CAS 765 Fall'15

Mobile Computing and Wireless Networking

Rong Zheng

Sensor & Sensor Data Processing

Part III Camera

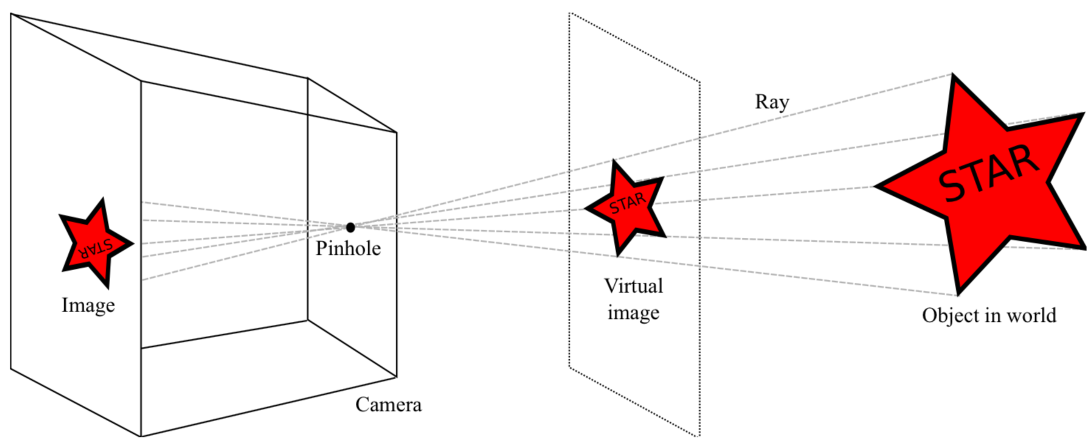
Materials from Prince, "Computer vision: models, learning and inference"

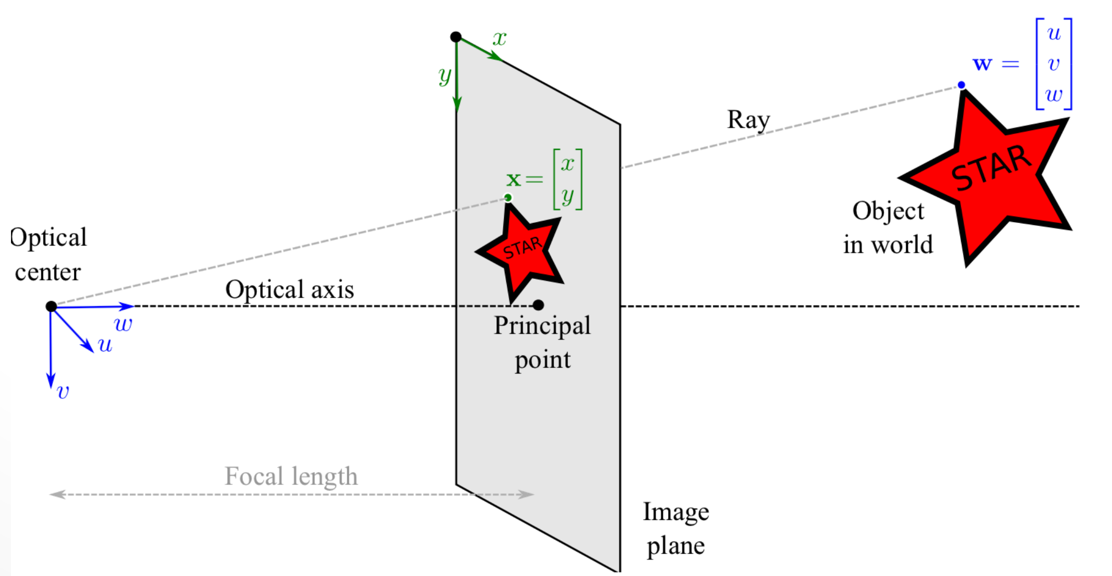
Learning Objectives

- Understand the pinhole camera model for monocular camera
- Understand the base principles behind camera calibration, camera pose estimation, vision-based object localization
- Understand basic image processing techniques

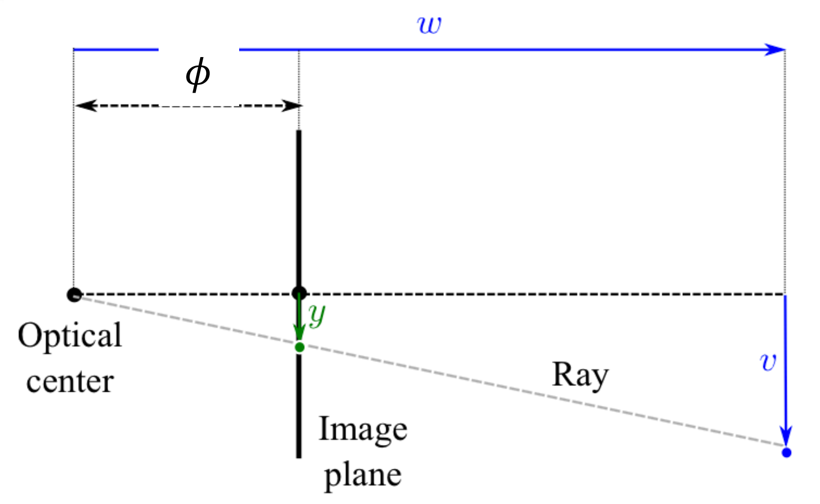
Pinhole Camera

- A geometric model that describes how points are projected onto the image





Geometric Relation



Assume the optical center is at the world origin, from geometry, we have

$$x = \frac{\phi u}{\omega},\ y = \frac{\phi v}{\omega}$$

If considering the spacing of photoperceptor

$$x = \frac{\phi_x u}{\omega},\ y = \frac{\phi_y v}{\omega}$$

Considering offsets

$$x = \frac{\phi_x u}{\omega} + \delta_x,\ y = \frac{\phi_y v}{\omega} + \delta_y \text{ (in pixel)}$$

Skew

$$x = \frac{\phi_x u + \gamma u}{\omega} + \delta_x,\ y = \frac{\phi_y v}{\omega} + \delta_y \text{ (in pixel)}$$

Geometric Relation (cont'd)

- If the optical center is NOT at the world origin

$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \underbrace{\begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} \\ \omega_{21} & \omega_{22} & \omega_{23} \\ \omega_{31} & \omega_{32} & \omega_{33} \end{bmatrix}}_{\text{Rotation matrix}} \begin{bmatrix} u \\ v \\ w \end{bmatrix} + \underbrace{\begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix}}_{\text{Translation vector}}$$

Camera frame

$$w' = \boldsymbol{\Omega} w + \boldsymbol{\tau}$$

World frame

- Full pinhole camera model
  - Extrinsic parameters $\{\Omega, \tau\}$
  - Intrinsic parameters $\{\phi_x, \phi_y, \delta_x, \delta_y, \gamma\}$

$$\begin{aligned} x &= \frac{\phi_x(\omega_{11}u + \omega_{12}v + \omega_{13}w + \tau_x) + \gamma(\omega_{21}u + \omega_{22}v + \omega_{23}w + \tau_y)}{\omega_{31}u + \omega_{32}v + \omega_{33}w + \tau_z} + \delta_x \\ y &= \frac{\phi_y(\omega_{21}u + \omega_{22}v + \omega_{23}w + \tau_y)}{\omega_{31}u + \omega_{32}v + \omega_{33}w + \tau_z} + \delta_y. \qquad (1 \end{aligned}$$

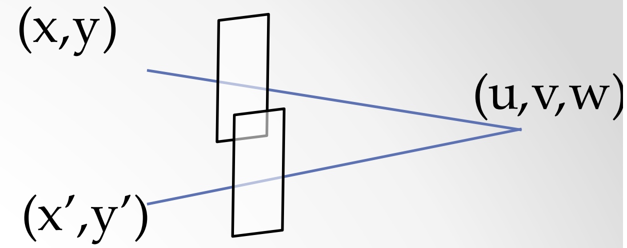
Three Related Problems

- Learning the extrinsic parameters (camera pose)
- Learning intrinsic parameters (calibration)
- Inferring 3D world points: estimate the 3D position of a point w in the scene, given its projections $\{x_j, y_j\}_{j=1}^{J}$ in J≥ 2 calibrated cameras with known poses

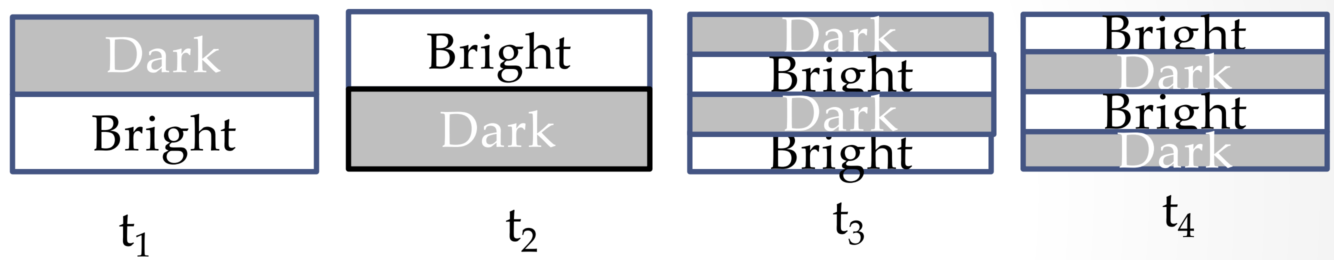
Tractability

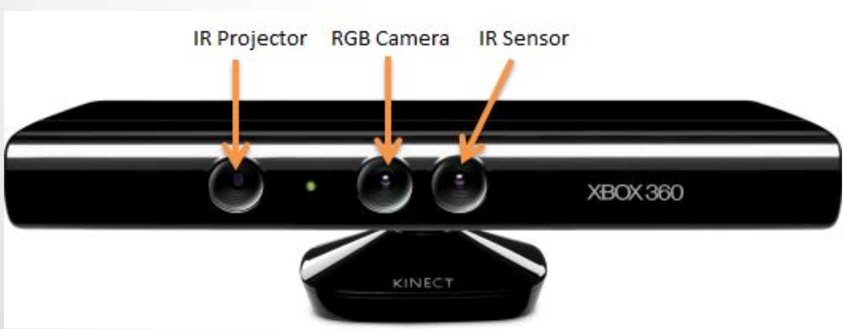
- How many unknowns for I objects (points) projected on a single camera at J unknown locations and poses?
    - 6 extrinsic parameters per camera poses, 5 intrinsic parameters per camera, 3 unknowns per one world point → 6J+5+3I
    - Knowns: J projections of I objects → $2I \cdot J$ equations
    - Conceptually solvable if 6J+5+3I $\leq 2I \cdot J$
- How to find *I* objects? → this is called the data association problem
- More *I* than needed? Noise? → treat as an optimization problem

Applications



- Depth from structured light
- Camera + Projector → J = 2
  - Projectors are essentially the same as cameras *geometrically*

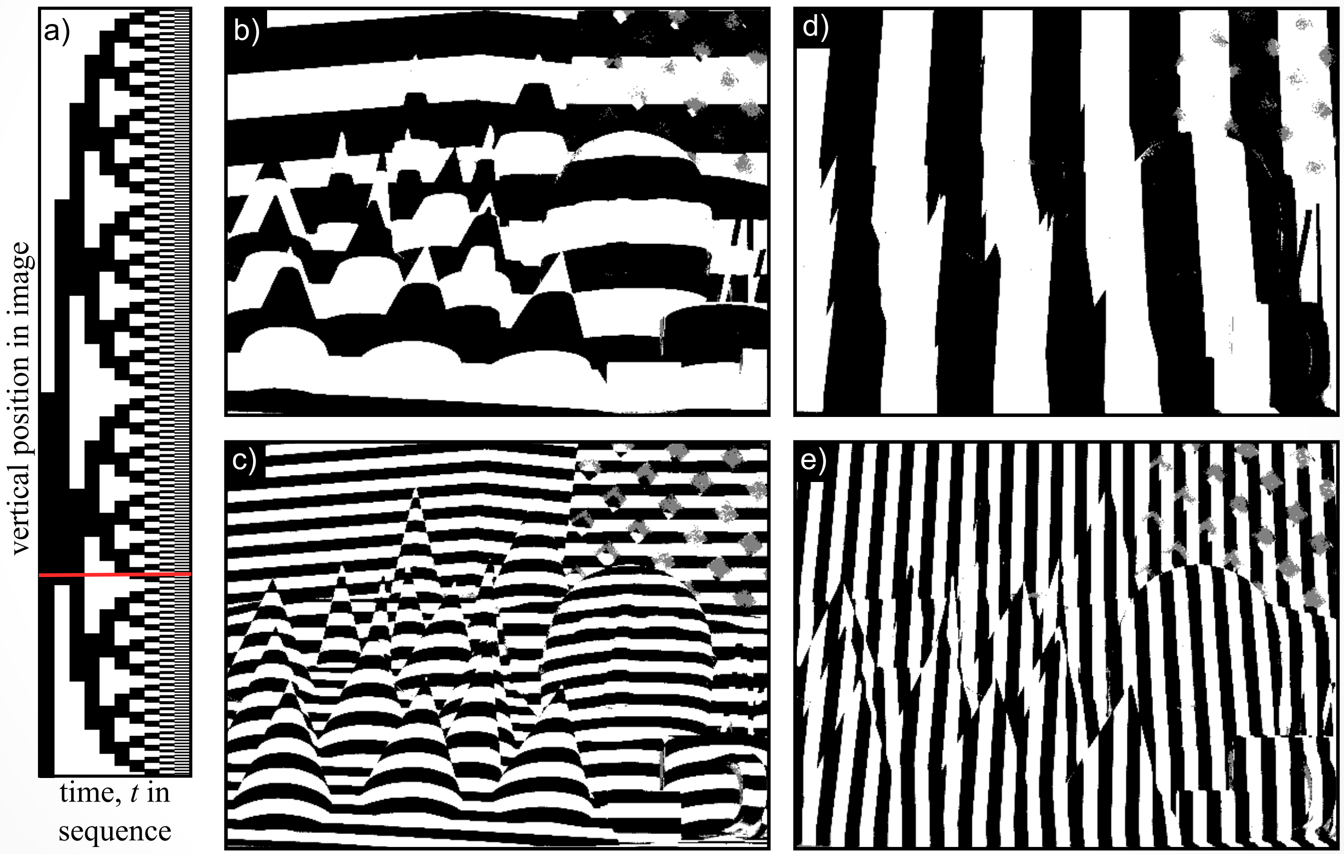




0 – Dark, 1 –Bright
0 1 0 1
0 1 1 0
1 0 0 1
1 0 1 0

Horizonal stripes → Vectical position
Vertical stripes → Horizontal position

More on Structured Light



Horizontal stripes

Vertical stripes

Homogeneous Coordinates

- Unifies the computation of geometric transformations: rotation, translation, affine transformation, scaling, projection
- From 2D (x,y) to 3D homogeneous coordinates
  - Any scalar λ represents the same 2D point

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

- Similar for 3D points (u,v,w)

$$\tilde{\mathbf{w}} = \lambda \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix}$$

Camera Model in Homogeneous Coordinates

- Pinhole projection

$$
\begin{aligned}
x &= \frac{\phi_x u + \gamma v}{w} + \delta_x \\
y &= \frac{\phi_y v}{w} + \delta_y,
\end{aligned}
\quad\Rightarrow\quad
\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} \phi_x & \gamma & \delta_x & 0 \\ 0 & \phi_y & \delta_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{[\boldsymbol{\Lambda}\ \mathbf{0}]} \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix}
$$

- Rotation and translation

$$
\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} \\ \omega_{21} & \omega_{22} & \omega_{23} \\ \omega_{31} & \omega_{32} & \omega_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} + \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix}
\quad\Rightarrow\quad
\begin{bmatrix} u' \\ v' \\ w' \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} & \tau_x \\ \omega_{21} & \omega_{22} & \omega_{23} & \tau_y \\ \omega_{31} & \omega_{32} & \omega_{33} & \tau_z \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\begin{bmatrix} \boldsymbol{\Omega} & \boldsymbol{\tau} \\ \mathbf{0}^T & 1 \end{bmatrix}} \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix}
$$

Camera Model in Homogeneous Coordinates

- Combined model

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \phi_x & \gamma & \delta_x & 0 \\ 0 & \phi_y & \delta_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} & \tau_x \\ \omega_{21} & \omega_{22} & \omega_{23} & \tau_y \\ \omega_{31} & \omega_{32} & \omega_{33} & \tau_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix}$$

- Or in matrix form

$$\lambda \tilde{\mathbf{x}} = \begin{bmatrix} \boldsymbol{\Lambda} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Omega} & \boldsymbol{\tau} \\ \mathbf{0}^T & 1 \end{bmatrix} \tilde{\mathbf{w}} \quad \Rightarrow \quad \lambda \tilde{\mathbf{x}} = \boldsymbol{\Lambda} \begin{bmatrix} \boldsymbol{\Omega} & \boldsymbol{\tau} \end{bmatrix} \tilde{\mathbf{w}}$$

Linear form compared to

$$\boxed{\begin{aligned} x &= \frac{\phi_x(\omega_{11}u + \omega_{12}v + \omega_{13}w + \tau_x) + \gamma(\omega_{21}u + \omega_{22}v + \omega_{23}w + \tau_y)}{\omega_{31}u + \omega_{32}v + \omega_{33}w + \tau_z} + \delta_x \\ y &= \frac{\phi_y(\omega_{21}u + \omega_{22}v + \omega_{23}w + \tau_y)}{\omega_{31}u + \omega_{32}v + \omega_{33}w + \tau_z} + \delta_y. \qquad (1 \end{aligned}}$$

Learning Extrinsic Parameters

- Everything in the red circles are known

$$\lambda_i \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} \phi_x & \gamma & \delta_x \\ 0 & \phi_y & \delta_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} & \tau_x \\ \omega_{21} & \omega_{22} & \omega_{23} & \tau_y \\ \omega_{31} & \omega_{32} & \omega_{33} & \tau_z \end{bmatrix} \begin{bmatrix} u_i \\ v_i \\ w_i \\ 1 \end{bmatrix}$$

- Let $\tilde{x}' = \Lambda^{-1}\tilde{x}$, we have

$$\lambda_i \begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} = \begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} & \tau_x \\ \omega_{21} & \omega_{22} & \omega_{23} & \tau_y \\ \omega_{31} & \omega_{32} & \omega_{33} & \tau_z \end{bmatrix} \begin{bmatrix} u_i \\ v_i \\ w_i \\ 1 \end{bmatrix}$$

- Observing that from the last equation

$$\lambda_i = \omega_{31}u_i + \omega_{32}v_i + \omega_{33}w_i + \tau_z$$

Learning Extrinsic Parameters

- Given *I* points and their projections, we have a set of linear equations of the form Ab = 0
  - Solutions can be found via singular value decomposition A = ULV^T and setting b to be the last column of V
  - Since the solution can be of arbitrary scale, need to normalize to get desired Ω

$$\begin{bmatrix} u_1 & v_1 & w_1 & 1 & 0 & 0 & 0 & 0 & -u_1x'_1 & -v_1x'_1 & -w_1x'_1 & -x'_1 \\ 0 & 0 & 0 & 0 & u_1 & v_1 & w_1 & 1 & -u_1y'_1 & -v_1y'_1 & -w_1y'_1 & -y'_1 \\ u_2 & v_2 & w_2 & 1 & 0 & 0 & 0 & 0 & -u_2x'_2 & -v_2x'_2 & -w_2x'_2 & -x'_2 \\ 0 & 0 & 0 & 0 & u_2 & v_2 & w_2 & 1 & -u_2y'_2 & -v_2y'_2 & -w_2y'_2 & -y'_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_I & v_I & w_I & 1 & 0 & 0 & 0 & 0 & -u_Ix'_I & -v_Ix'_I & -w_Ix'_I & -x'_I \\ 0 & 0 & 0 & 0 & u_I & v_I & w_I & 1 & -u_Iy'_I & -v_Iy'_I & -w_Iy'_I & -y'_I \end{bmatrix} \begin{bmatrix} \omega_{11} \\ \omega_{12} \\ \omega_{13} \\ \tau_x \\ \omega_{21} \\ \omega_{22} \\ \omega_{23} \\ \tau_y \\ \omega_{31} \\ \omega_{32} \\ \omega_{33} \\ \tau_z \end{bmatrix} = \mathbf{0}.$$

Learning Intrinsic Parameters

- If known extrinsic parameters

$$\lambda_i \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} \phi_x & \gamma & \delta_x \\ 0 & \phi_y & \delta_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} & \tau_x \\ \omega_{21} & \omega_{22} & \omega_{23} & \tau_y \\ \omega_{31} & \omega_{32} & \omega_{33} & \tau_z \end{bmatrix} \begin{bmatrix} u_i \\ v_i \\ w_i \\ 1 \end{bmatrix}$$

- Can solve using similar approach as before or solving a least squares problem by defining

$$\mathbf{h} = [\phi_x, \gamma, \delta_x, \phi_y, \delta_y]^T$$

$$\mathrm{A}_i = \begin{bmatrix} \frac{\omega_{11}u_i+\omega_{12}v_i+\omega_{13}w_i+\tau_x}{\omega_{31}u_i+\omega_{32}v_i+\omega_{33}w_i+\tau_z} & \frac{\omega_{21}u_i+\omega_{22}v_i+\omega_{23}w_i+\tau_x}{\omega_{31}u_i+\omega_{32}v_i+\omega_{33}w_i+\tau_z} & 1 & 0 & 0 \\ 0 & 0 & 0 & \frac{\omega_{21}u_i+\omega_{22}v_i+\omega_{23}w_i+\tau_y}{\omega_{31}u_i+\omega_{32}v_i+\omega_{33}w_i+\tau_z} & 1 \end{bmatrix}$$

And solve

$$\hat{\mathbf{h}} = \underset{\mathbf{h}}{\operatorname{argmin}} \left[\sum_{i=1}^{I} (\mathbf{A}_i\mathbf{h} - \mathbf{x}_i)^T (\mathbf{A}_i\mathbf{h} - \mathbf{x}_i) \right]$$

Finding World Coordinates

- Now, we assume the intrinsic and extrinsic coordinates are both known

$$\lambda_i \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} \phi_x & \gamma & \delta_x \\ 0 & \phi_y & \delta_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} & \tau_x \\ \omega_{21} & \omega_{22} & \omega_{23} & \tau_y \\ \omega_{31} & \omega_{32} & \omega_{33} & \tau_z \end{bmatrix} \begin{bmatrix} u_i \\ v_i \\ w_i \\ 1 \end{bmatrix}$$

- Finding the world coordinates corresponding to solving a collection of equations (in least squares)

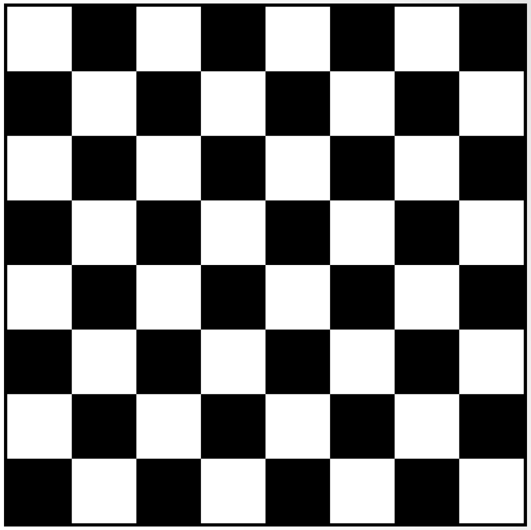
$$\begin{bmatrix} \omega_{31j}x'_j - \omega_{11j} & \omega_{32j}x'_j - \omega_{12j} & \omega_{33j}x'_j - \omega_{13j} \\ \omega_{31j}y'_j - \omega_{21j} & \omega_{32j}y'_j - \omega_{22j} & \omega_{33j}y'_j - \omega_{23j} \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} \tau_{xj} - \tau_{zj}x'_j \\ \tau_{yj} - \tau_{zj}y'_j \end{bmatrix}$$

All Unknown?

- Assume data association is done
- Typically intrinsic parameters are determined in a calibration step using known images (e.g., chessboard)
  - In this case, the world coordinates are known, jointly solve for intrinsic and extrinsic parameters
- Now with intrinsic parameters, jointly solve for extrinsic and world coordinates
  - Gist of simultaneous localization and mapping (SLAM)

Camera Calibration

- A few images from different orientations of a known image with w = 0



$$\lambda_i \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} \phi_x & \gamma & \delta_x \\ 0 & \phi_y & \delta_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} & \tau_x \\ \omega_{21} & \omega_{22} & \omega_{23} & \tau_y \\ \omega_{31} & \omega_{32} & \omega_{33} & \tau_z \end{bmatrix} \begin{bmatrix} u_i \\ v_i \\ w_i \\ 1 \end{bmatrix}$$

$$\lambda_i \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \boldsymbol{\Lambda}\underbrace{[\boldsymbol{r}_1^j \ \boldsymbol{r}_2^j \ \boldsymbol{\tau}]}\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \text{ since } w = 0 \quad \text{For jth orientation}$$

H= [$\mathbf{h}_1$, $\mathbf{h}_2$, $\mathbf{h}_3$]

Detecting corners → Determine H → Estimate $\boldsymbol{\Lambda}$

$$\min_{\mathbf{H}} \sum_i \|\bar{\mathbf{m}}_i - \hat{\mathbf{m}}_i\|^2$$

$$\hat{\mathbf{m}}_i = \frac{1}{\bar{\mathbf{h}}_3^T \mathtt{M}_i} \begin{bmatrix} \bar{\mathbf{h}}_1^T \mathtt{M}_i \\ \bar{\mathbf{h}}_2^T \mathtt{M}_i \end{bmatrix}$$

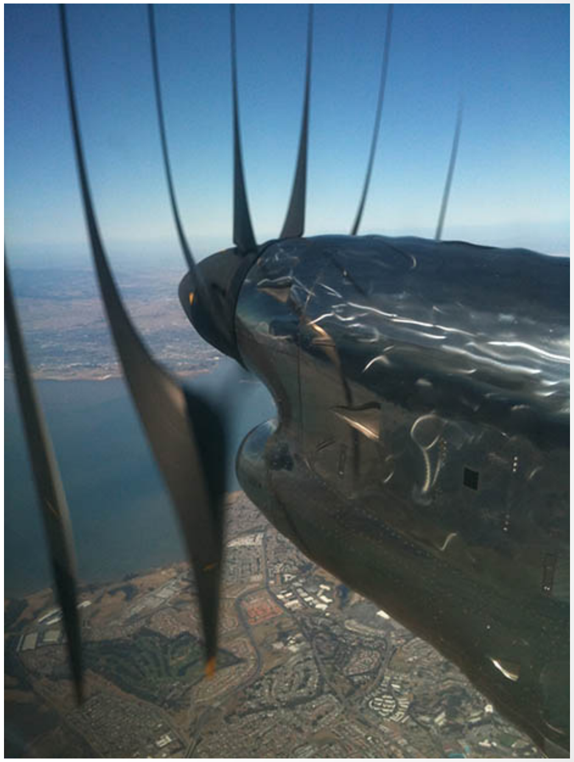
$$h_1^T \Lambda^{-T} \Lambda h_2 = 0$$

$$h_1^T \Lambda^{-T} \Lambda h_1 = h_2^T \Lambda^{-T} \Lambda h_2$$

- Zhengyou Zhang, A Flexible New Technique for Camera Calibration, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000

Practical Matters

- Numerical issues
  - Even in cases where closed-form solutions exist, it is still beneficial to find solution via optimization due to measurement noises
  - Stable solutions are non-trivial
  - Be aware of computation complexity
- Combining IMU
  - Recall IMUs can provide estimation of device pose → may combine with camera data
- CMOS camera: today's digital cameras are predominantly based on CMOS sensors
  - Rolling shutter effects: a still image or a frame of a video is captured by scanning across the scene rapidly, either vertically or horizontally.



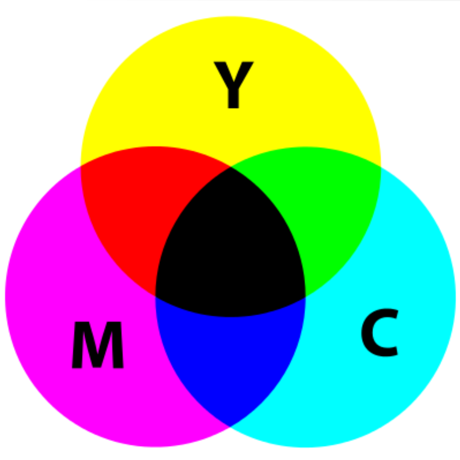
https://www.youtube.com/watch?v=EaB9EHeDLSk

Image Processing

- In discussing the camera calibration process, we assume that corners in the chessboard images can be detected
- Now we discuss some basic image processing operations
  - Per-pixel transformation: whitening, histogram equalization, local filtering
  - Edges, corners, and interest points
  - SIFT descriptors

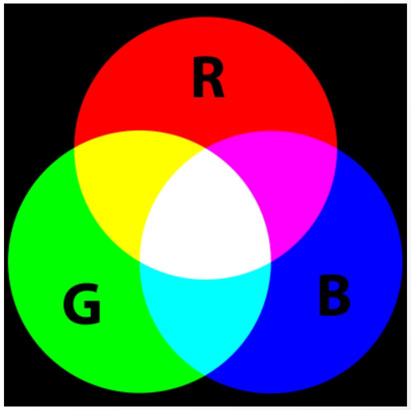
Digital Color Coding

- CMYK (used in printing)
- RGB 8-bit each (r,g,b)
- YUV breaks RGB into four parts: luminance (Y) and 3 chrominance component (Cr, Cg, Cb)
  - Cr + Cg + Cb = 1



RGB to YCbCr	YCbCr to RGB
Y = 0.299**R** + 0.587**G** + 0.114**B**	R = **Y** + 1.402**Cr**
Cb = 0.564(**B** – Y)	B = **Y** + 1.772**Cb**
Cr = 0.713(**R** – Y)	G = **Y** – 0.344**Cb** – 0.714**Cr**

Mapping for SDTV BT.601



$$\begin{bmatrix} Y' \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.2126 & 0.7152 & 0.0722 \\ -0.09991 & -0.33609 & 0.436 \\ 0.615 & -0.55861 & -0.05639 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.28033 \\ 1 & -0.21482 & -0.38059 \\ 1 & 2.12798 & 0 \end{bmatrix} \begin{bmatrix} Y' \\ U \\ V \end{bmatrix}$$

Mapping for HDTV BT.709

Per-pixel Transformation

- For simplicity, we consider gray-scale images
- 2D array of pixel data as P, where p_{ij} is the element at the i^{th} of I rows and the j^{th} of J columns
- Whitening

$$\mu = \frac{\sum_{i=1}^{I}\sum_{j=1}^{J} p_{ij}}{IJ}$$

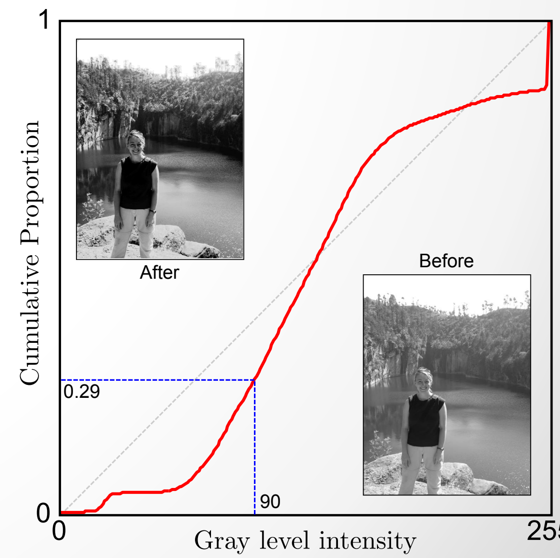
$$\sigma^2 = \frac{\sum_{i=1}^{I}\sum_{j=1}^{J} (p_{ij}-\mu)^2}{IJ}. \qquad x_{ij} = \frac{p_{ij}-\mu}{\sigma}.$$

- Histogram equalization (K levels)

histogram $\quad h_k = \sum_{i=1}^{I}\sum_{j=1}^{J} \delta[p_{ij}-k]$

CDF $\quad c_k = \frac{\sum_{l=1}^{k} h_l}{IJ}$

Transformation $\quad x_{ij} = K c_{p_{ij}}$



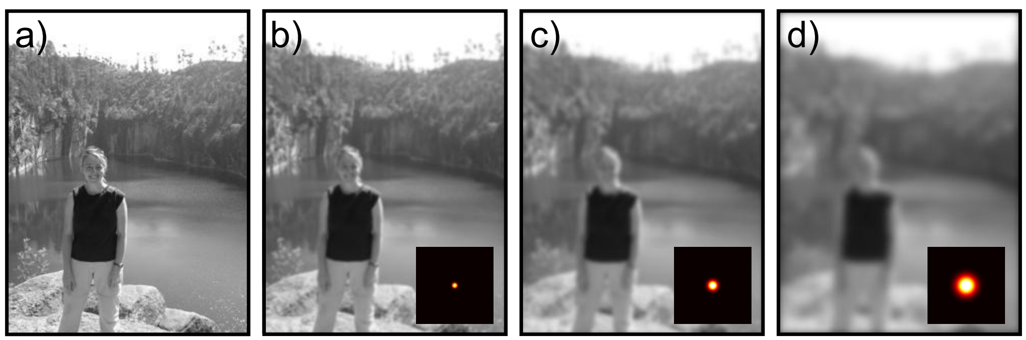
Linear Filter

- Images as 2D digital signal -- convoluting an image with a filter of window MxN

$$x_{ij} = \sum_{m=-M}^{M} \sum_{n=-N}^{N} p_{i-m,j-n} f_{m,n}$$

- Gaussian blur filter
  - σ can be thought of as scale

$$f(m,n) = \frac{1}{2\pi\sigma^2} \exp\left[-\frac{m^2+n^2}{2\sigma^2}\right]$$



Effects of Gaussian blur filter with different σ's

Linear Filters

- First derivative filters and edge filters
  - Prewitt operators

$$\mathbf{F}_x = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}, \qquad \mathbf{F}_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

  - Sobel operators

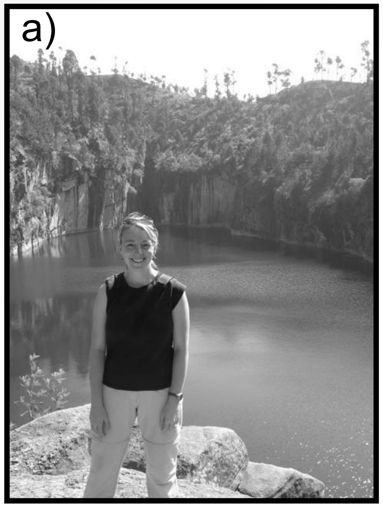
$$\mathbf{F}_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \qquad \mathbf{F}_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

  - Laplacian filters

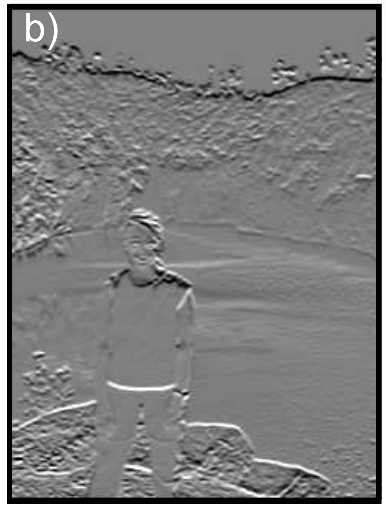
$$\mathbf{F} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

  - Laplacian of Gaussian filter
    - Convolution of a Laplacian and Gaussian filter
  - Gabor filter
    - σ for scale, the phase ϕ, orientation ω, and wavelength of the sine wave λ.

$$f_{mn} = \frac{1}{2\pi\sigma^2} \exp\left[-\frac{m^2+n^2}{2\sigma^2}\right] \sin\left[\frac{2\pi(\cos[\omega]m + \sin[\omega]n)}{\lambda} + \phi\right]$$

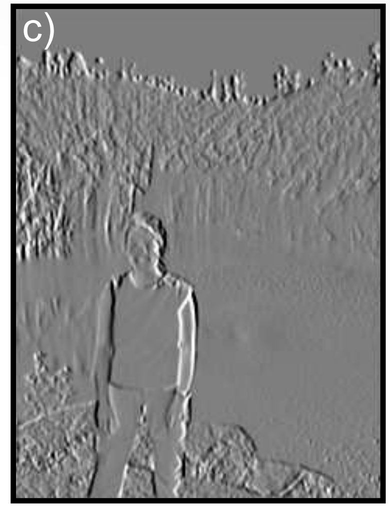


Original image



Prewitt (vertical)

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$



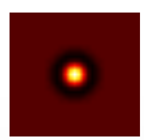
Prewitt (horizontal)

$$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

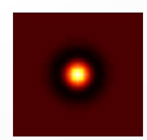
Laplacian

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Laplacian of Gaussian



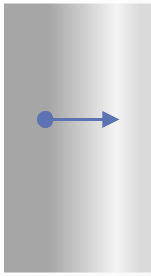
Difference of Gaussians



Edges, Corners, and Interest Points

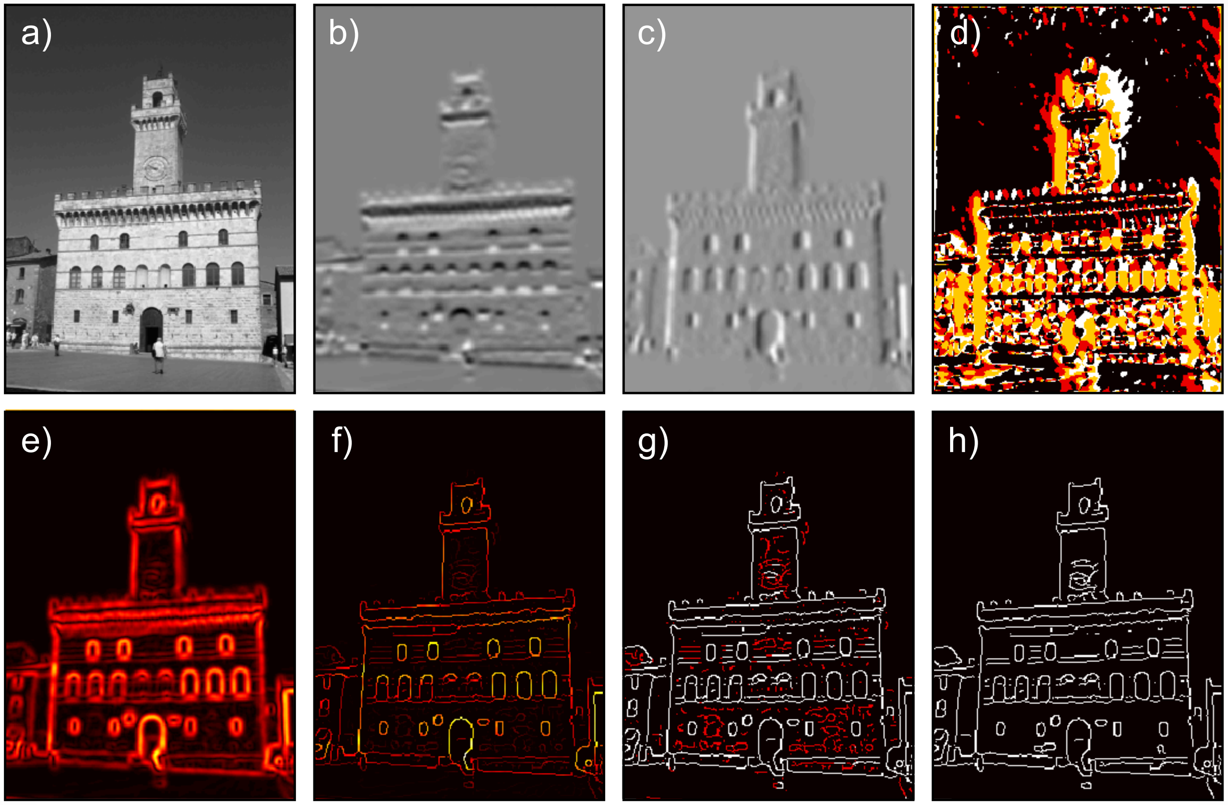
- Canny edge detection
    1. (Blured)
    2. Convoluted with a pair of orthogonal derivative filters such as Prewitt filters to create images H and V in the horizontal and vertical directions, respectively
    3. For pixel (i,j), the orientation θ_{ij} and magnitude a_{ij} of the gradient are computed as

$$\begin{aligned} \theta_{ij} &= \arctan[v_{ij}/h_{ij}] \\ a_{ij} &= \sqrt{h_{ij}^2 + v_{ij}^2}. \end{aligned}$$



    4. Orientation quantized into {0°, 45°, 90°, 135°}
    5. Non-local maximum suppression: magnitude of a pixel set to zero if if either of the neighboring two pixels perpendicular to the gradient have higher values
    6. Two thresholds: 1) pixels above the high threshold and 2) pixels above the low threshold but connect to existing edges are part of the edge

Example



Canny edge detection. a) Original image. b) Result of vertical Prewitt filter. c) Results of horizontal Prewitt filter. d) Quantized orientation map. e) Gradient amplitude map. f) Amplitudes after non-maximal suppression. g) Thresholding at two levels: the white pixels are above the higher threshold. The red pixels are above the lower threshold but below the higher one. h) Final edge map after hysteresis thresholding contains all of the white pixels from (g) and those red pixels that connect to them.

Harris Corner Detector

- Find points in the image where the image intensity is varying in both directions
  - h_{mn}, v_{mn} are the response to a horizontal and vertical derivative filter, respectively
  - w_{mn} weights → smaller away from the center of window (2D+1)x(2D+1)
  - Compute the image structure tensor (2-by-2 matrix) as follows:

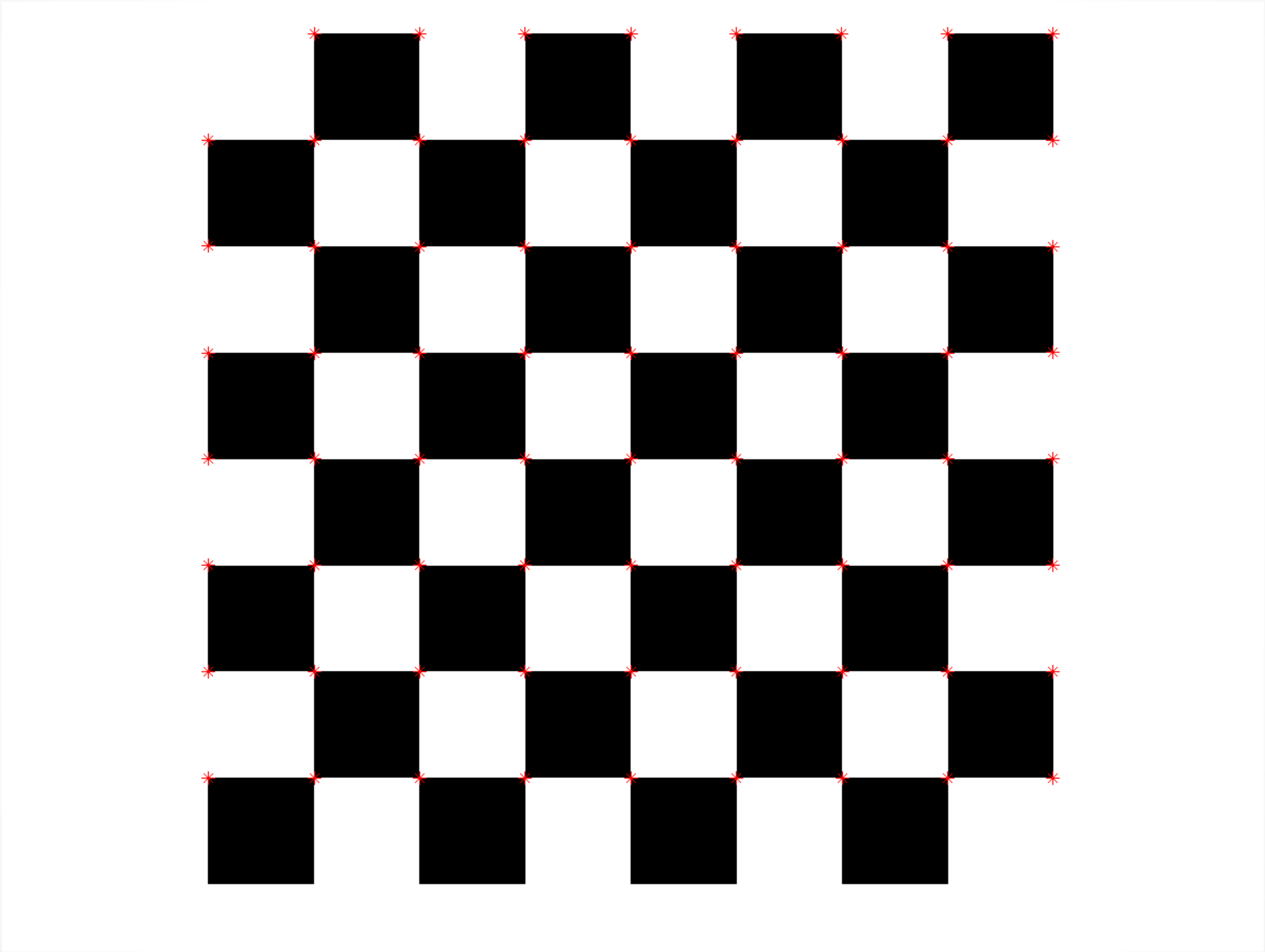
$$\mathbf{S}_{ij} = \sum_{m=i-D}^{i+D} \sum_{n=j-D}^{j+D} w_{mn} \begin{bmatrix} h_{mn}^2 & h_{mn}v_{mn} \\ h_{mn}v_{mn} & v_{mn}^2 \end{bmatrix}$$

  - Compute the singular values (κ between 0.04 to 0.15)

$$c_{ij} = \lambda_1\lambda_2 - \kappa(\lambda_1^2 + \lambda_2^2) = \det[\mathbf{S}_{ij}] - \kappa \cdot \mathrm{trace}[\mathbf{S}_{ij}]$$

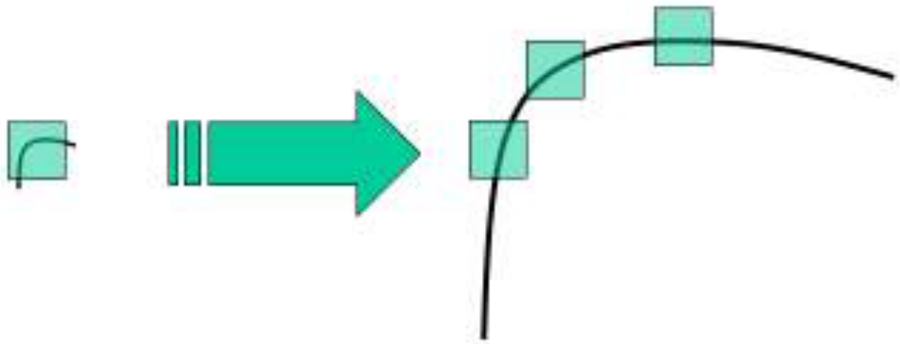
  - If c_{ij} is greater than a threshold → corner
  - Intuition: if singular values are both small → no change; if one singular value is large and the other is small → edge; otherwise, corner
  - Rotation invariant

Example



Scale Free Features

- Harris corner detector is sensitive to the scale of the image

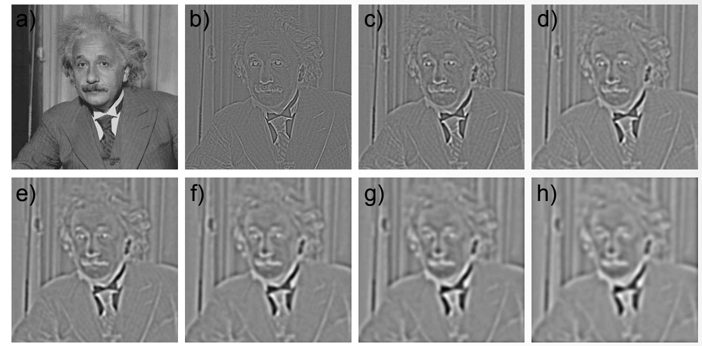


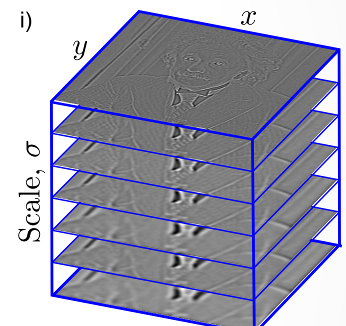
http://docs.opencv.org/master/da/df5/tutorial_py_sift_intro.html#gsc.tab=0

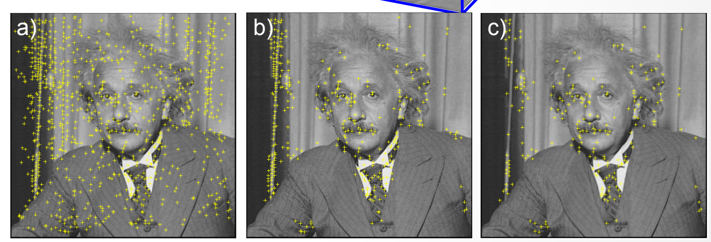
SIFT Detector

Scale invariant feature transform (SIFT) detector

1. The intensity image is filtered with a difference of Gaussian kernel at a series of K increasingly coarse scales
2. Local extrema are identified in the IxJxK volume as points who is either greater or smaller than all 26 neighbors in 3x3x3 block (voxel)
3. Apply quadratic approximation at each extrema point → finding the peak or trough at sub-pixel level
4. Apply Harris corner detector
5. The amplitude and orientation of local gradient in the regions surrounding interest points
6. A histogram is computed for the region and the peak of the histogram is assigned as the orientation of the orientation
7. Each interest point marked with an arrow with the orientation and its scale (SIFT descriptor)







(a) extrema (c) corner

Further Reading

- Chapter 13, 14, Simon J.D. Prince, Computer vision: models, learning and inference, Cambridge University Press (electronic version available from the author's web site)