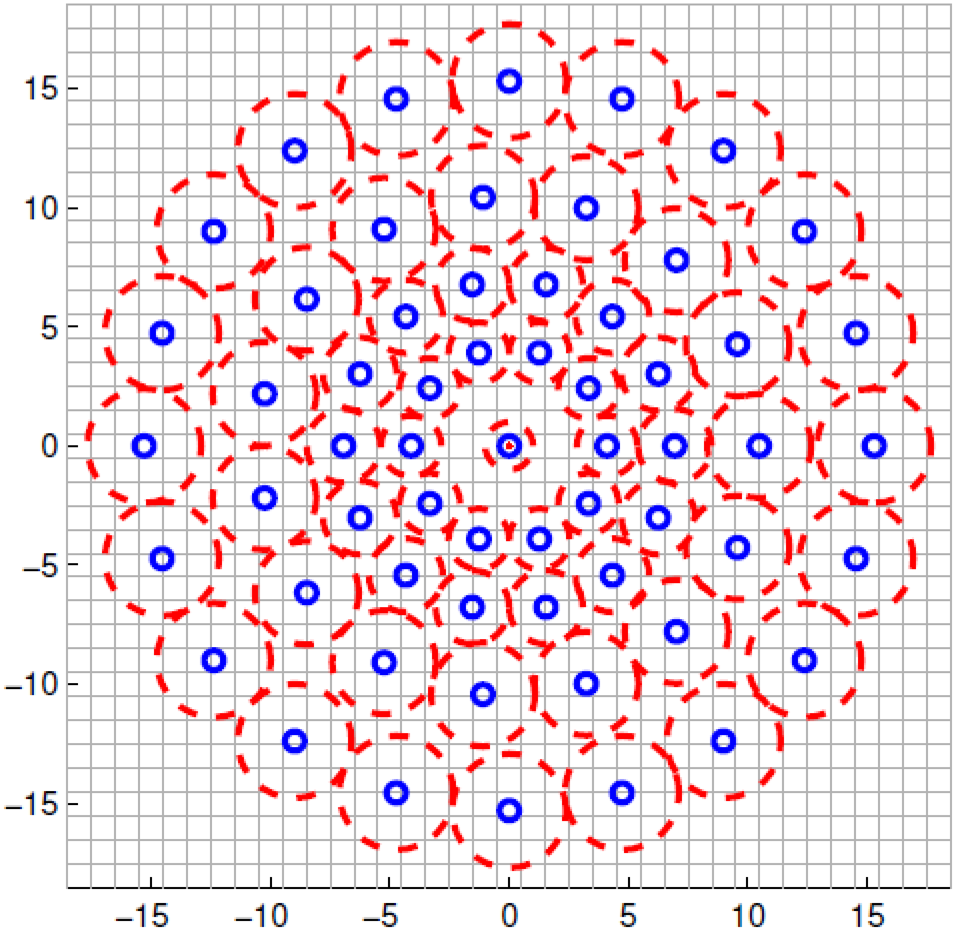

```
// vector of keypoints
std::vector<cv::KeyPoint> keypoints;
// Construction of the Fast feature detector object
cv::FastFeatureDetector fast(
        40); // threshold for detection
// feature point detection
fast.detect(image,keypoints);
```

BRISK:Binary Robust Invariant Scalable Keypoints

img	h(high)	w(width)
c0	h	w
d0	$\frac{2}{3}h$	$\frac{2}{3}h$
c1	$\frac{1}{2}h$	$\frac{1}{2}h$
d1	$\frac{1}{3}h$	$\frac{1}{3}h$
c2	$\frac{1}{4}h$	$\frac{1}{4}h$
d2	$\frac{1}{6}h$	$\frac{1}{6}h$
c3	$\frac{1}{8}h$	$\frac{1}{8}h$
d3	$\frac{1}{12}h$	$\frac{1}{12}h$



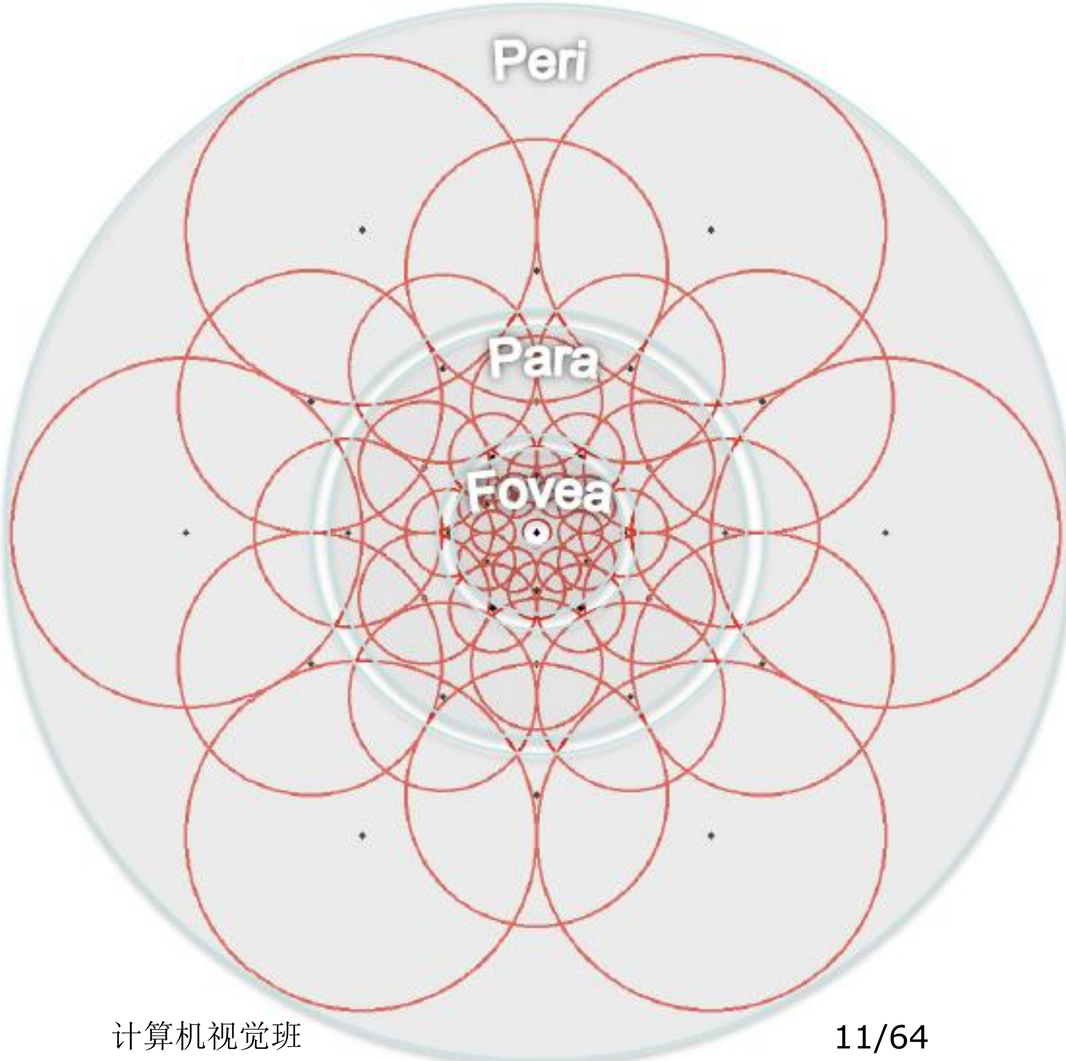
BRISK Descriptor



$$b = \begin{cases} 1, & I(\mathrm{p}_j^\alpha, \sigma_j) > I(\mathrm{p}_i^\alpha, \sigma_i) \\ 0, & \text{otherwise} \end{cases}$$

$$\forall (\mathrm{p}_i^\alpha, \mathrm{p}_j^\alpha) \in \mathcal{S}$$

FREAK: Fast Retina Keypoint



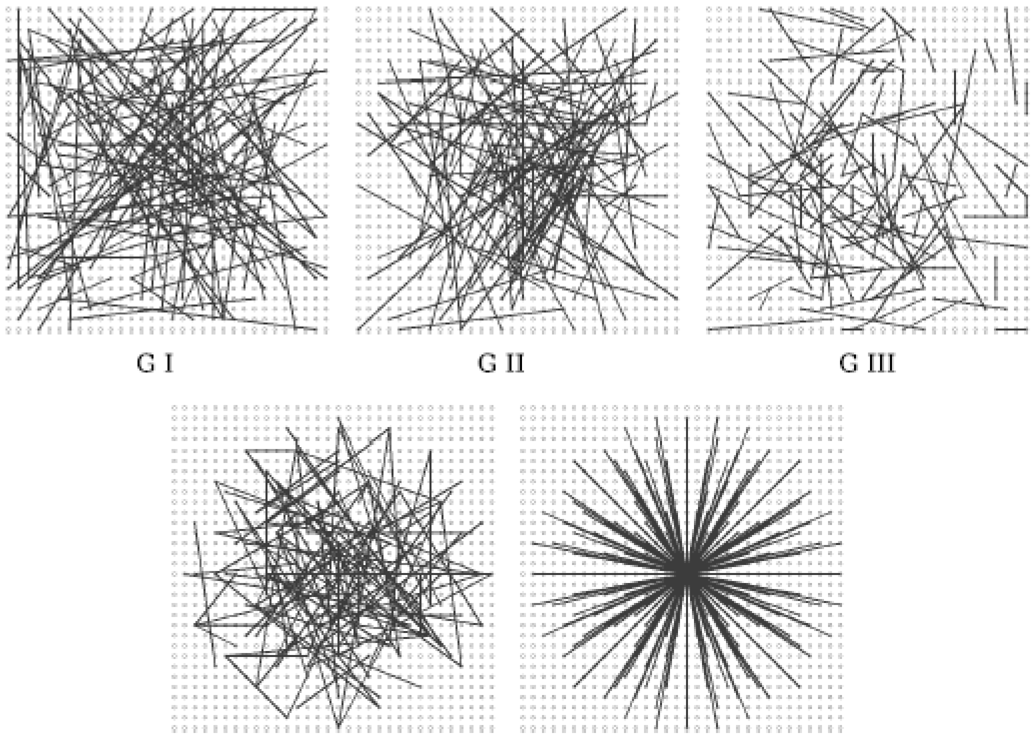
$$T(P_a) = \begin{cases} 1 & \text{if } (I(P_a^{r_1}) - I(P_a^{r_2}) > 0 \\ 0 & \text{otherwise,} \end{cases}$$

$$F = \sum_{0 \le a < N} 2^a T(P_a)$$

BRIEF Binary Robust Independent Elementary Features

- BRIEF is just a descriptor

$$\tau(p;x,y) := \begin{cases} 1 & \text{if } p(x) < p(y) \\ 0 & \text{otherwise} \end{cases}$$



G I G II G III G IV G V

ORB An efficient alternative to SIFT or SURF

- Detector

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y)$$

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right)$$

$$\theta = \text{atan2}(m_{01}, m_{10})$$

- Descriptor

$$\mathbf{S} = \begin{pmatrix} \mathbf{x}_1, \ldots, \mathbf{x}_n \\ \mathbf{y}_1, \ldots, \mathbf{y}_n \end{pmatrix}$$

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

$$\mathbf{S}_\theta = \mathbf{R}_\theta \mathbf{S}$$

$$\tau(\mathbf{p}; \mathbf{x}, \mathbf{y}) := \begin{cases} 1 & : \mathbf{p}(\mathbf{x}) < \mathbf{p}(\mathbf{y}) \\ 0 & : \mathbf{p}(\mathbf{x}) \geq \mathbf{p}(\mathbf{y}) \end{cases}$$

$$f_n(\mathbf{p}) := \sum_{1 \leq i \leq n} 2^{i-1} \tau(\mathbf{p}; \mathbf{x}_i, \mathbf{y}_i)$$

topics

- 成像：相机几何模型
- 坐标系统转换（2D-2D，2D-3D）
- 相机标定
- 计算两幅图像的投影关系

Image formation – (bad) method

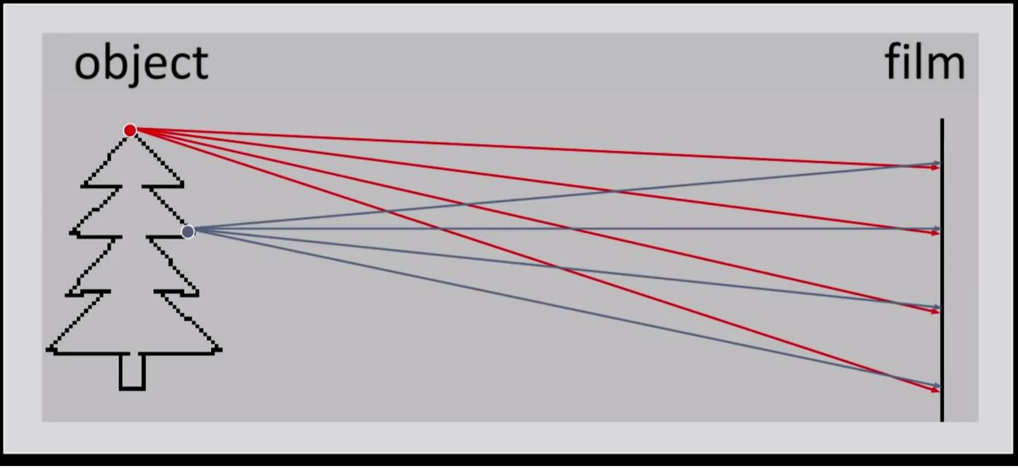
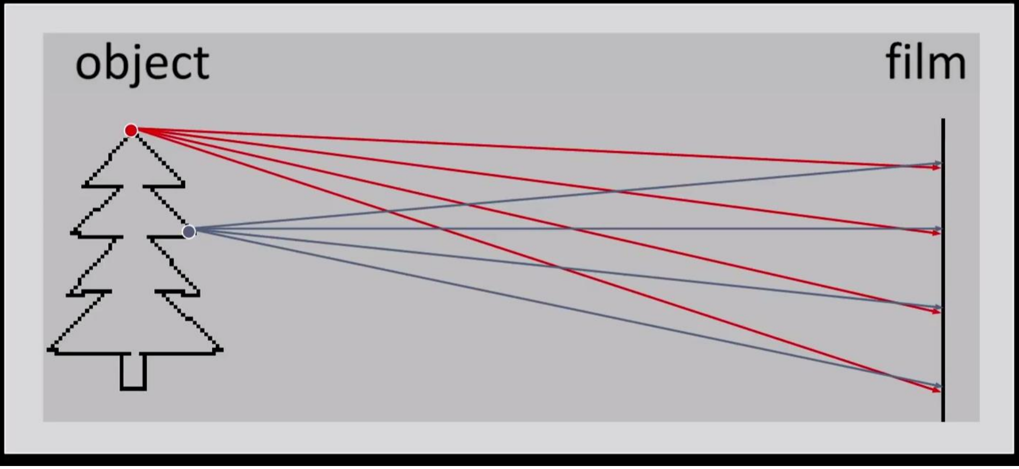
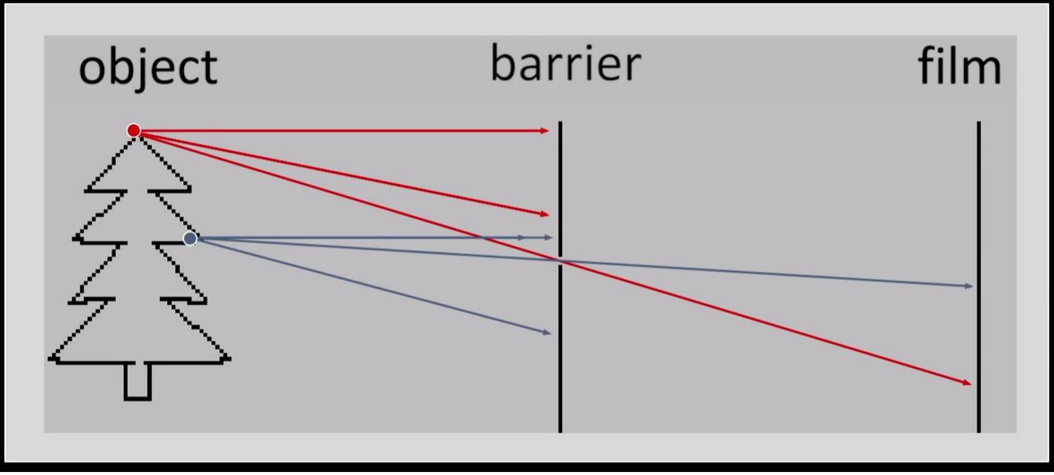


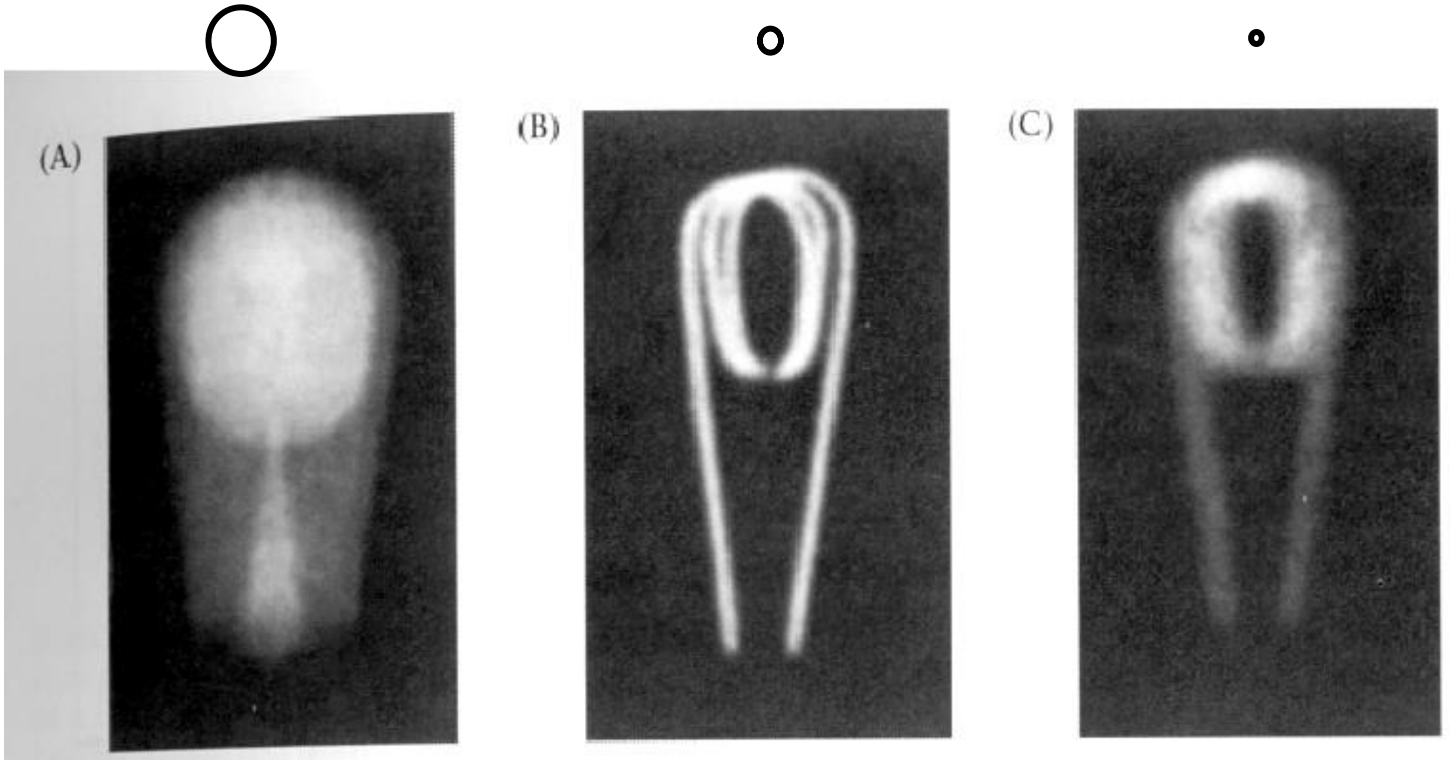
Image formation – (bad) method



Pinhole camera

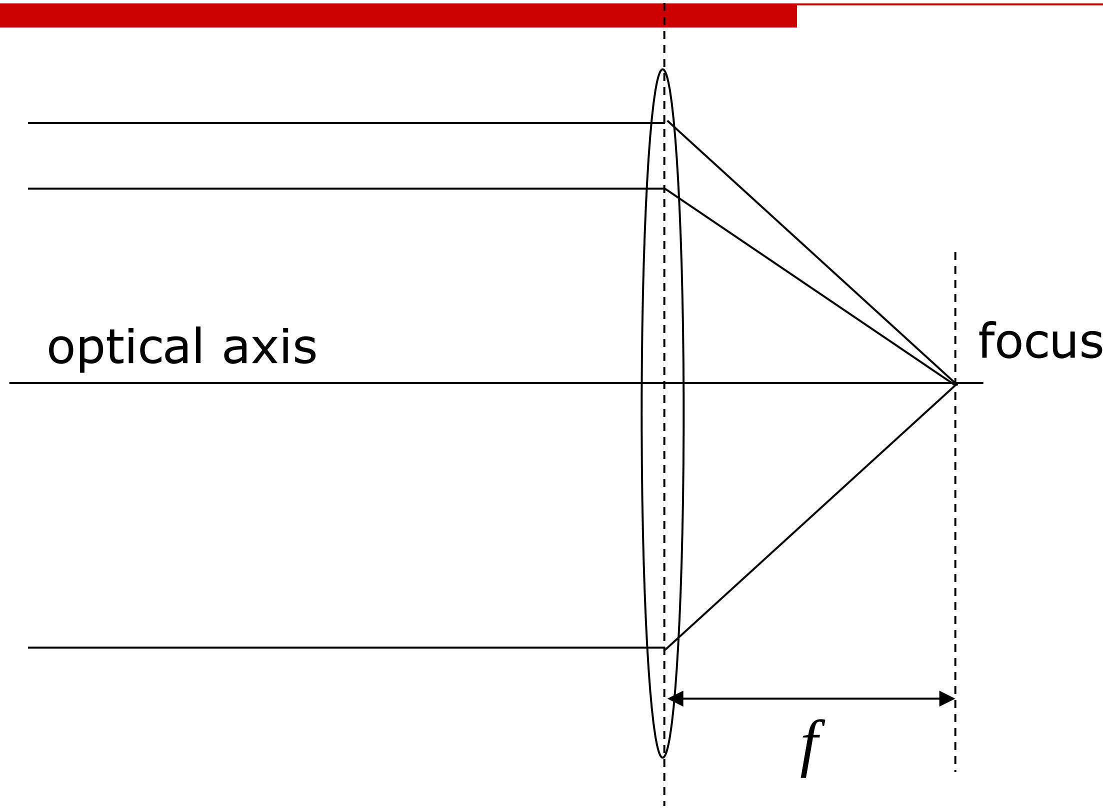


Limits for pinhole cameras



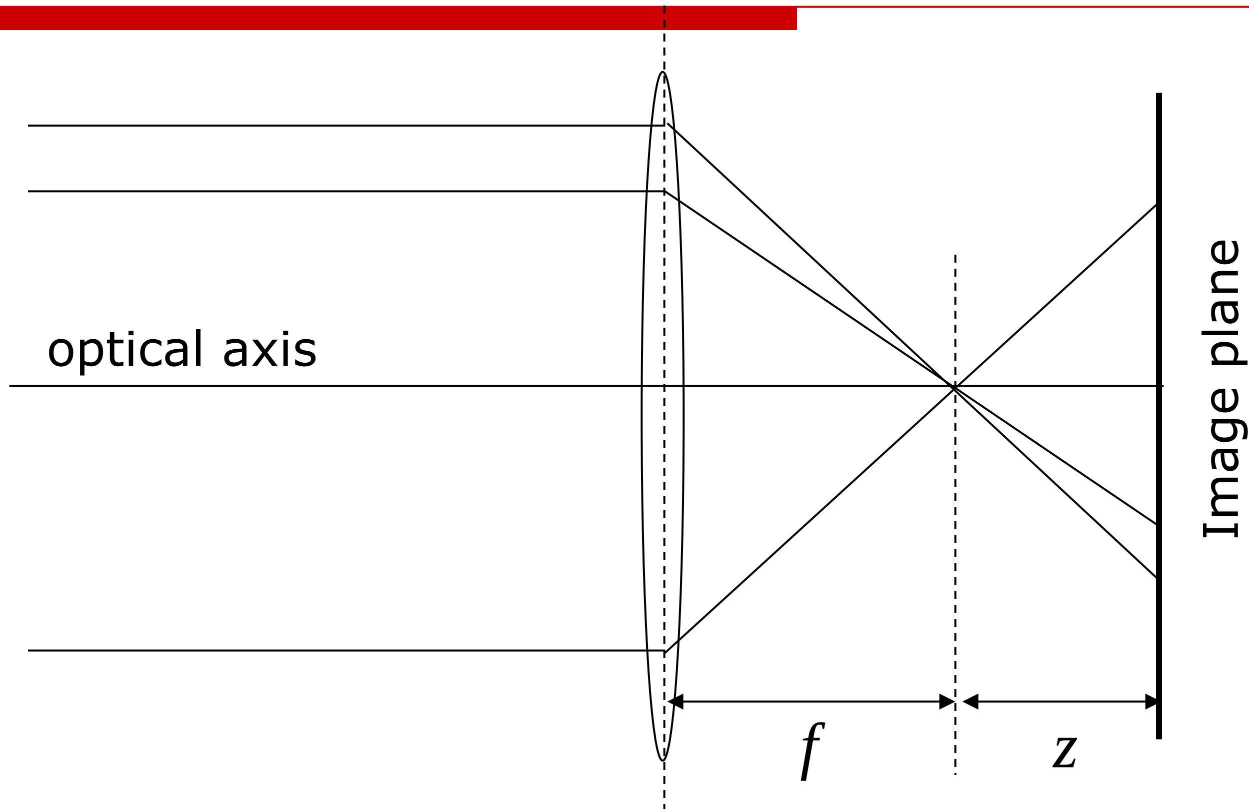
2.18 DIFFRACTION LIMITS THE QUALITY OF PINHOLE OPTICS. These three images of a bulb filament were made using pinholes with decreasing size. (A) When the pinhole is relatively large, the image rays are not properly converged, and the image is blurred. (B) Reducing the size of the pinhole improves the focus. (C) Reducing the size of the pinhole further worsens the focus, due to diffraction. From Ruechardt, 1958.

Thin Lens: Definition



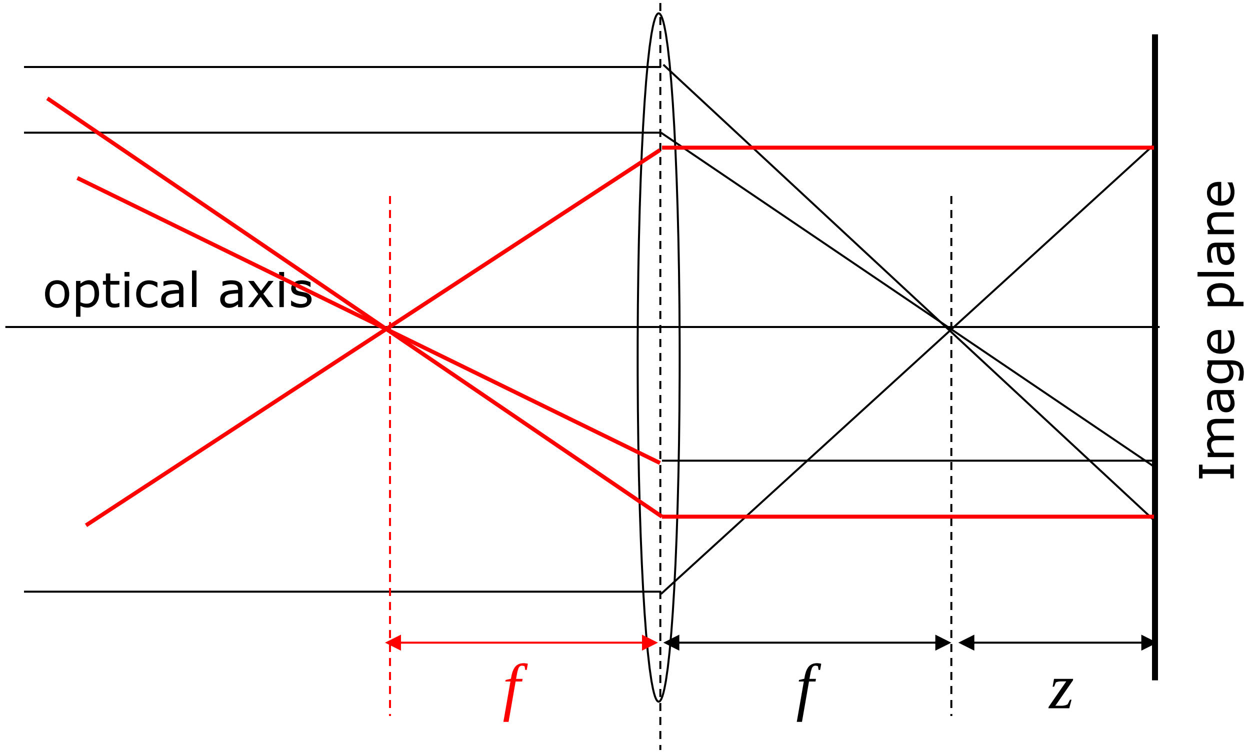
Spherical lens surface: Parallel rays are refracted to single point

Thin Lens: Projection



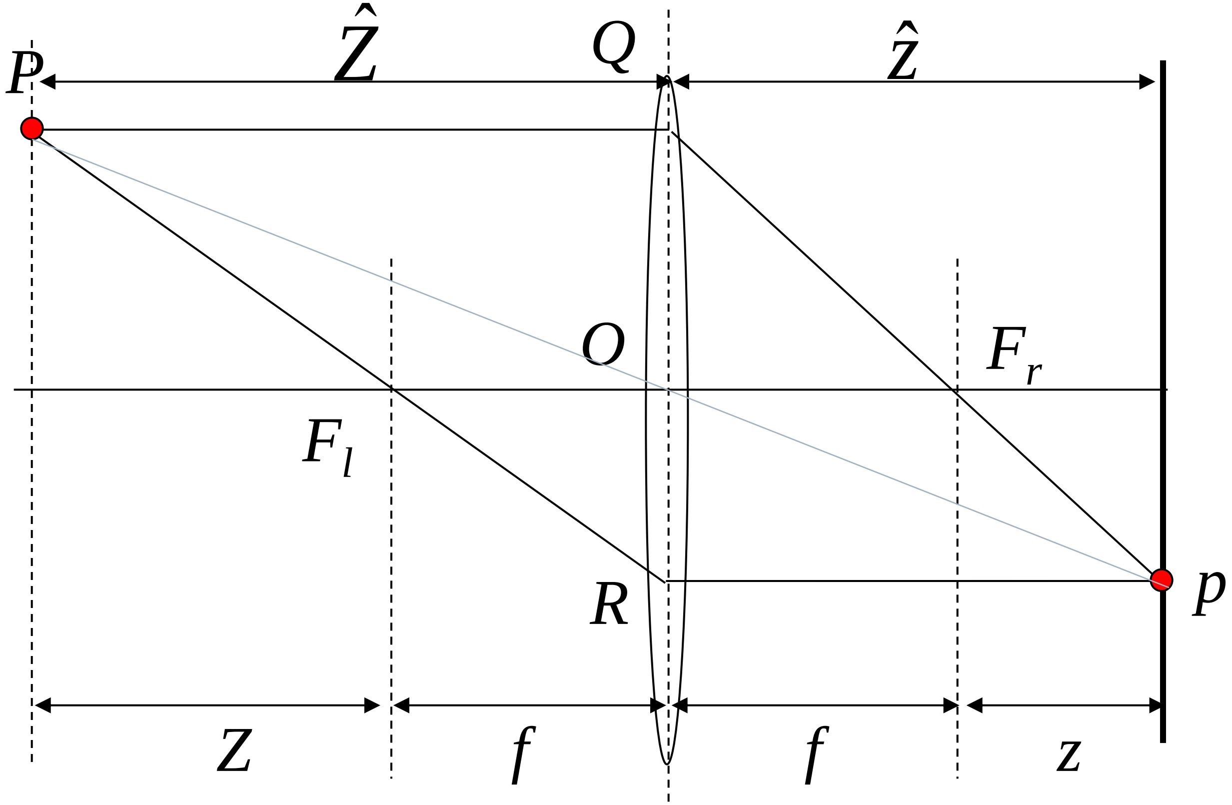
Spherical lens surface: Parallel rays are refracted to single point

Thin Lens: Projection



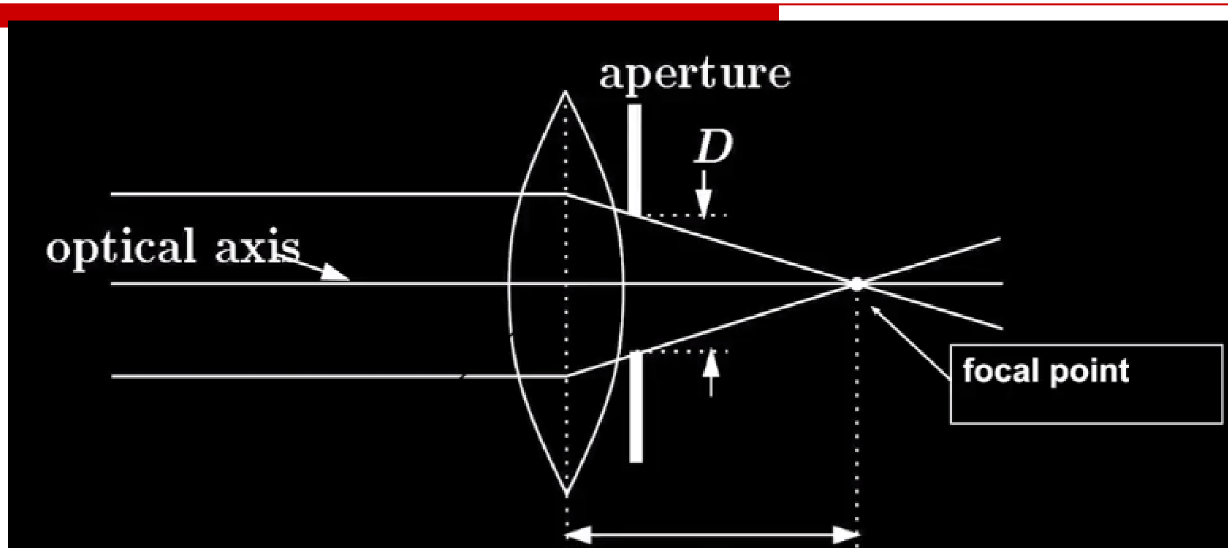
Spherical lens surface: Parallel rays are refracted to single point

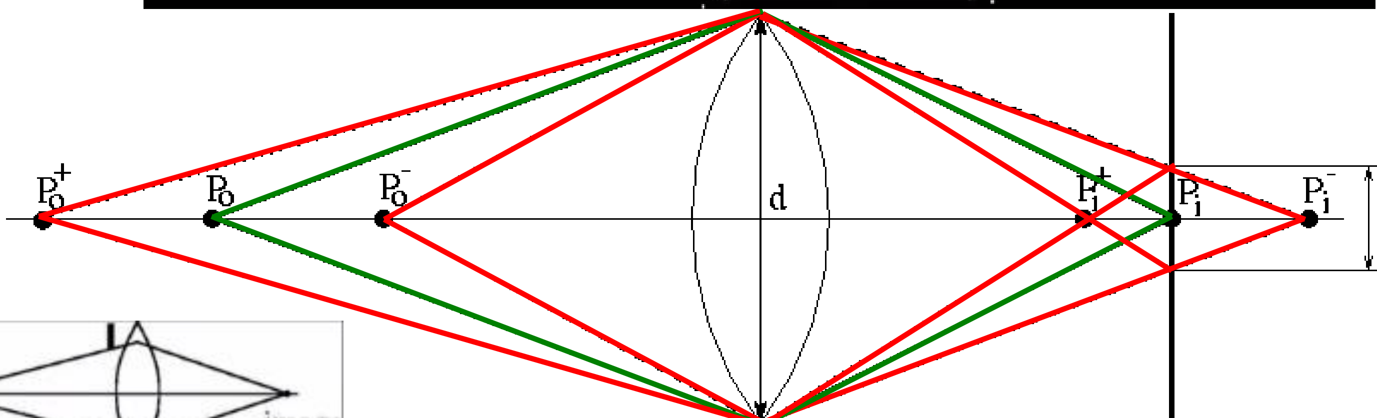
The Thin Lens Law

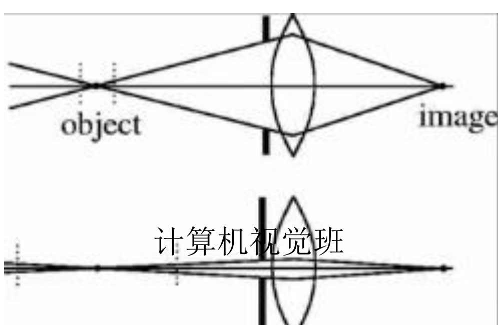


$$\frac{1}{\hat{z}} + \frac{1}{\hat{Z}} = \frac{1}{f}$$

The depth-of-field

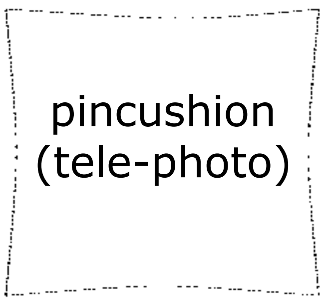


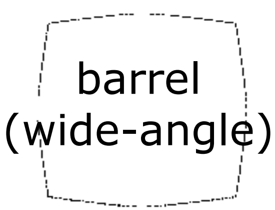


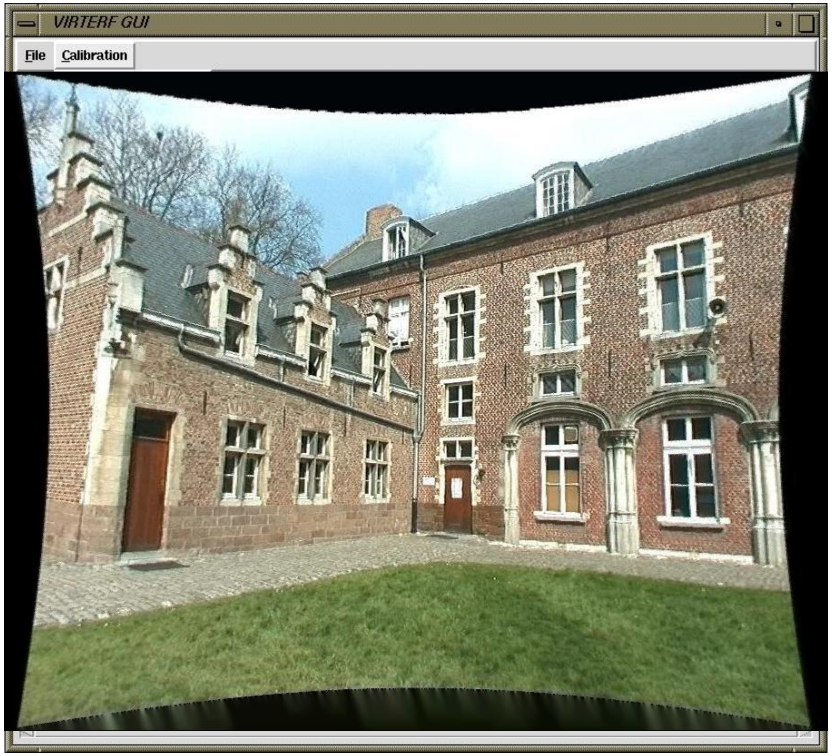


Distortion

magnification/focal length different for different angles of inclination







Marc Pollefeys

Chromatic Aberration

rays of different wavelengths focused in different planes

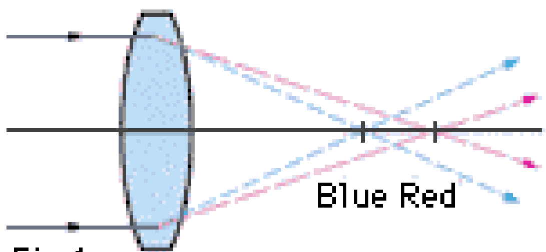


Fig.1
Axial chromatic aderration

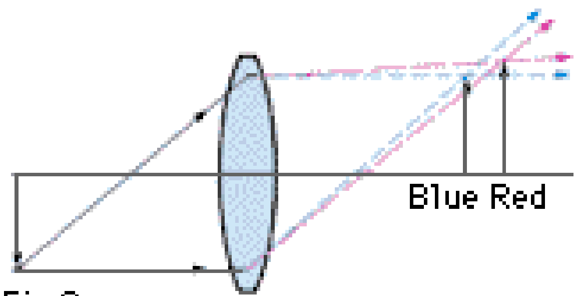


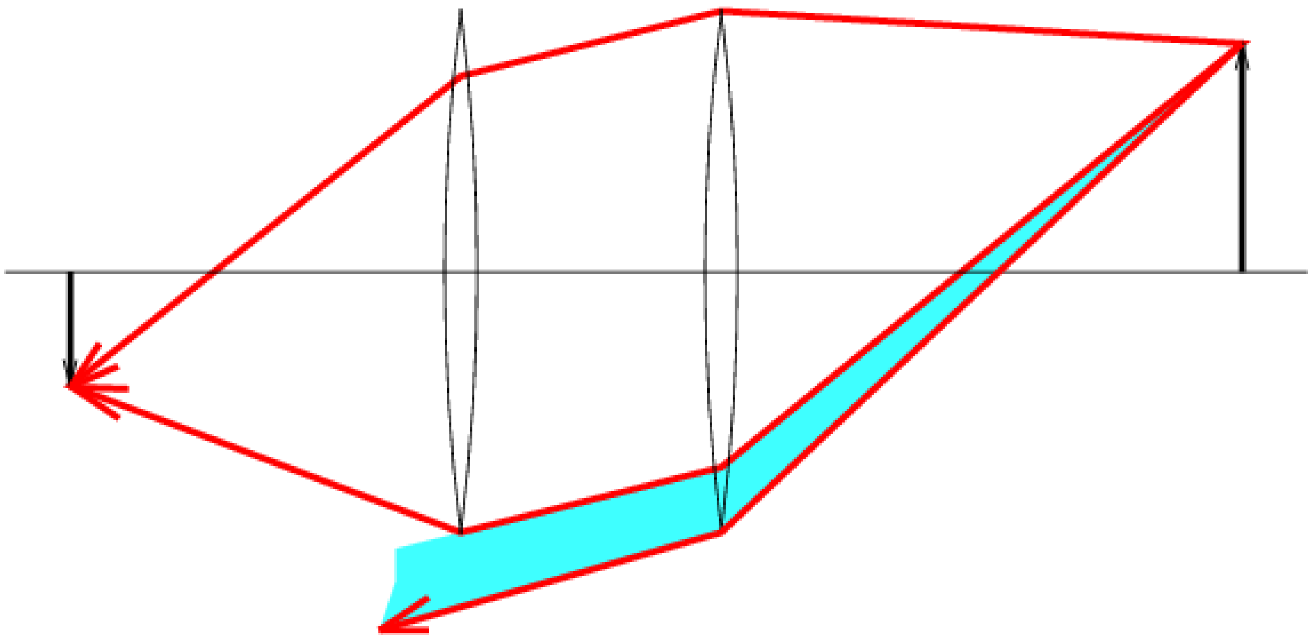
Fig.2
Magnification chromatic aderration

cannot be removed completely

The image is blurred and appears colored at the fringe.

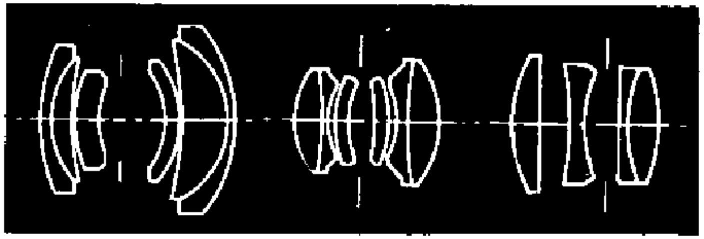
Marc Pollefeys

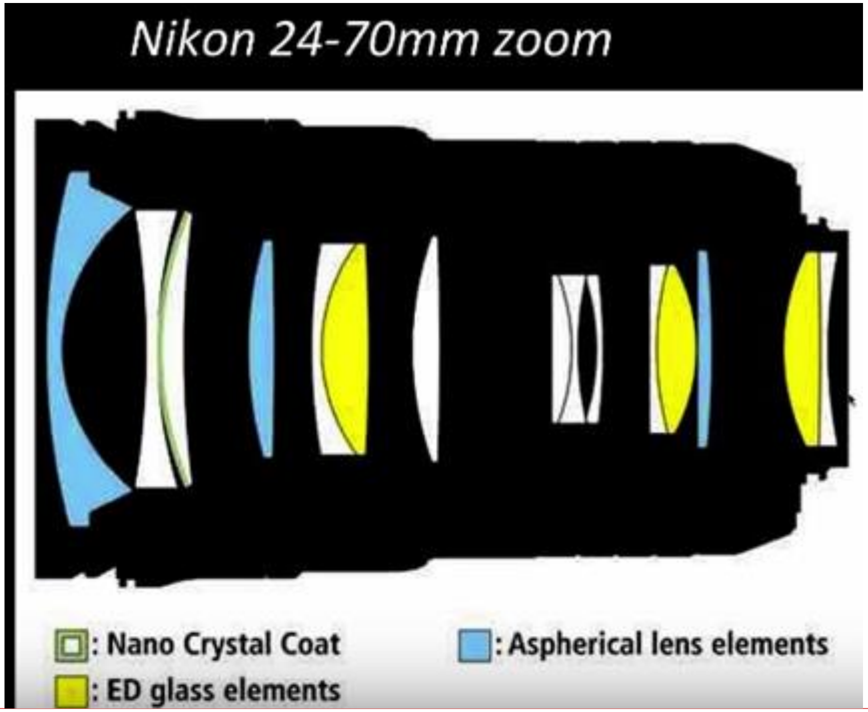
Vignetting



Effect: Darkens pixels near the image boundary

Solutions




Nikon 24-70mm zoom
: Nano Crystal Coat
: Aspherical lens elements
: ED glass elements

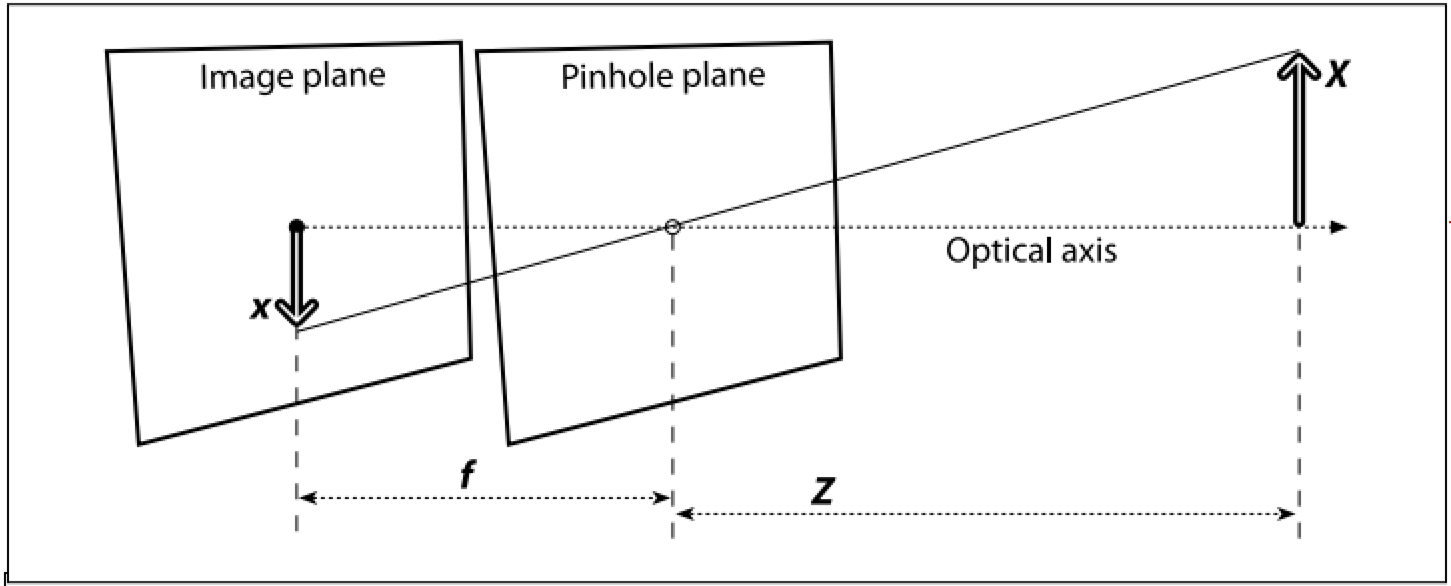
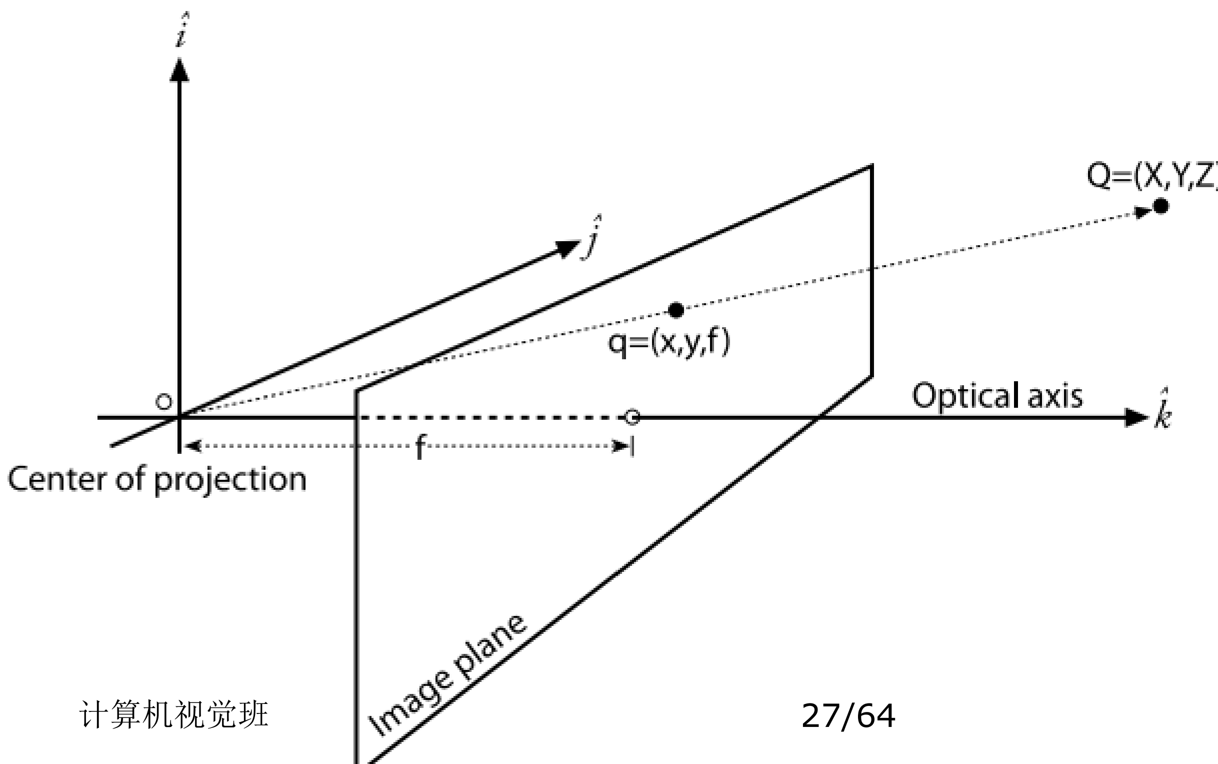
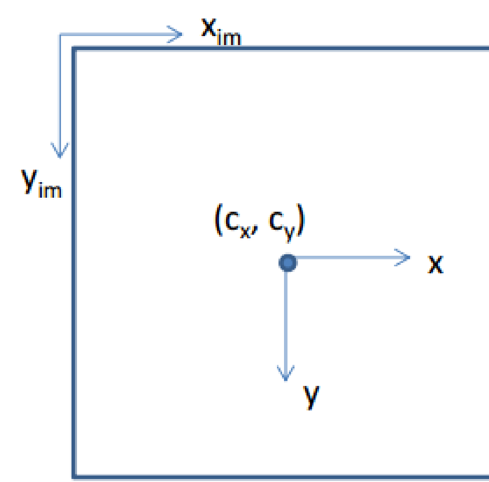


Image plane
Pinhole plane
X
Optical axis
x
f
Z



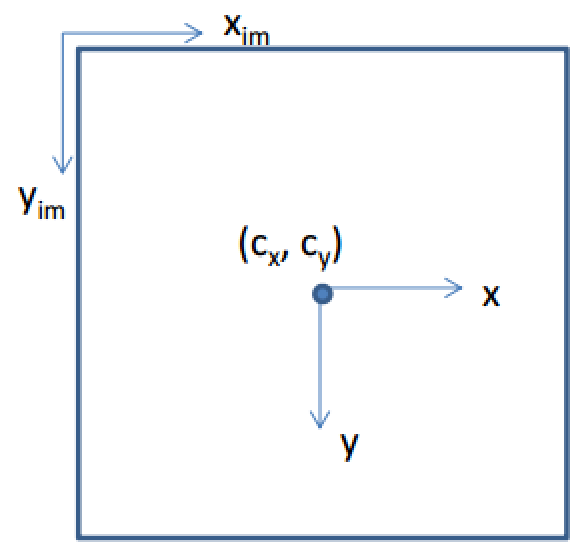
$\hat{i}$
$\hat{j}$
Q=(X,Y,Z)
q=(x,y,f)
O
Optical axis
$\hat{k}$
f
Center of projection
Image plane



x_{im}
y_{im}
(c_x, c_y)
x
y

Conversion between real image and pixel image coordinates

- Assume
  - The image center (principal point) is located at pixel (c_x, c_y) in the pixel image
  - The spacing of the pixels is (s_x, s_y) in millimeters

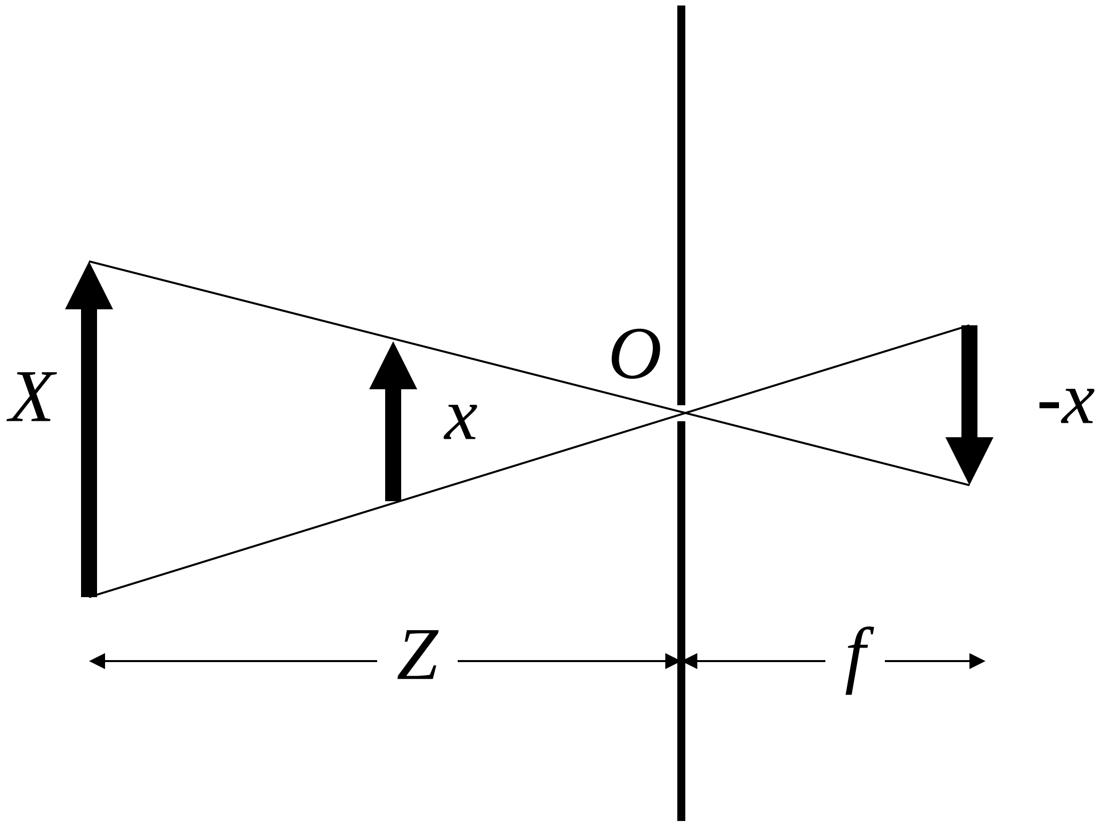


- Then

$$x = (x_{im} - c_x)\, s_x \qquad x_{im} = x/s_x + c_x$$
$$y = (y_{im} - c_y)\, s_y \qquad y_{im} = y/s_y + c_y$$

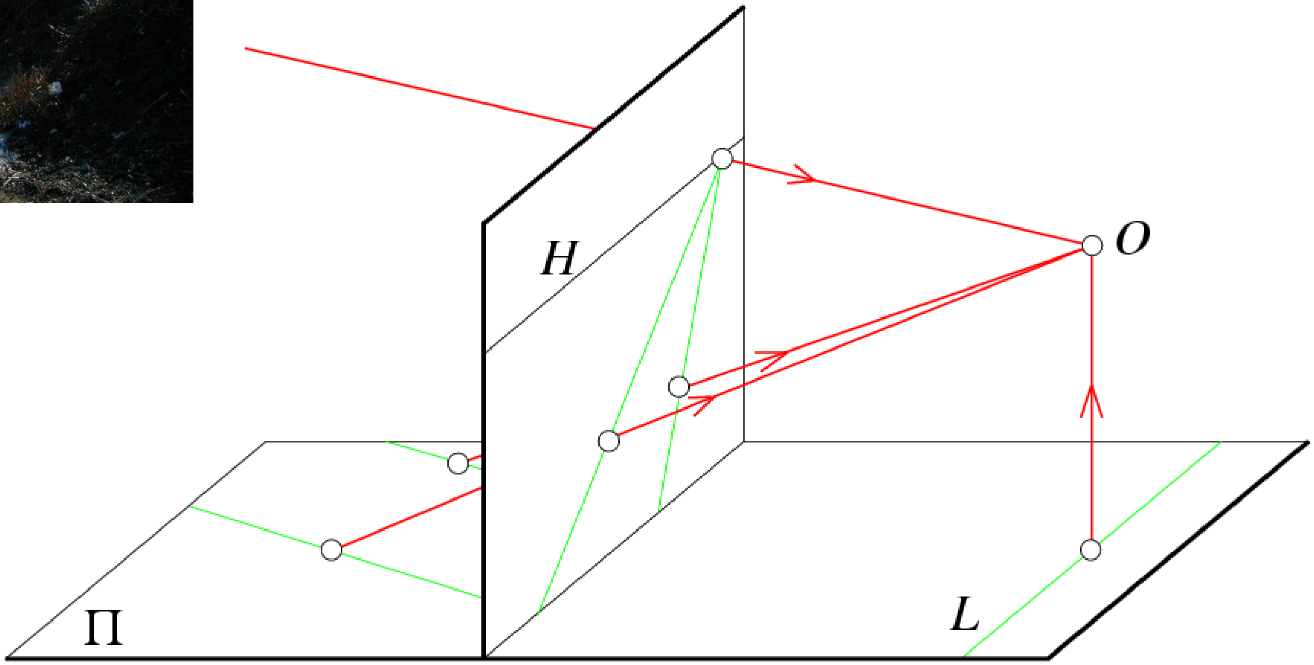
$$x_{\text{screen}} = f_x\left(\frac{X}{Z}\right) + c_x, \qquad y_{\text{screen}} = f_y\left(\frac{Y}{Z}\right) + c_y$$

Perspective Projection

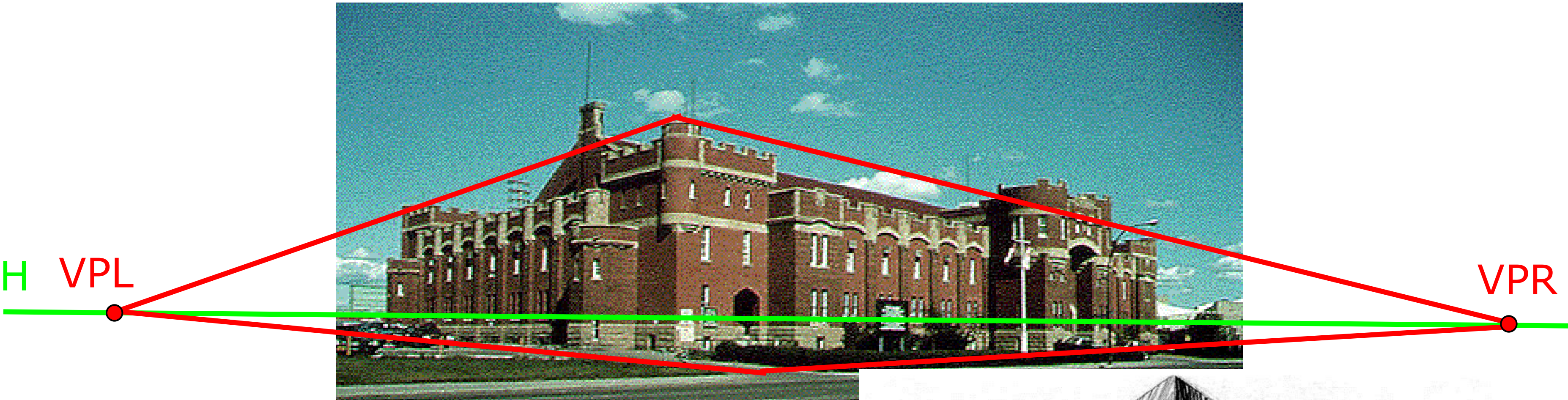


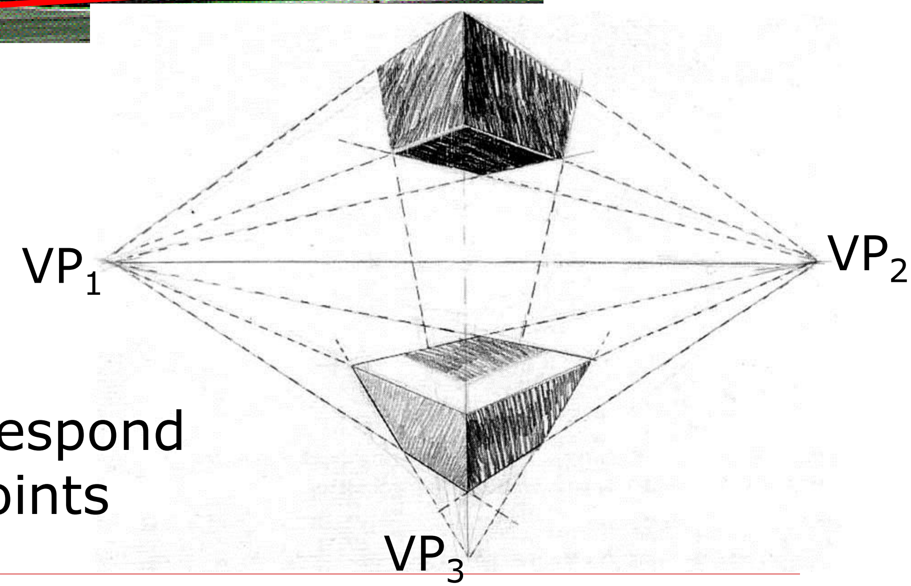
$$x = X\frac{f}{Z}$$

Geometric properties of projection



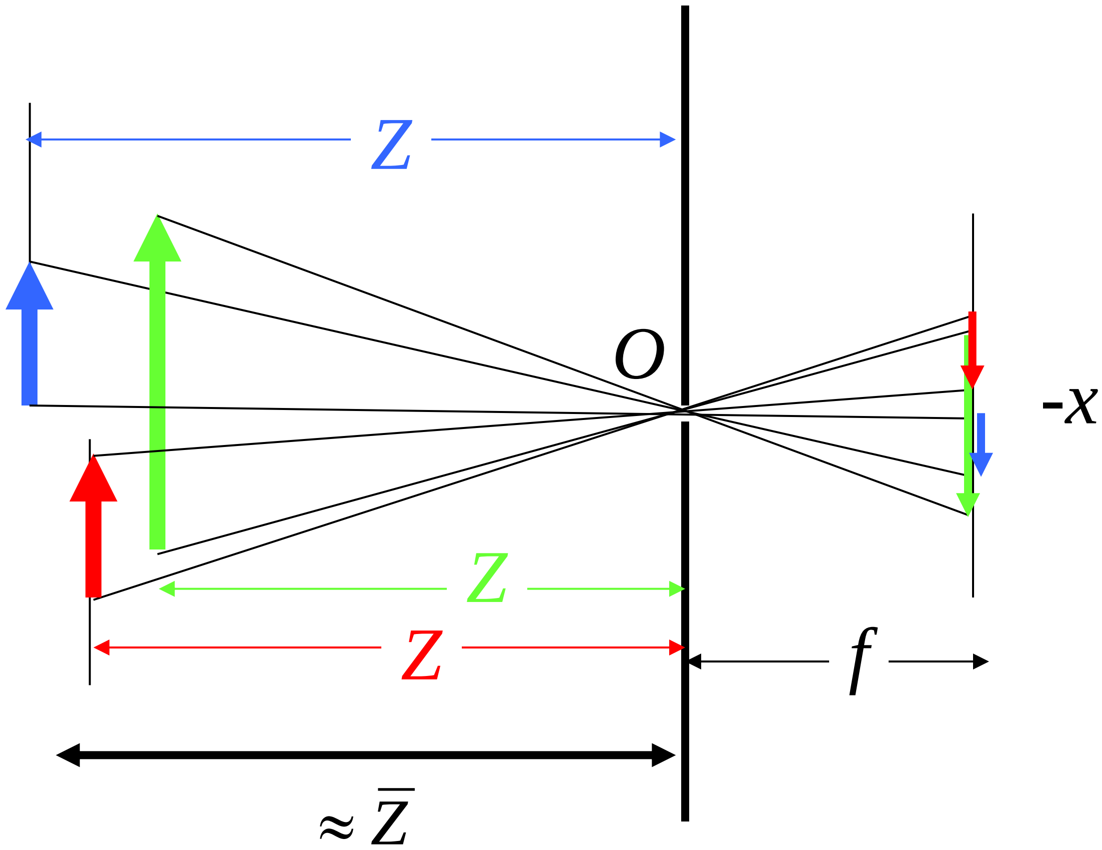
Vanishing points





Different directions correspond
to different vanishing points

Weak Perspective Projection



$$x \approx X \frac{f}{\bar{Z}} = const \cdot X$$

Camera Parameters

- Intrinsic parameters
  - Those parameters needed to relate an image point (in pixels) to a direction in the camera frame
  - f_x, f_y, c_x, c_y
  - Also lens distortion parameters (will discuss later)

- Extrinsic parameters
  - Define the position and orientation (pose) of the camera in the world

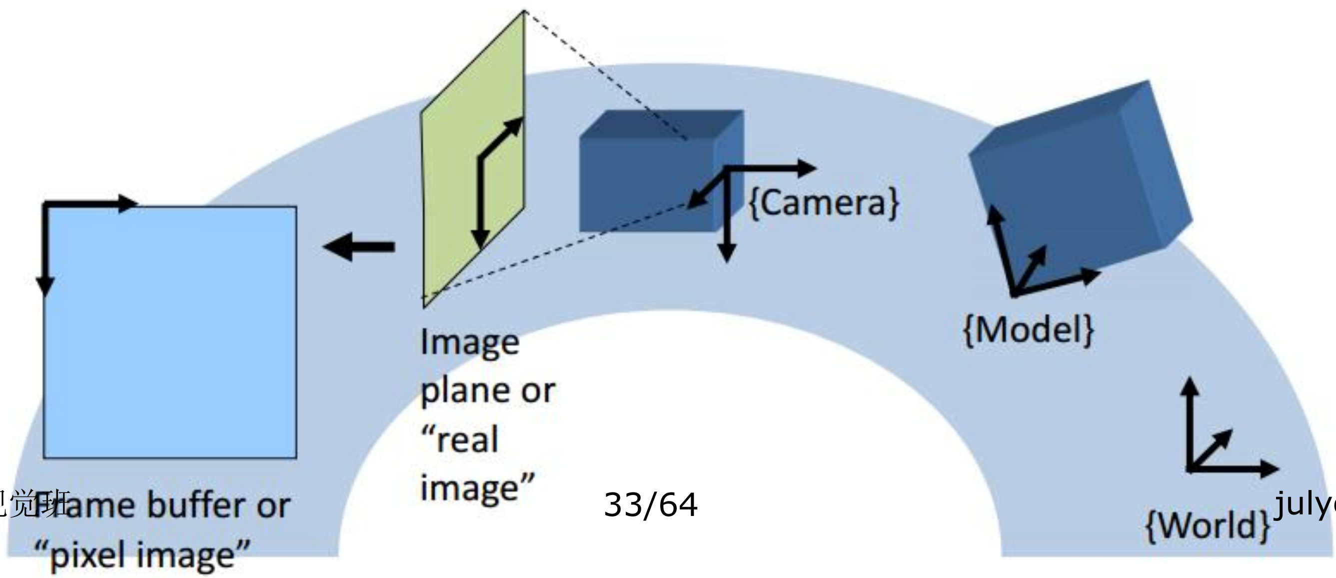


Image to image projection

- Homogeneous coordinate
- 2D/3D transformations

Homogeneous coordinates

Trick: add one more coordinate:

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

homogeneous image (2D) coordinates

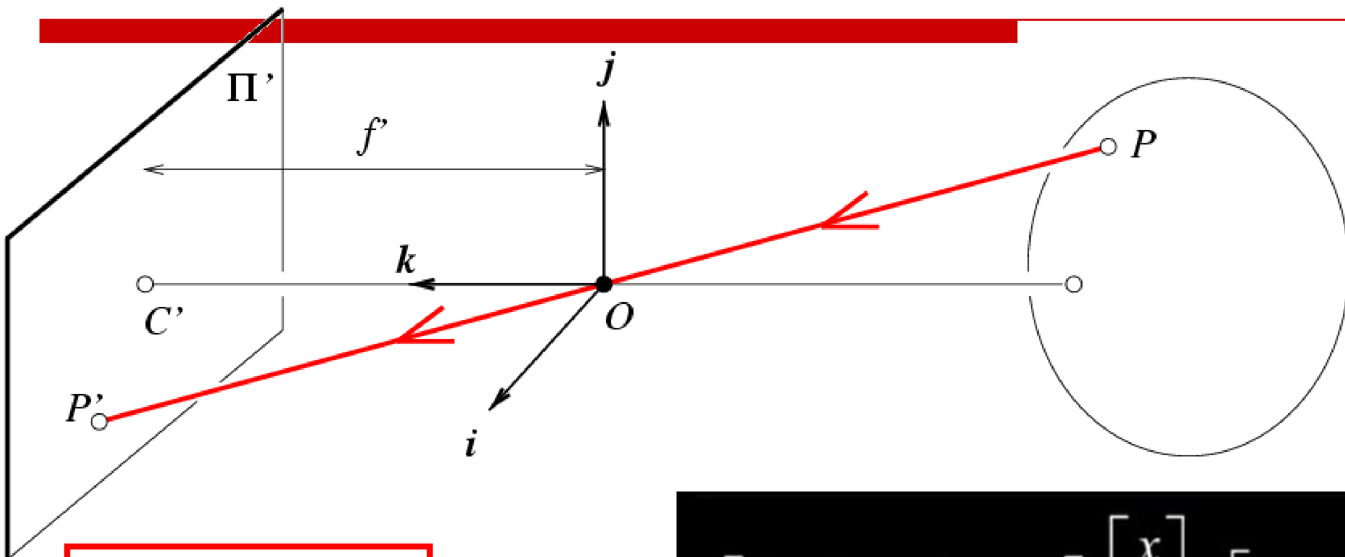
homogeneous scene (3D) coordinates

Converting *from* homogeneous coordinates:

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

(this makes homogenous coordinates invariant under scale)

Perspective projection

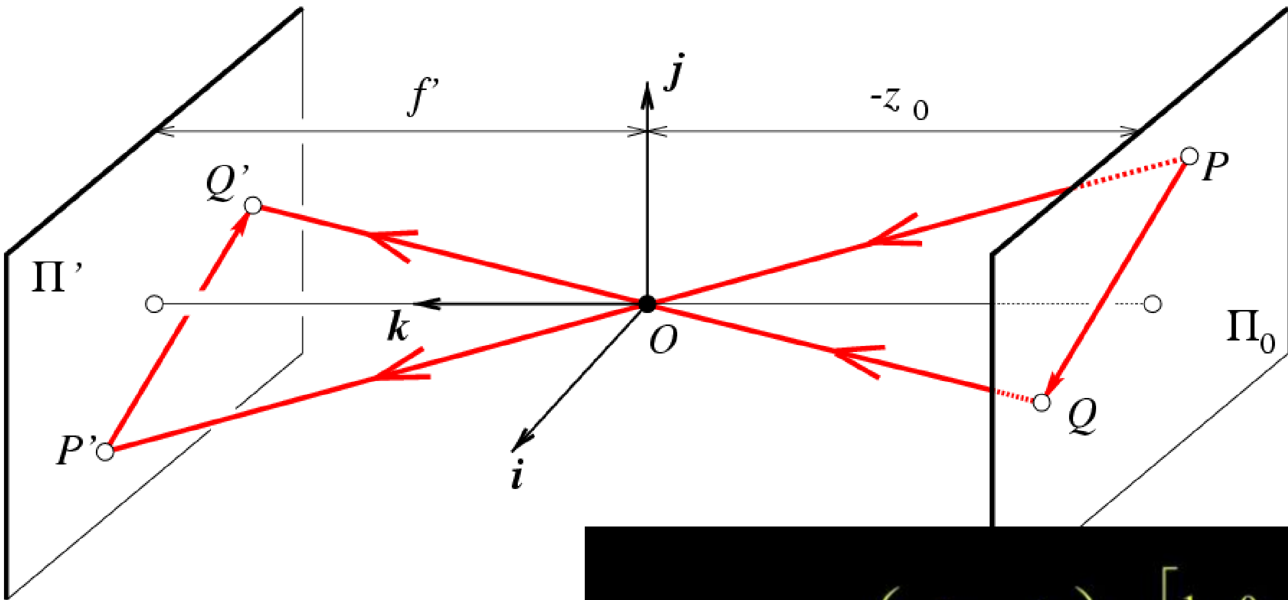


$$\begin{cases} x' = f'\dfrac{x}{z} \\ y' = f'\dfrac{y}{z} \end{cases}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z/f \end{bmatrix} \Rightarrow \left(f\frac{x}{z}, f\frac{y}{z} \right)$$

$$\Rightarrow (u, v)$$

Weak perspective projection

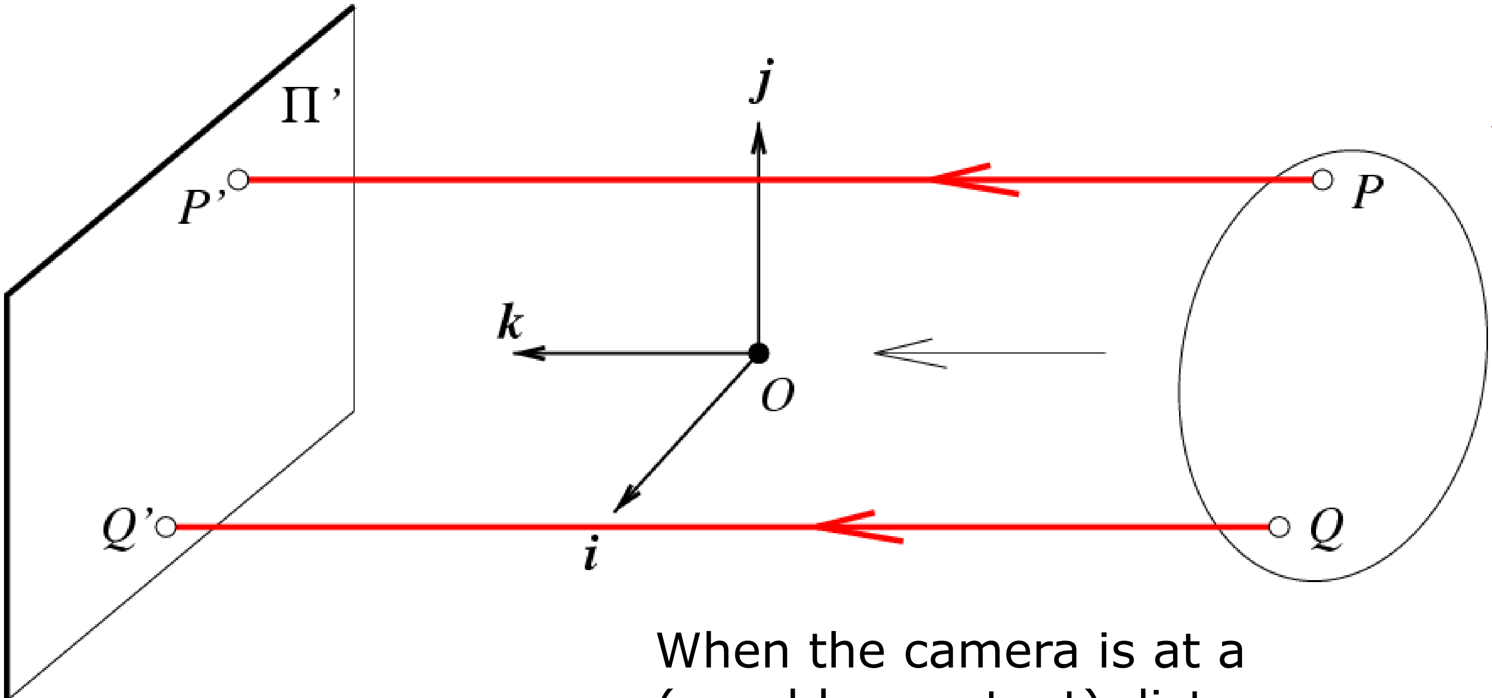


$$(x,y,z) \rightarrow \left(\frac{fx}{z_0}, \frac{fy}{z_0}\right) \qquad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/s \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1/s \end{bmatrix} \Rightarrow (sx, sy)$$

$$\boxed{\begin{cases} x' = -mx \\ y' = -my \end{cases}}$$ where $m = -\dfrac{f'}{z_0}$ is the magnification.

When the scene relief is small compared its distance from the Camera, m can be taken constant: weak perspective projection.

Orthographic projection

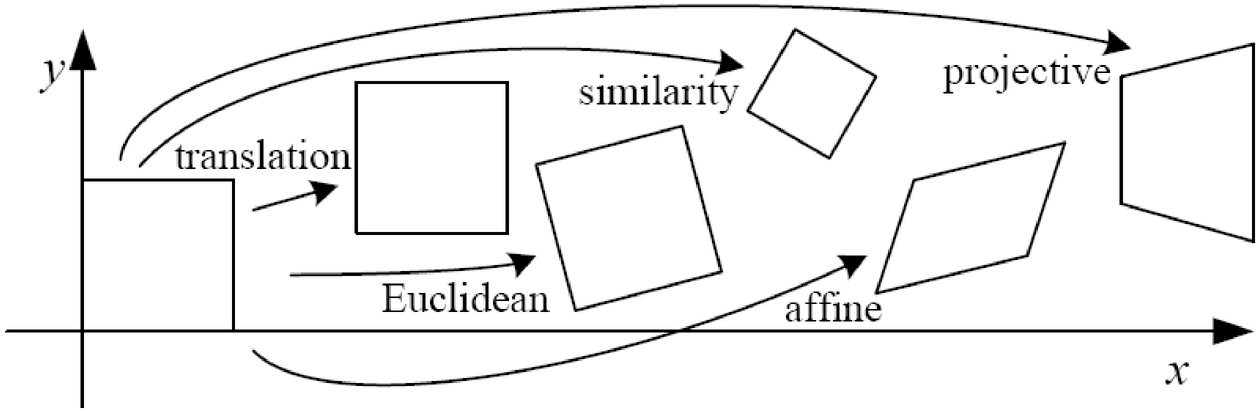


When the camera is at a (roughly constant) distance from the scene, take m=1.

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

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow (x, y)$$

2D image transformations (reference table)



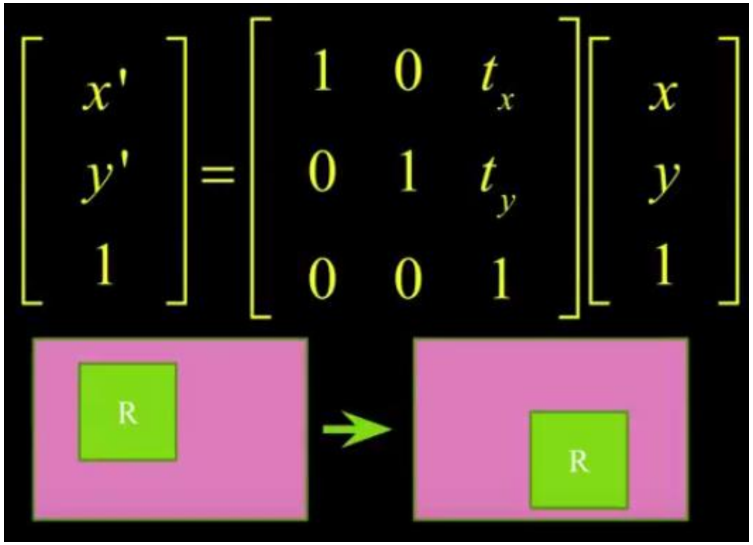
Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	3	lengths $+\cdots$	
similarity	$\begin{bmatrix} s\boldsymbol{R}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

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

Special projective transformations

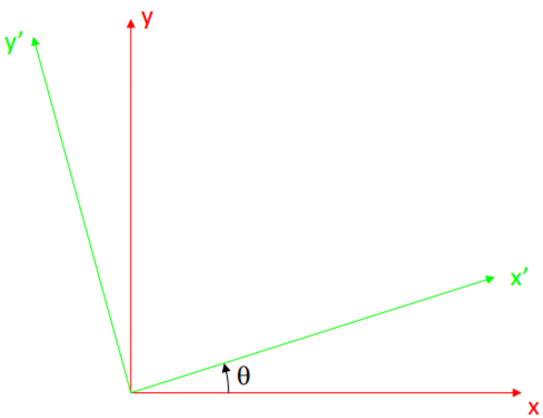
- Translation

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



Special projective transformations

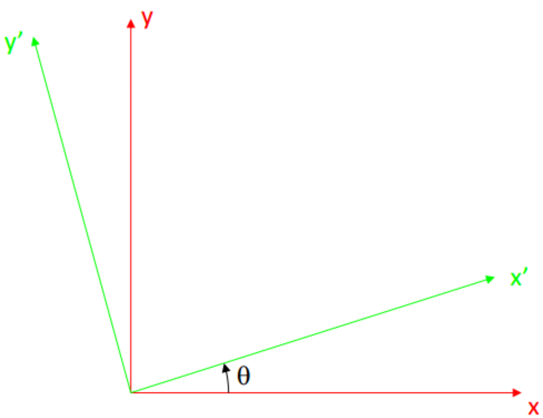
- Euclidean（Rigid body）



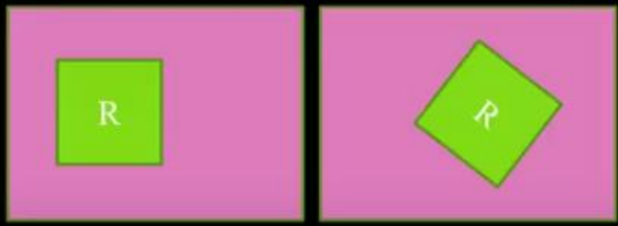
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & t_x \\ \sin(\theta) & \cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Special projective transformations

- Euclidean（Rigid body）



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



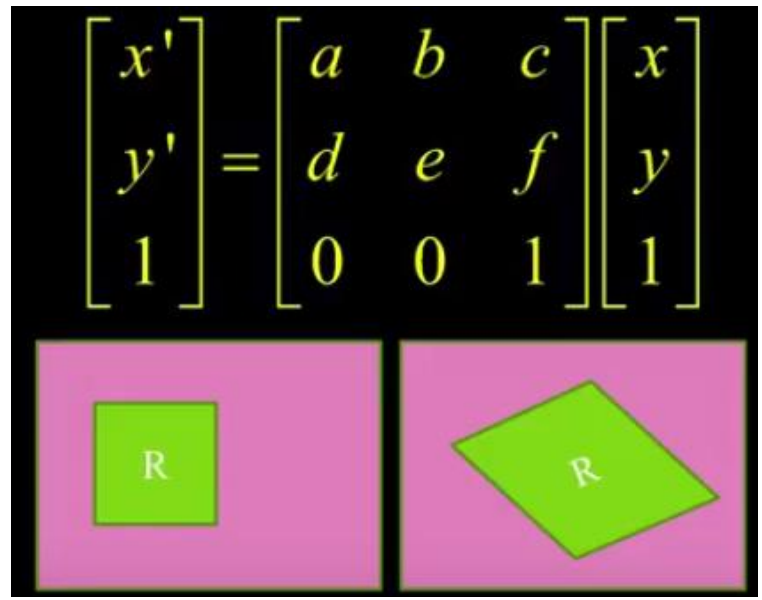
- Similarity transform

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a\cos(\theta) & -a\sin(\theta) & t_x \\ a\sin(\theta) & a\cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Special projective transformations

- Affine transform

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



Projective Transformations

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

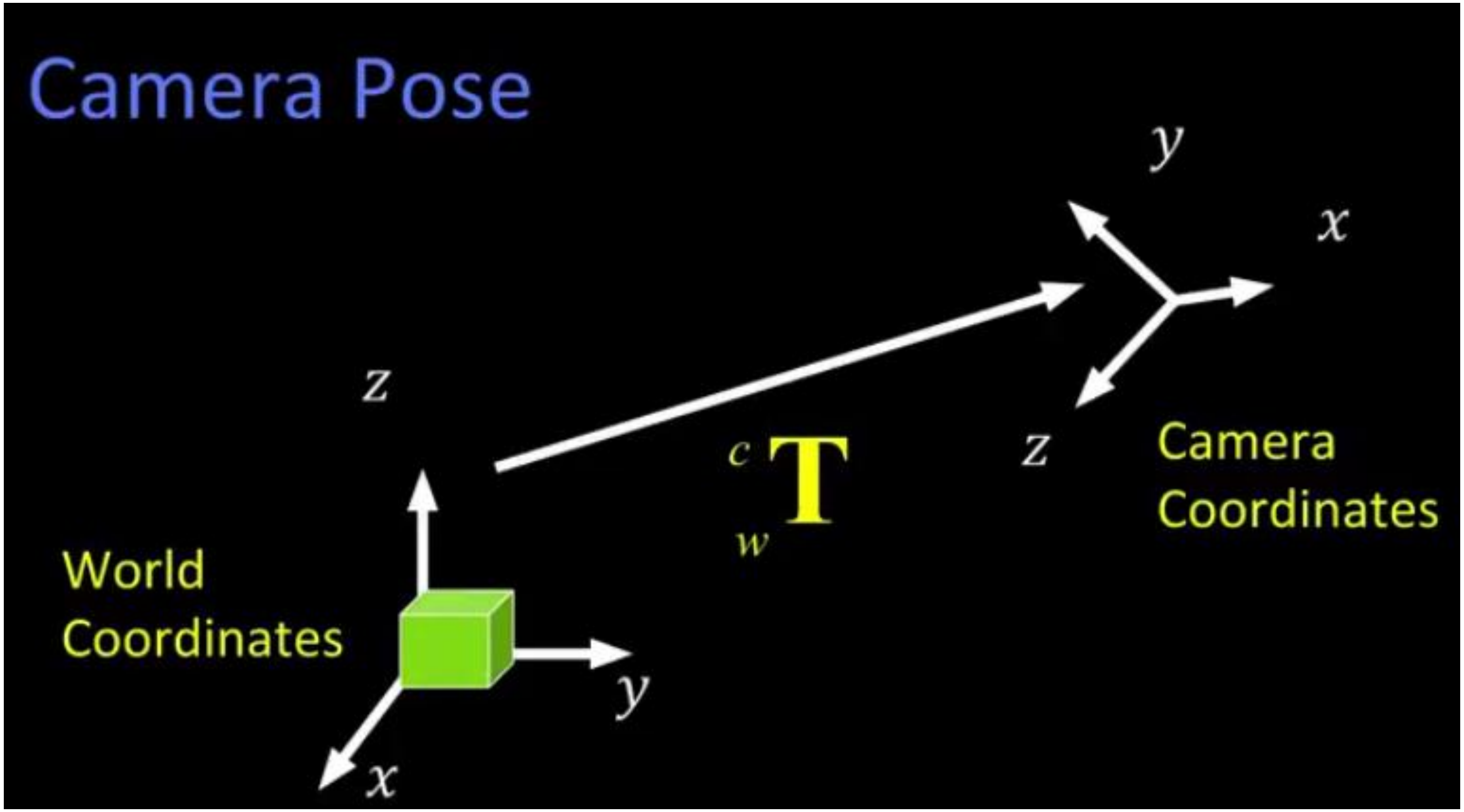
Projective transformations

- Affine transformations, and
- Projective warps

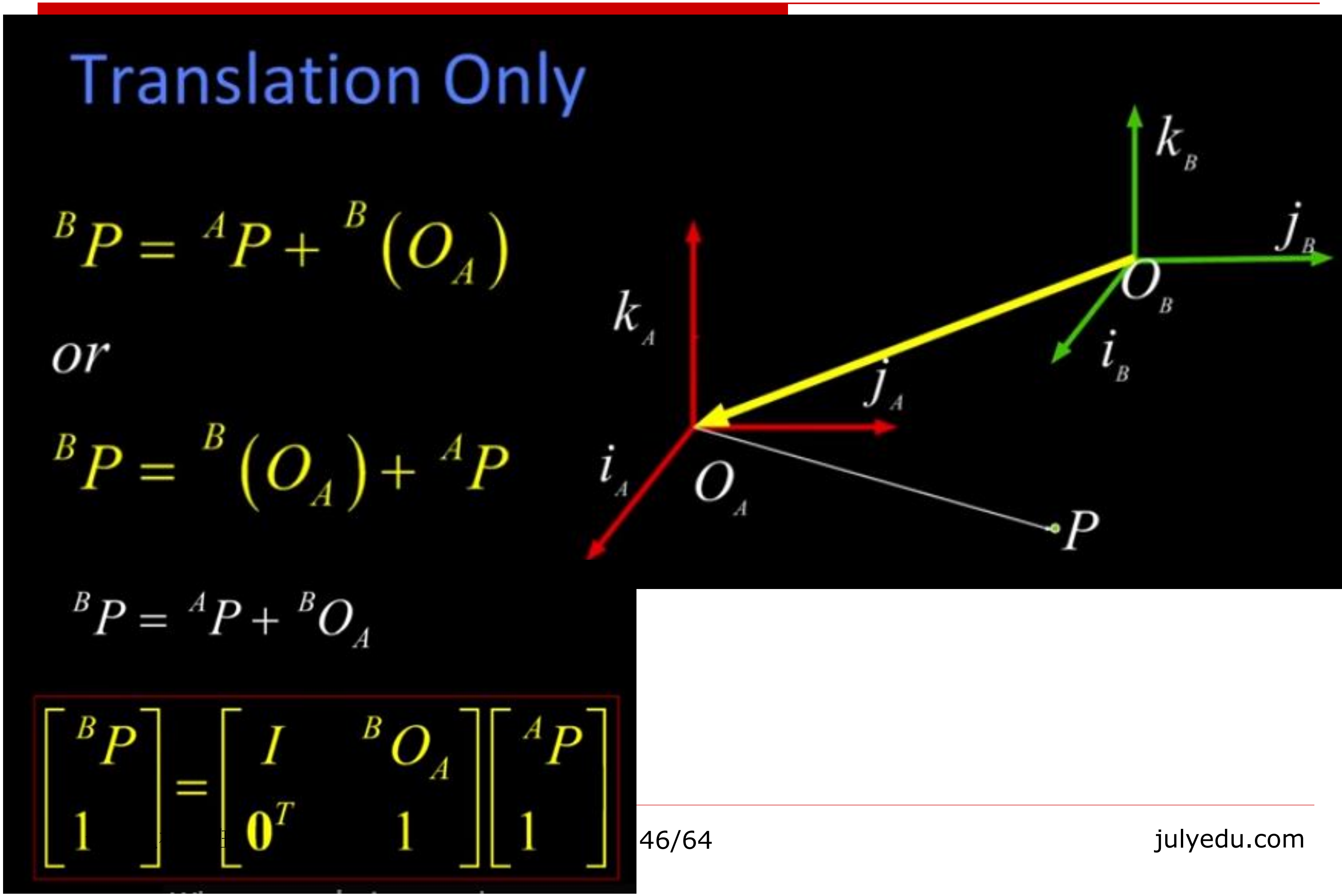
Properties of projective transformations:

- Lines map to lines
- Parallel lines do not necessarily remain parallel
- Ratios are not preserved
- Projective matrix is defined up to a scale (8 DOF)

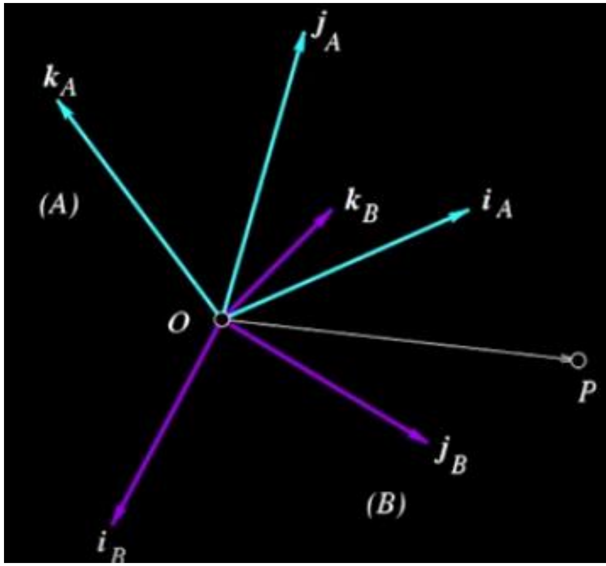
Geometric transform



Geometric transform



Geometric transform



$$ {}^{B}P = {}^{B}_{A}R\, {}^{A}P $$

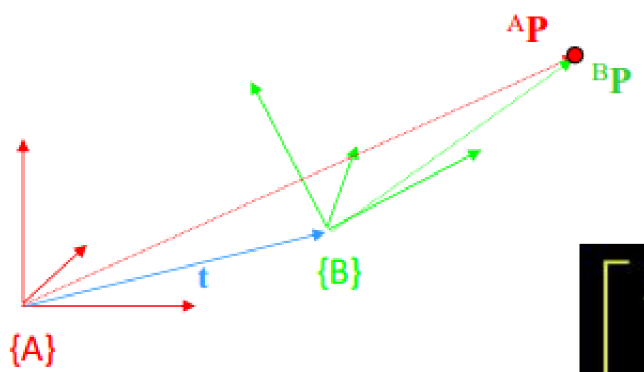
$$ \begin{bmatrix} {}^{B}P \\ 1 \end{bmatrix} = \begin{bmatrix} {}^{B}_{A}R & 0 \\ \mathbf{0}^{T} & 1 \end{bmatrix} \begin{bmatrix} {}^{A}P \\ 1 \end{bmatrix} $$

- R represents a rotational transformation of frame A to frame B
  - I'll use the leading subscript to indicate "from"
  - I'll use the leading superscript to indicate "to"

$$ {}^{B}_{A}\mathbf{R} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} $$

$$ \left({}^{B}_{A}\mathbf{R}\right)^{-1} = \left({}^{B}_{A}\mathbf{R}\right)^{T} = {}^{A}_{B}\mathbf{R} $$

Transform in 3D



$$ {}^{B}\mathbf{P} = {}^{B}_{A}\mathbf{R}\ {}^{A}\mathbf{P} + {}^{B}\mathbf{t}_{Aorg} $$

$$ {}^{A}\mathbf{P} = {}^{A}_{B}\mathbf{R}\ {}^{B}\mathbf{P} + \mathbf{t} $$

$$ \begin{bmatrix} {}^{B}P \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & {}^{B}O_{A} \\ \mathbf{0}^{T} & 1 \end{bmatrix} \begin{bmatrix} {}^{B}_{A}R & 0 \\ \mathbf{0}^{T} & 1 \end{bmatrix} \begin{bmatrix} {}^{A}P \\ 1 \end{bmatrix} $$

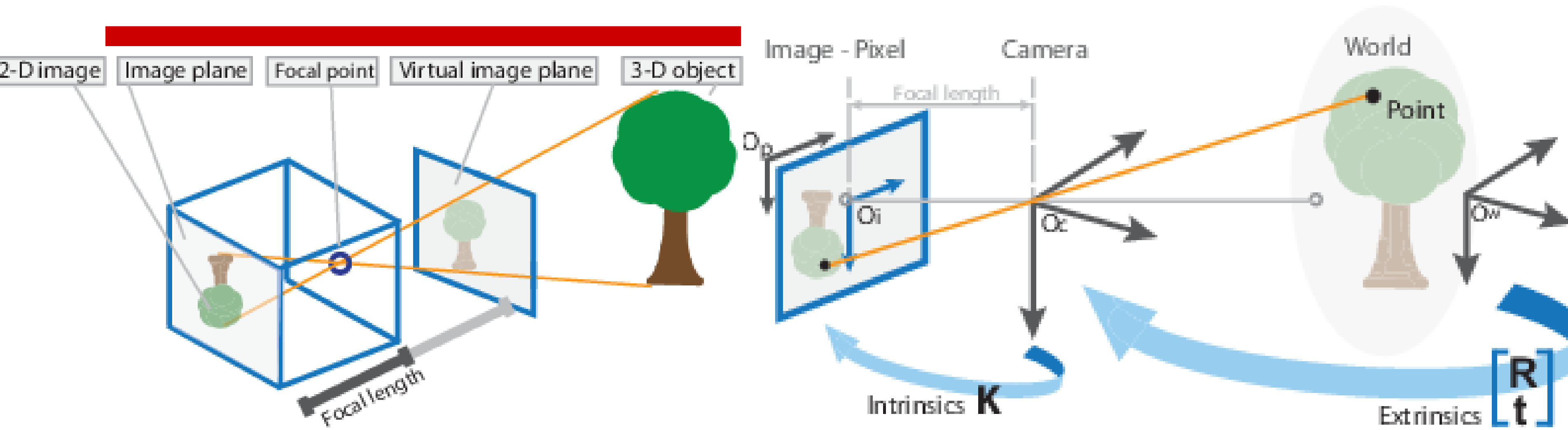
$$ {}^{A}_{B}\mathbf{H} = \left({}^{B}_{A}\mathbf{H}\right)^{-1} $$

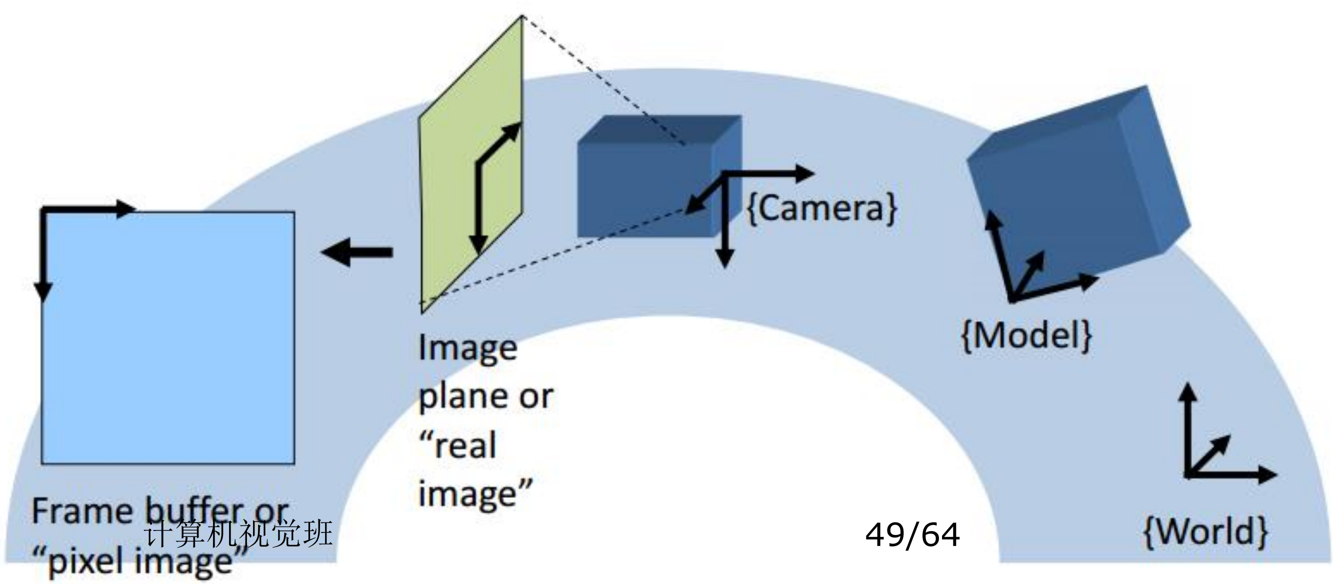
$$ {}^{B}\mathbf{P} = \mathbf{H}\ {}^{A}\mathbf{P}, \quad \text{where } \mathbf{H} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

$$ {}^{A}_{B}\mathbf{H} \neq \left({}^{B}_{A}\mathbf{H}\right)^{T} $$

$$ {}^{C}_{A}\mathbf{H} = {}^{C}_{B}\mathbf{H}\ {}^{B}_{A}\mathbf{H} \qquad {}^{D}_{A}\mathbf{H} = {}^{D}_{C}\mathbf{H}\ {}^{C}_{B}\mathbf{H}\ {}^{B}_{A}\mathbf{H}, \quad \text{etc} $$

Camera calibration

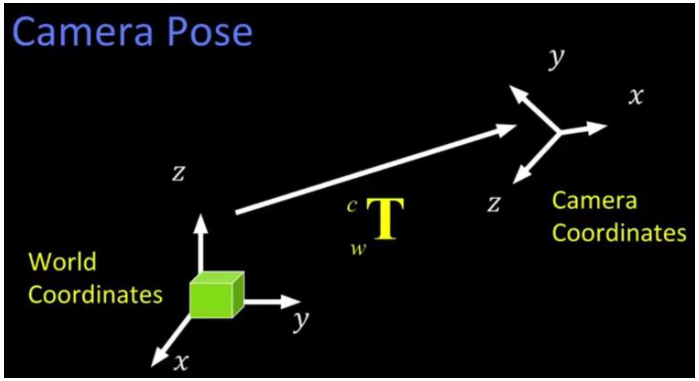




Calibration: 2 steps

- Step 1: Transform into camera coordinates

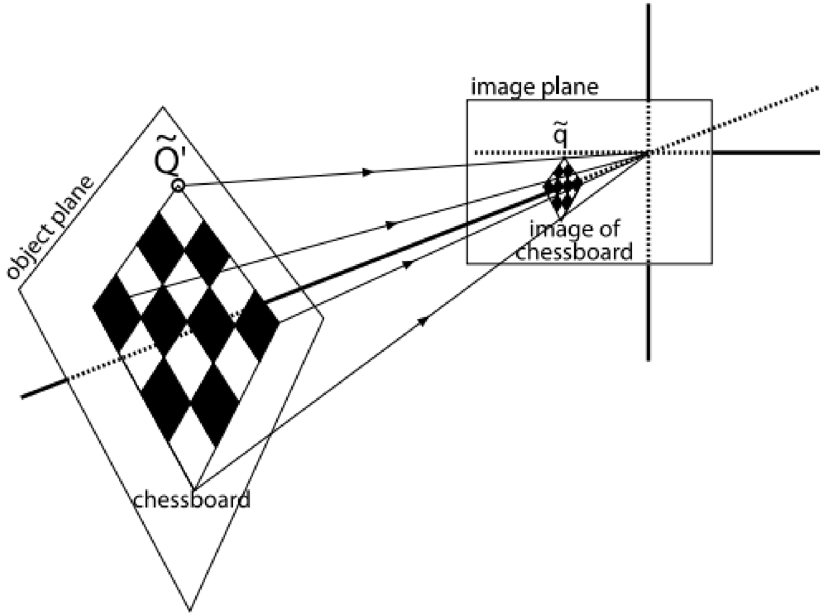
$$\begin{pmatrix} \tilde{X}^C \\ \tilde{Y}^C \\ \tilde{Z}^C \end{pmatrix} = f\left(\begin{pmatrix} X^W \\ Y^W \\ Z^W \end{pmatrix}, \phi, \varphi, \psi, T\right)$$

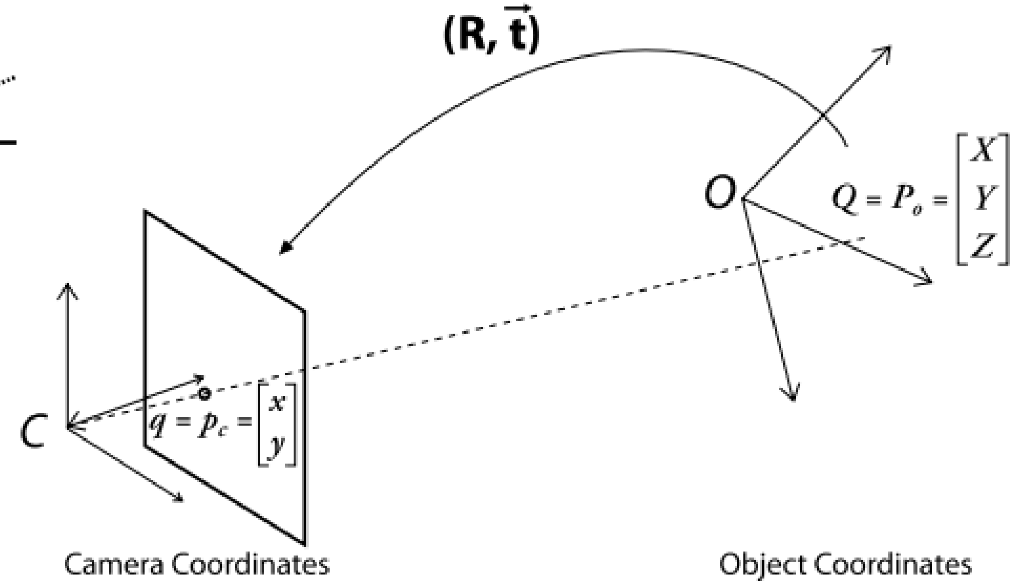


- Step 2: Transform into image coordinates

$$x_{im} = -\frac{f}{s_x}\frac{\tilde{X}^c}{\tilde{Z}^c} + o_x$$

$$y_{im} = -\frac{f}{s_x}\frac{\tilde{Y}^c}{\tilde{Z}^c} + o_y$$





$$x = \frac{f_x X}{Z} + u_0$$

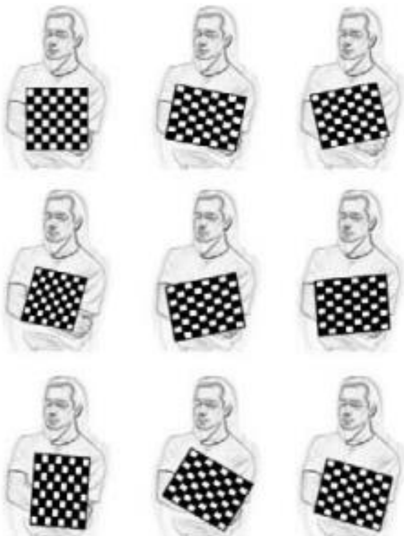
$$y = \frac{f_y Y}{Z} + v_0$$

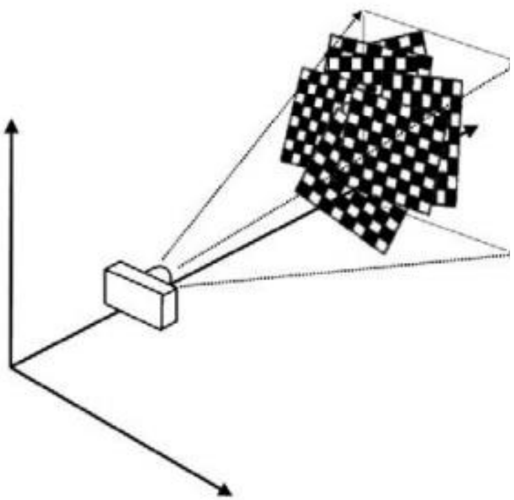
$$s\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

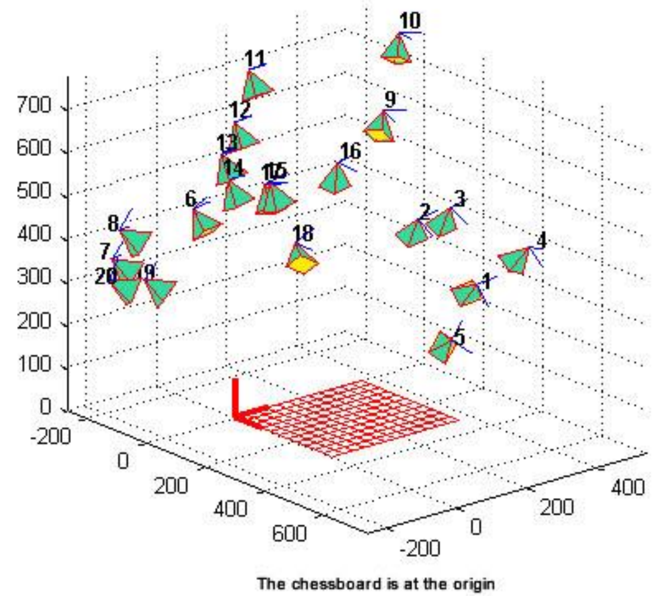
$$s\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} r1 & r2 & r3 & t1 \\ r4 & r5 & r6 & t2 \\ r7 & r8 & r9 & t3 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Calibration in OpenCV

- http://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html
- “Learning OpenCV”



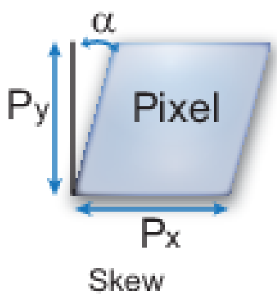




Intrinsic parameters (Matlab)

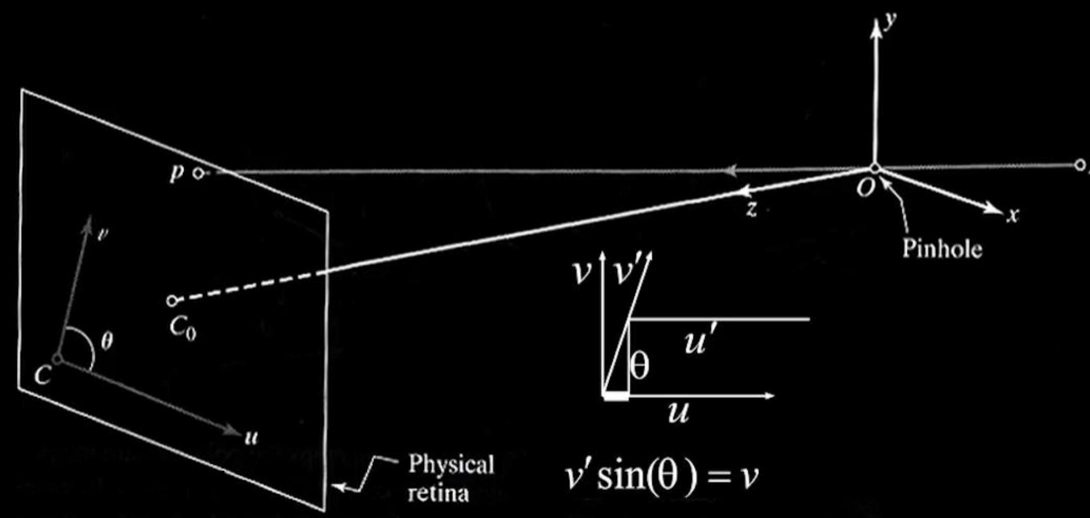
$$u = \alpha \frac{x}{z} + u_0$$

$$v = \beta \frac{y}{z} + v_0$$



$$u = \alpha \frac{x}{z} - \alpha \cot(\theta) \frac{y}{z} + u_0$$

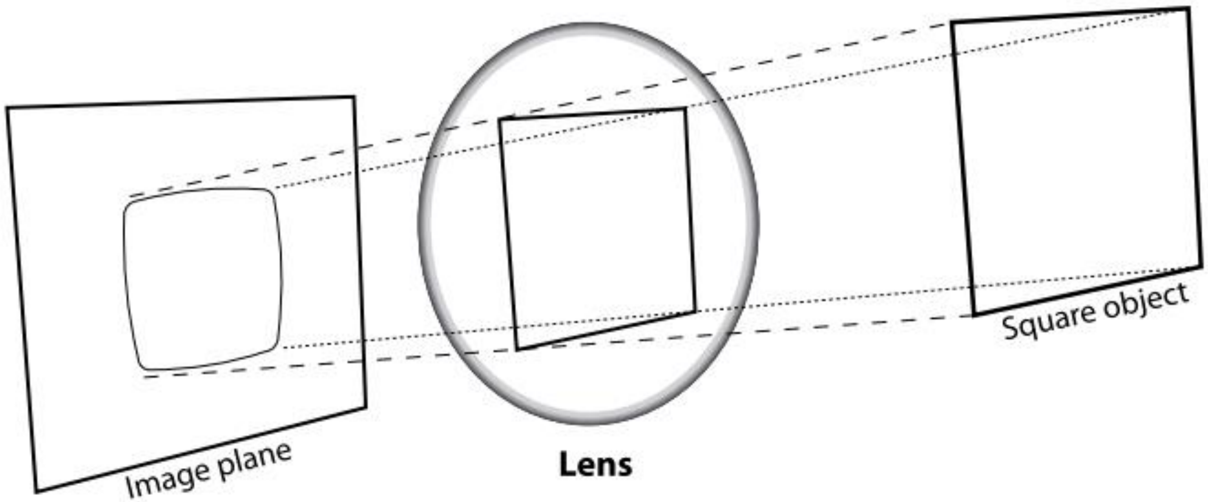
$$v = \frac{\beta}{\sin(\theta)} \frac{y}{z} + v_0$$



$$v' \sin(\theta) = v$$

$$u' = u - \cos(\theta) v' = u - \cot(\theta) v$$

Models of Radial Distortion

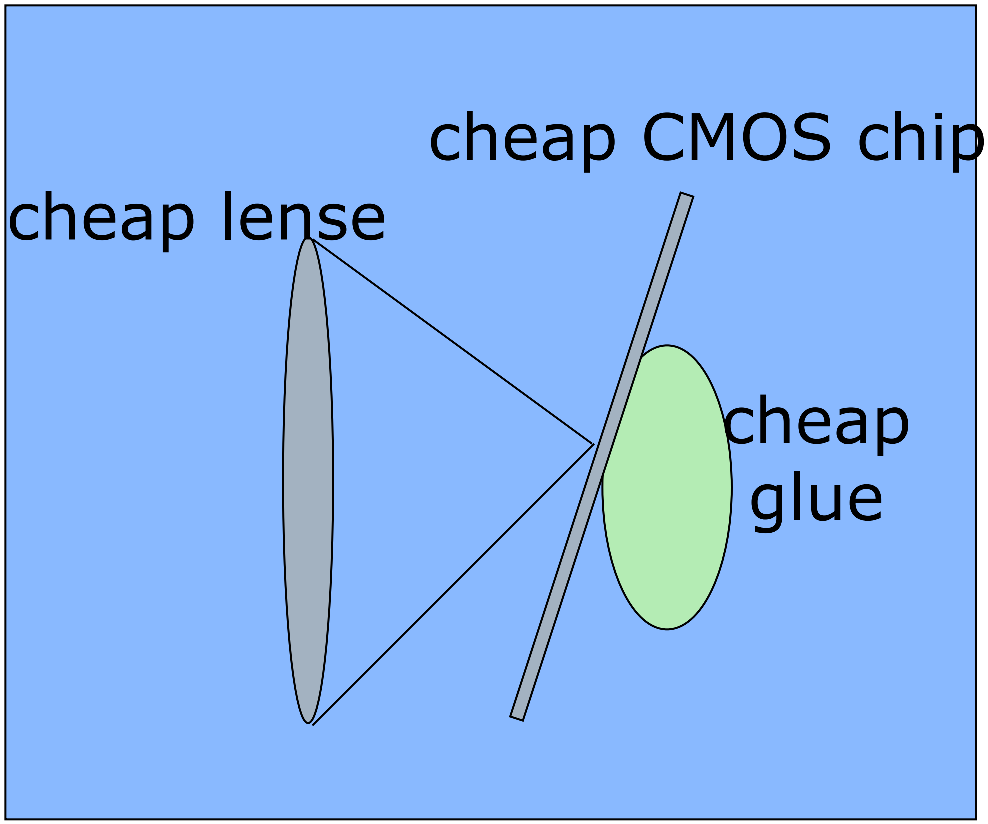


$$x_{\text{corrected}} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

$$y_{\text{corrected}} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

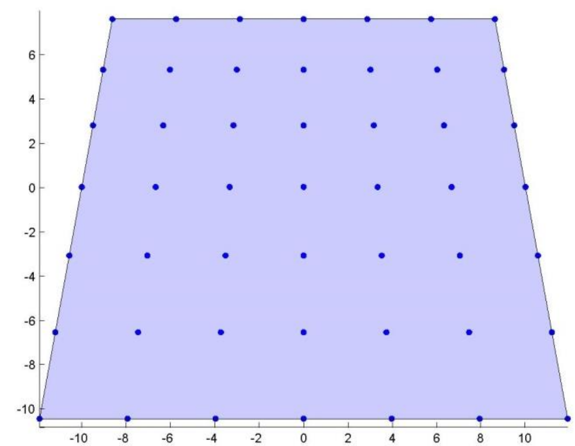
distance from center

Tangential Distortion



cheap camera

image



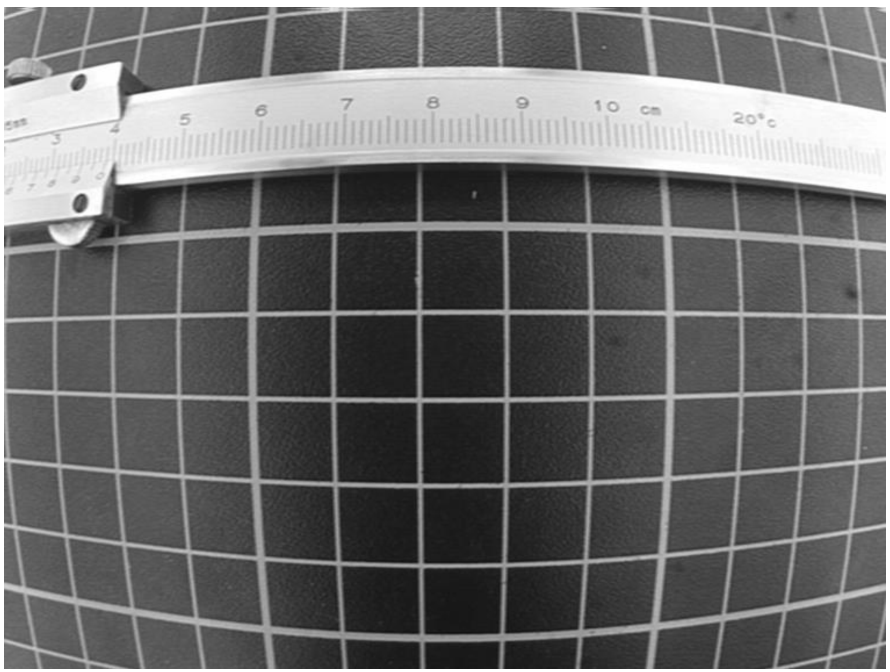
$$x_{\text{corrected}} = x + [2p_1 y + p_2(r^2 + 2x^2)]$$

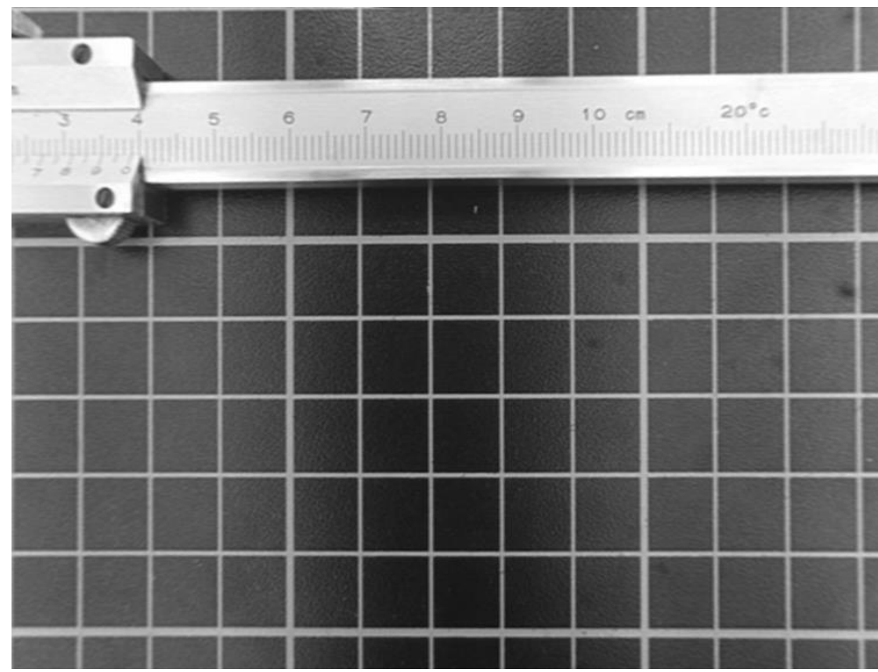
$$y_{\text{corrected}} = y + [p_1(r^2 + 2y^2) + 2p_2 x]$$

Image Rectification

```
undistort(image, imageUndistorted, intrinsic, distCoeffs);
```

http://www.vision.caltech.edu/bouguetj/calib_doc/index.html#parameters



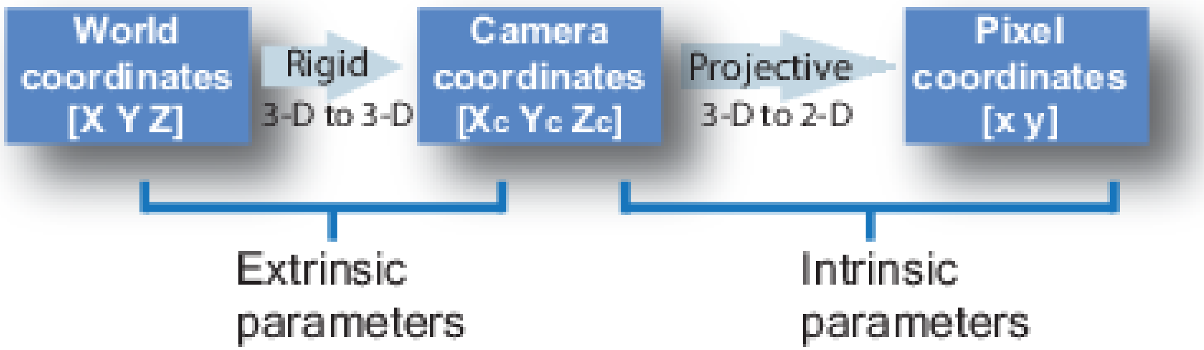


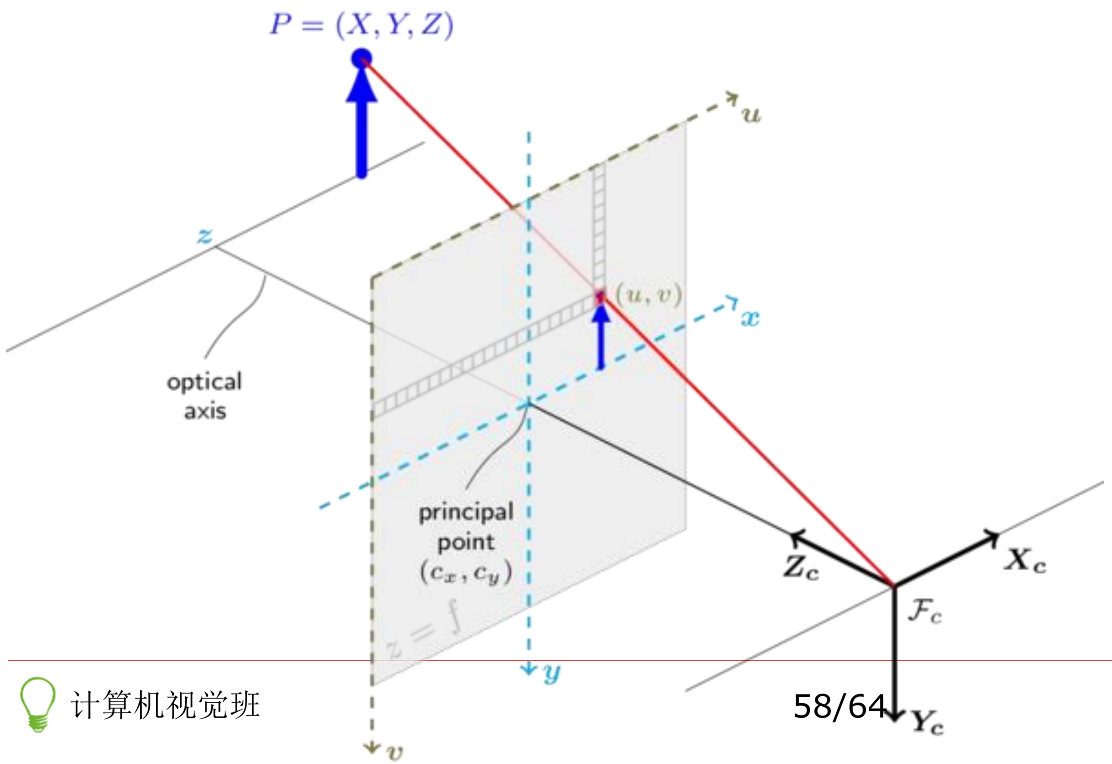
Camera Calibration Toolbox for Matlab

Summary Parameters

- Extrinsic
  - Rotation $\phi.\varphi,\psi$
  - Translation T
- Intrinsic
  - Focal length f
  - Pixel size (s_x, s_y)
  - Image center coordinates (o_x, o_y)
  - (Distortion coefficients) $k_1,\ldots$

Calibration in all



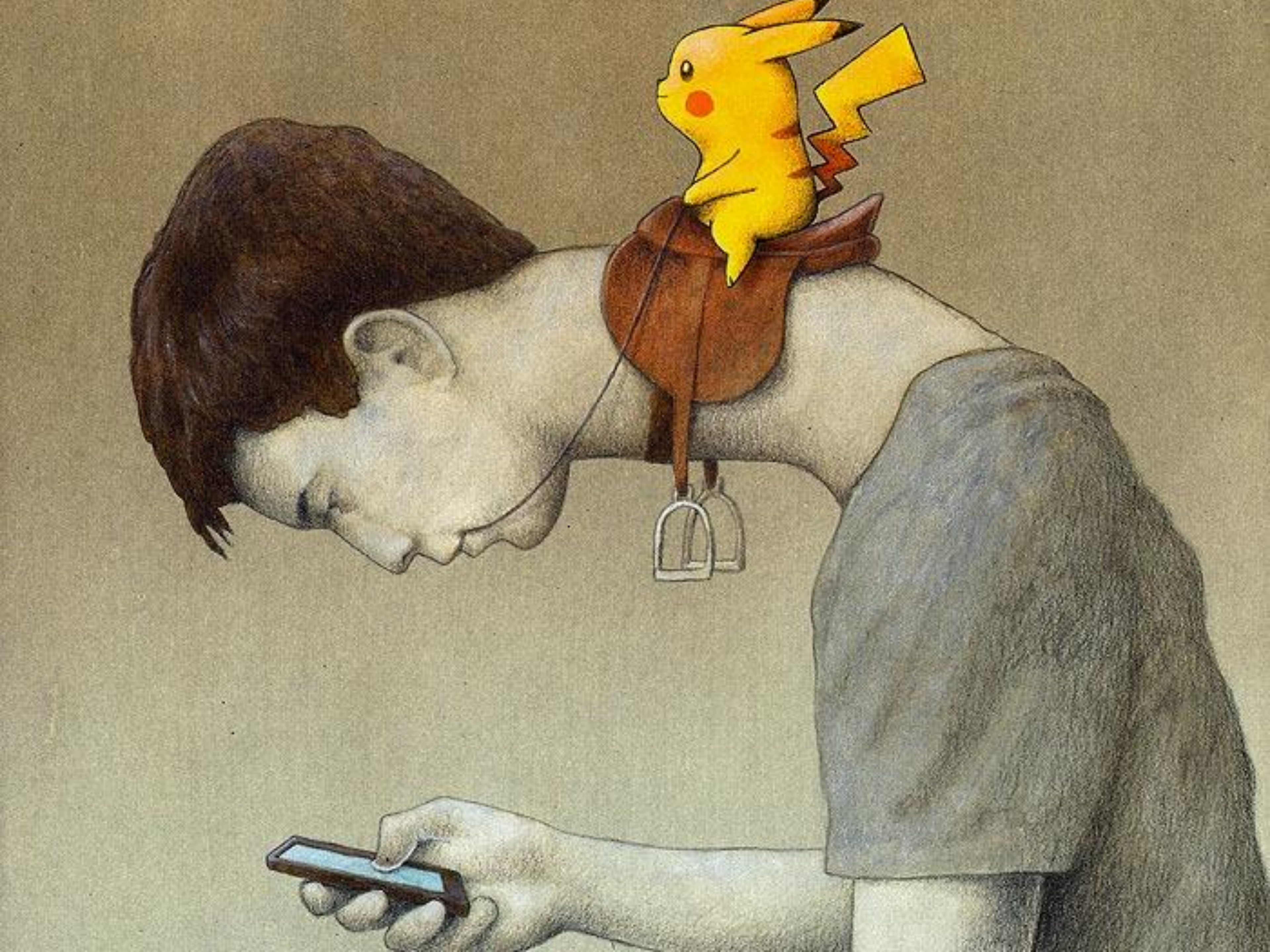


$$\vec{p}\,' = \underset{\boxed{\substack{K\\3\times3}}}{K} \underbrace{\left({}^C_W R \quad {}^C_W \vec{t} \right)}_{\boxed{3\times4}} {}^W\vec{p}$$

$$\vec{p}\,' = M \; {}^W\vec{p}$$

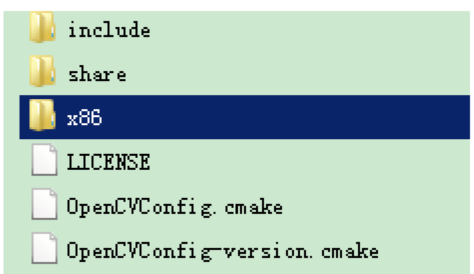
Appendix

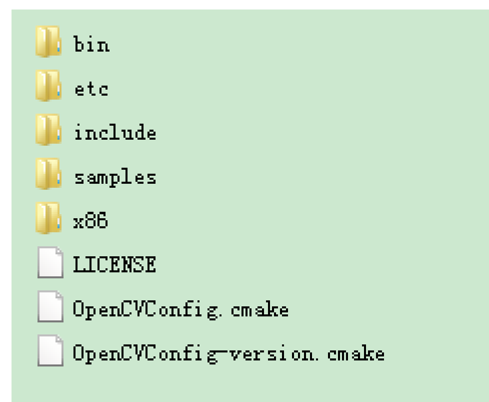
- A calibration sample based on a sequence of images can be found at opencv_source_code/samples/cpp/calibration.cpp
- A calibration example on stereo calibration can be found at opencv_source_code/samples/cpp/stereo_calib.cpp



Set up OpenGL in OpenCV with CMake

- 下载cmake：http://www.cmake.org/download/
- 路径：sources文件夹；勾选advanced，然后configure；Generate
  - 检查 ‘CMAKE_LINKER’, 保证是 Visual Studio 12.0 （vs2013）选上 ‘WITH_OPENGL’
  - 取消 ‘BUILD_DOCS’ and ‘BUILD_EXAMPLES’



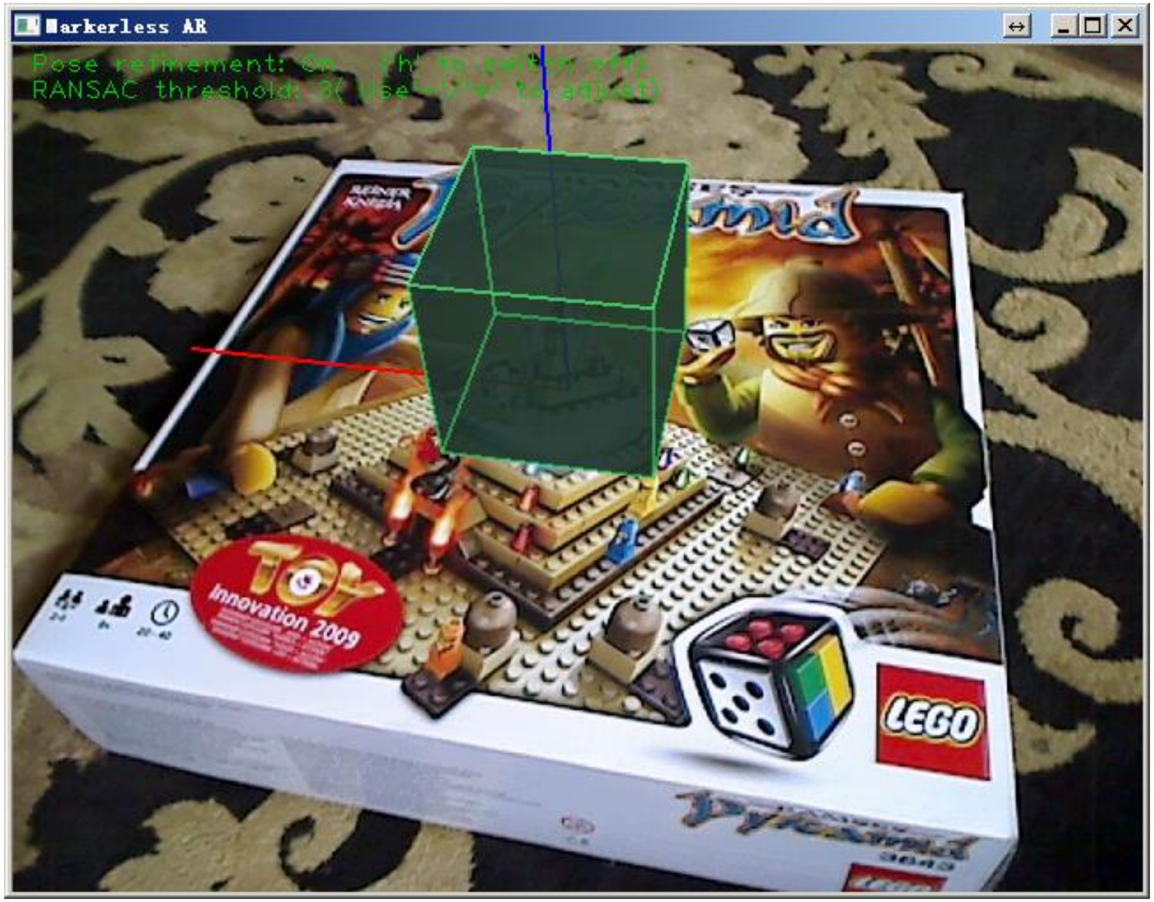


Example: Simple AR

"Mastering OpenCV with Practical Computer Vision Projects"

- 相机标定
- 提取模板图像的特征与描述子
  - ORB + FREAK
- 图像（实时）匹配
  - 特征检测，描述子提取，离群值过滤
- 单应性变换
- 姿态估计并画图

Markerless AR



感谢大家！

恳请大家批评指正！