

Lecture 1.1: Introduction

Typical CV Problems. ① *Segmentation Boundary detection, Clustering* (vector quantization, Lloyd alg., GPCA) ② *Recognition* ③ *Object / feature tracking* ④ *3D reconstruction* ⑤ *Structure from Motion*.
Why CV difficult? Viewpoint variation, Illumination (light change), Scale, Intra-class variation, Motion, Background clutter, Occlusion (Covered by ...)

Lecture 1.2: Image Filtering

Grayscale image. ① func. $f : \mathbb{R}^2 \mapsto \mathbb{R}$, gives intensity at (x, y) ② *Digital image*: grid of intensity values (0=black), discrete (sampled, quantized) version of f .
Filters. ① *Filtering*: form a new image whose pixel values are a function of the pixel values in input image ② *Why*: get useful information + enhance image ③ *Problem*: restoration (denoising, deblurring), compression, locating structural features, computing field properties.
Linear filtering. ① *Cross-correlation*: replace each pixel by a linear combination of its neighbors (using kernel / mask / filter H):

$$G[i, j] = \sum_{u=-k}^k \sum_{v=-k}^k H[u, v] F[i + u, j + v] \Leftrightarrow G = H \otimes F \tag{1}$$

② *Convolution*:

$$G[i, j] = \sum_{u=-k}^k \sum_{v=-k}^k H[u, v] F[i - u, j - v] \Leftrightarrow G = H * F \tag{2}$$

commutative: $-F * G = G * F$, associative: $-F * (G * I) = (F * G) * I$.

③ *Examples*: identical: $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$, left shifted: $\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$, blur / mean filter

(suppresses high-frequency signal): $\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$, sharpening: $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} -$

$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ ④ *Gaussian kernel*: $G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$, $\sigma \uparrow \rightarrow$ blur \uparrow , low-pass filter, removes high-frequency components, $G_1 * G_2 = G_3$. ⑤ *Sharpening*: detail = origin - blurred / smoothed, sharpened = origin + $\alpha \times$ detail.

Lecture 1.3: Edge Detection Part 1

Edge detection. ① *Goal*: identify visual changes (discontinuities) in an image. ② Intuitively, semantic information is encoded in edges. ③ *Characterizing*: edge is a place of big change in the image intensity function.
Edge detection = differentiate a digital image $F[x, y]$. ① *reconstruct* a continuous image. ② *finite difference / discrete derivative*: $\frac{\partial f}{\partial x}[x, y] \approx F[x + 1, y] - F[x, y]$

$$\frac{\partial f}{\partial x} : H_x = \begin{bmatrix} 0 & 0 & 0 \\ 1 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \frac{\partial f}{\partial y} : H_y = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{3}$$

Edge strength: $\|\nabla f\| = \sqrt{(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2}$, $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$, $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$,
gradient direction: $\tan^{-1} \left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right)$.

Lecture 1.3: Edge Detection Part 2

Noisy image. ① *Problem*: gradient will be noisy ② *Smooth*: $f \rightarrow f * h \rightarrow \frac{d}{dx}(f * h) = f * \frac{d}{dx}h$, where h is a gaussian kernel \rightarrow the edge is the **peak** of $\frac{d}{dx}(f * h)$.

1D Gaussian: $G_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$, $G'_\sigma(x) = \frac{d}{dx}G_\sigma(x) = -\frac{1}{\sigma} \left(\frac{x}{\sigma} \right) G_\sigma(x)$

2D Gaussian: $h_\sigma(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$

Sobel operator. approximation of derivative of Gaussian

$$S_x = \frac{1}{8} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad S_y = \frac{1}{8} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{1}$$

where $\frac{1}{8}$ is to get right gradient magnitude, make no impact on edge detection.
Laplacian of Gaussian (LoG). ① $\frac{\partial^2}{\partial x^2}(h * f) = (\frac{\partial^2}{\partial x^2}h) * f \rightarrow$ edge is the **zero-crossings** of $(\frac{\partial^2}{\partial x^2}h) * f$ ② *LoG using*: positive "+" on one side, negative "-" on the other side, zero just at the edge.

Good edge detector. ① *good detection*: find all real edges, ignoring noise or other artifacts ② *good localization*: detected edges must be as close as possible to the true edges + return one point only for each true edge point.

Canny edge detector. ① *Filtering*: use derivative of Gaussian ② *Find*: magnitude and orientation of gradient ③ *Non-maximum suppression*: check if pixel is local maximum along gradient direction for thinning ④ *Hysteresis thresholding and Linking*: define low (α) and high (β) thresholds to start edge curves and the low threshold to continue them (noise $< \alpha <$ weak edge $< \beta <$ strong edge) \rightarrow *Linking*: continue weak edges only when they are between strong edges.
Canny Characteristics. ① large σ fo Gaussian \rightarrow detects large scale edges, small $\sigma \rightarrow$ detects fine features ② the most widely used edge operator ③ gives single-pixel-wide images with good continuation between adjacent pixels ④ very sensitive to parameters, need to be adjusted for different applications.

Lecture 2.1: Image Sampling and Resizing

No exam about this subsection.

Lecture 2.2: Color and Texture Part 1

Color spaces. ① *RGB*: standard for cameras, $0 \sim 255$, normalized $r = \frac{R}{R+G+B}$ ② *HSI / HSV*: hue ($0 \sim 2\pi$), saturation ($0 \sim 1$), intensity / value ($0 \sim 1$) ③ *YIQ*: color TVs, Y is intensity ...

Color histogram. ① *gray histogram* H : $H[i]$ gives the count of pixels with gray tone i ② *normalized histogram* P : $P[i]$ is the percentage of pixels with gray tone i ③ *property*: 1. fast and easy to compute 2. invariant to 2D rotation (\times 3D/distance) 3. size can easily normalize \rightarrow compare different image histograms 4. match color histograms for database query & classification 5. no spatial info
Make color histogram. ① *Options*: 1. single histogram for 3D color vectors 2. 3 scalar histograms (\times) ② *opponent encoding*: $wb = R + G + B$ (lightness / white-black), $rg = R - G$ (red-green), $by = 2B - R - G$ (blue-yellow) ③ *histograms*: $8 \times 16 \times 16 = 2048$ ④ *match score*: normalized intersection $Match(h_I, h_M) = \frac{\sum \min(h_I[j], h_M[j])}{\sum h_M[j]}$ ⑤ *Multi-scale spatial color representation*: divide image into $N \times N$ grids \rightarrow compute color histogram for each grid \rightarrow concatenate all the histograms across grids as a feature vector.

Lecture 2.2: Color and Texture Part 2

Texture. ① *def*: description of the spatial arrangement of color/intensities in (selected region of) image ② *structural approach*: a set of texels in some regular or repeated pattern
Statistical texture. ① *case*: 1. segmenting out texels is difficult or impossible in natural images 2. numeric quantities or statistics that describe a texture can be computed from the gray tones (or colors) alone 3. less intuitive but computationally efficient 4. can be used for classification and segmentation
Statistical texture measures. ① *Edge-based*: 1. edgeness per unit area (measure of texture richness) $F = |\{p | \text{gradient magnitude}(p) \geq \text{threshold}\}| / N$ 2. edge magnitude and direction histograms (measure of texture uniqueness) $F = (H_{\text{magnitude}}, H_{\text{direction}})$ ② *local binary pattern (LBP)*: for each pixel p , create an 8-bit number b_1, \dots, b_8 where $b_i = 0$ if neighbor $i \leq p$ and 1 otherwise, $LBP(N_c) = \sum_{p=0}^{P-1} s(N_p - N_c) \cdot 2^p$, where N_c, N_p are center and neighbor pixel, r radius, $s(\cdot)$ is Sign function.

Co-occurrence Matrix Features. ① *co-occurrence matrix*: 2D array C where each position represents a possible image value, $C_d(i, j)$ indicates times of value i co-occurring with j in a particular spatial relationship / distance $d = (\Delta_x, \Delta_y)$ ② *normalized co-...* N_d : each value is divided by the sum of all the values of C_d

Filter banks. ① 4 Gaussian filters with $\sigma = \{1, \sqrt{2}, 2, 2\sqrt{2}\}$ ② 8 LOG with $\sigma = \{1, \sqrt{2}, 2, 2\sqrt{2}, 4, 3\sqrt{2}, 6, 6\sqrt{2}, 12\}$ ③ 18 x-directional *first* derivation of Gaussian filters with $\sigma_x = \sigma, \sigma_y = 3\sigma$ with $\sigma = \{1, \sqrt{2}, 2\}$ and 6 rotation orientations $\theta = \{0, \frac{\pi}{6}, \frac{\pi}{3}, \frac{\pi}{2}, \frac{2\pi}{3}, \frac{5\pi}{6}\}$ ④ 18 x-directional *second* derivation of ...

Lecture 3: Deep Learning

Linear methods. PCA, LDA, LR, $y = Wx, y = f(Wx)$.
Neural networks / MLP. limitation: an NN / MLP can only have at most 4 layers
Kernel methods. ① enjoy low computation complexity of linear methods and nonlinear representation capability ② *limitation*: can't cope with large dataset
Deep learning. ...

Lecture 4: Image Segmentation + Object Detection + Object Recognition

Why context is important? ① can be treated as prior probability ② to "guess" small / blurry objects based on a prior ③ narrows down the search space.
Semantic segmentation. ① *goal*: label each pixel in the image with a category label ② *general method*: design network as a bunch of convolutional layers, with downsampling (pooling, strided convolution) and upsampling inside CNN.
Object detection as regression. ① *problem*: each image needs a different number of outputs ② *sliding window*: apply a CNN to many different crops of image then classify each crop as object or background \rightarrow *problem*: huge number of locations and scales, very computationally expensive ③ *region proposals*: find image regions that are likely to contain objects
R-CNN. 1. extract region proposals ($\sim 2k$) / crop 2. scale to fixed size 3. forward propagate / extract features by different CNNs 4. object proposal refinement by linear reg on CNN features. **Fast R-CNN.** 1. forward image through one CNN 2. extract region of interest (Rols) from proposal method (RoI Polling layer) 3. classify. **Faster R-CNN.** ① *idea*: CNN do proposals ② *method*: insert Region Proposal Network (RPN) to predict proposals from features
YOLO (You Only Look Once). ① *idea*: treat detection as regression problem, rather than classification ② *steps*: 1. divide image into a grid of size $S \times S$ 2. each grid predicts B bounding boxes and their confidence scores, while also predicting class probabilities 3. non-maximum suppression to remove duplicate boxes.

Lecture 5.1: Harris Corner Detection

Motivation. ① *panorama stitching*: combine images, 1. detect points of interest 2. extract features 3. match features 4. align images ② *image matching* ③ *feature matching* ④ *motion tracking* ⑤ *3D reconstruction* ⑥ *robot navigation*.

Invariant local features. features that are invariant to transformations ① *geometric invariance*: translation, rotation, scale ② *photometric invariance*: brightness, exposure ③ *Advantage*: 1. *locality*: robust to occlusion and clutter 2. *quantity*: hundreds or thousands in a single image 3. *distinctiveness*: can differentiate a large database of objects 4. *efficiency*: real-time performance achievable ④ *main components*: 1. *detection*: identify points of interest 2. *description*:: extract vector feature descriptor surrounding each point of interest 3. *matching*: determine correspondence between descriptors in two views.

Local measures of uniqueness. ① *motivation*: good feature is "unique" ② *flat region*: no change in all directions ③ *edge*: no change along the edge direction ④ *corner*: significant change in all directions.

Harris corner detection. ① *Sum of the squared differences (SSD)* (with widow W):

$$E(u, v) = \sum_{(x, y) \in W} [I(x + u, y + v) - I(x, y)]^2. \quad (4)$$

② *small motion assumption*: Taylor expansion: $I(x + u, y + v) \approx I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v \approx I(x, y) + \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \Rightarrow$ ③ *SSD error*: $E(u, v) \approx \sum_{(x, y) \in W} [I_x u + I_y v]^2 = \begin{bmatrix} u & v \end{bmatrix} H \begin{bmatrix} u \\ v \end{bmatrix} = Au^2 + 2Buv + Cv^2, A = \sum_{(x, y) \in W} I_x^2, B = \sum_{(x, y) \in W} I_x I_y, C = \sum_{(x, y) \in W} I_y^2, H = \begin{bmatrix} A & B \\ B & C \end{bmatrix}$ is **Harris operator** ④ $E(u, v)$ is locally approximated as a quadratic function of displacement (u, v) ⑤ *example*: 1. horizontal edge $I_x = 0, H = \begin{bmatrix} 0 & 0 \\ 0 & C \end{bmatrix}$ 2. vertical edge $I_y = 0, H = \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix}$ ⑥ *practice*: $H = \sum_{(x, y) \in W} w_{x, y} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$ where w based on its distance from center pixel.

Eigenvalues of H . ① *edge*: $\lambda_1 \gg \lambda_2 / \lambda_1 \ll \lambda_2 \Leftrightarrow R = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2$ is large ② *corner*: $\lambda_1 \sim \lambda_2$, are large $\Leftrightarrow R$ is negative with large magnitude ($R \ll 0$) ③ *flat*: $\lambda_1 \sim \lambda_2$, but small $\Leftrightarrow R$ is small ($R \approx 0$) ④ *Harris response calculation*: is R and $k \in [0.04, 0.06]$.

Harris operator variant. $\lambda_{\min} \approx f = \frac{\lambda_1 \lambda_2}{(\lambda_1 + \lambda_2)} = \frac{\det(H)}{\text{tr}(H)}$.

Corner detection summary. ① compute gradient at each point; ② create H (with Gaussian filter) ③ compute eigenvalues ④ find $\lambda_{\min} > \text{threshold}$ ⑤ choose points where λ_{\min} is a local maximum as features.

Parameters of Harris Corner Detection. threshold (✓) window size (✓) σ for Gaussian filter (✓) value of (u, v) (✗).

Lecture 5.2: Harris Corner Detection

Harris Invariance property. ① *translation*: because derivatives are shift-invariant ② *rotation* ③ *affine intensity change*: $I \leftarrow aI + b$, because derivatives are invariant to $I + b$ and maximum is invariant to aI . ④ *scaling*: **not invariant to scaling**.

Scale invariant detection. ① *idea*: find scale that gives local maximum of f in both position and scale ② *definition of f* : 1. Harris 2. *Laplacian of Gaussian (LoG)* $\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$ ③ *implementation*: variable window + fixed figure \rightarrow fixed window W + Gaussian pyramid (variant figure).

Blob detector. ① find maxima and minima of *LoG* operator in space and scale ② find local maxima in position-scale space with Gaussian Blurring $g(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$ with different scale σ and position list (x, y, σ) .

How to match good points? *answer*: come up with a descriptor for each point, find similar descriptors between the two images using like *SIFT* and matching square windows around the point.

Lecture 6.1: Scale Invariant Feature Transform (SIFT)

DoG. *Difference of Gaussians* to approx *LoG*, $\sigma \Delta^2 G = \frac{\partial G}{\partial \sigma} = \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma} \Rightarrow G(x, y, k\sigma) - G(x, y, \sigma) \approx (k - 1)\sigma^2 \Delta^2 G$, typically $\sigma = 1.6, k = \sqrt{2}$.

SIFT. ① *building scale space*: Gaussian blurs of different $k \rightarrow$ DoG ② *peak detection*: find a peak in the scale space to localize a key point, i.e. compare pixel x in current (8 pixels) and adjacent ($2 \times 9 = 18$) scales, select x if $x \geqslant / \leqslant$ all 26 pixels ③ *orientation assignment*: 1. compute the gradient magnitude and direction of each pixel at the scale of the keypoint

$$m(x, y) = \sqrt{(L(x + 1, y) - L(x - 1, y))^2 + (L(x, y + 1) - L(x, y - 1))^2} \\ \theta(x, y) = \tan^{-1}((L(x, y + 1) - L(x, y - 1)) / (L(x + 1, y) - L(x - 1, y))),$$

where $L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$. 2. construct an **orientation histogram** to assign an orientation to the key point ④ *SIFT / key point descriptor*: 1. divide 16×16 window into 4×4 grid of cells, each cell contains $4 \times 4 = 16$ pixels 2. compute an orientation histogram (8 bins) for each cell (weighted by the gradient magnitude) 3. get $16(\text{cells}) \times 8(\text{bins}) = 128$ dimensional descriptor.

Feature matching for features I_1, I_2 . ① *idea*: 1. define distance function that compares two descriptors 2. test all the features in I_2 , find the one with min distance ② *distance*: 1. L_2 / *SSD*: $\|f_1 - f_2\|^2$, but can give good scores to ambiguous (incorrect) matches; 2. ratio distance $\frac{\|f_1 - f_2\|^2}{\|f_1 - f'_2\|^2}$ where f_2 is best *SSD* match to f_1 in I_2 and f'_2 is 2nd best, can give small values to ambiguous matches.

Lecture 6.2: Bag-of-Words Feature

① *Motivation*: brute force approach is computational consuming (using geometric transformation and lighting adjustment for the whole image) ② *Idea*: use global similarity measure instead of pixel-level or feature-point matching

Bag of Words (BoW). ① *extract features*: like patches/blobs, basic patterns from a set of images ② *learn visual vocabulary*: use *K-means (Lloyd)* to learn K clusters, centers are Visual Words ③ *quantization*: map each blob to a visual word in the visual vocabulary according to distance ④ *frequency histogram*: represent images by frequencies of visual words.

Weighting. ① *Idea*: some visual words are more discriminative than others (but "and, or, is" are not) ② *TF (Term Frequency)*: just histogram values (✗) ③ *IDF (Inverse Document Frequency)*: $IDF_j = \log \frac{\# \text{document}}{\# \text{documents include } j}$ ④ *TF-IDF*: Histogram bin value = TF \times IDF.

Inverted file. ① *motivation*: dictionary is very large thus histogram of a single image is sparse ② *idea*: instead of storing "images \rightarrow word list", store "words \rightarrow image list".

Spatial pyramid. ① *problem of BoW*: disregards the spatial layout of visual words ② *solution*: compute a histogram in each spatial region

Lecture 7.1: Geometric Transformations Part 1

Image warping. ① *image filtering*: change the brightness of an image $g(x) = h(f(x))$ ② *image warping*: change position of each point in an image $g(x) = f(h(x))$ ③ *forward warping*: calculate new position $(x', y') = T(x, y)$ for every $(x, y) \rightarrow$ fill color $g(x', y') = f(x, y)$ ④ *problem of forward*: can result in holes ⑤ *inverse warping*: calculate source position $(x, y) = T^{-1}(x', y')$ for every $(x', y') \rightarrow$ fill color $g(x', y') = f(x, y)$ ⑥ *interpolation*: use interpolation to generate a continuous image from a discrete image when (x, y) is not on a grid.

Parametric (global) warping. ① *transformation*: T is a coordinate-changing machine $p' = T(p)$ ② *global*: same for any point p + described by a few parameters

Lecture 7.1: Geometric Transformations Part 2

Linear transformation. ① $p' = Tp, \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$ ② *rotation* θ : $T = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$ ③ *mirror*: about y-axis ($x' = -x$): $\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$, about $y = x$: $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ ④ *scale*: $\begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$ ⑤ *translation* $x' = x + t$: ✗ because it is addition but not linear multiplication.

Homogeneous coordinates. ① *idea*: add one more coordinate to unified handling of translation and linear transformation $(x, y) \Leftrightarrow (x, y, 1)$ ② *converting*:

$$(x, y, w) \Leftrightarrow (x/w, y/w) \text{ ③ translation: } \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix}.$$

Affine transformation. ① *def*: any transformation with last row $[0, 0, 1]$ ② *translate*: $\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$ ③ *scale*: $\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$ ④ *shear*: $\begin{bmatrix} 1 & sh_x & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

$$\text{⑤ 2D in-plane rotation: } \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}.$$

$$\text{⑥ Projective Transformation / Homography. ① homography: } H = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix}.$$

Lecture 7.2: Image Alignment: Regression

Image alignment problem. given a set of matches between images A and B, find transform T that best "agrees" with the matches.

Translation (simple case). ① *mean displacement*: $(x_t, y_t) = (\frac{1}{n} \sum_{i=1}^n (x'_i - x_i), \frac{1}{n} \sum_{i=1}^n (y'_i - y_i))$ ② *overdetermined*: $2n$ equations for n points, 2 unknown.

Least squares. ① *residual*: $r_{x_i}(x_t) = (x_i + x_t) - x'_i, r_{y_i}(y_t) = (y_i + y_t) - y'_i$ ② *goal*: minimize sum of squared residuals $C(x_t, y_t) = \sum_{i=1}^n (r_{x_i}(x_t)^2 + r_{y_i}(y_t)^2) \Leftrightarrow$ mean displacement ③ *matrix form*: $\min ||At - b||^2 \Rightarrow A^T A t = A^T b \Rightarrow t = (A^T A)^{-1} A^T b$.

Image alignment algorithm. ① compute image features for image A and B ② match features between A and B ③ compute homography between A and B using least squares on set of matches.

RANSAC. ① *motivation*: outlier problem ② *alg*: 1. randomly choose s samples, typically s = minimum sample size that lets you fit a model; 2. fit a model (e.g., line) to those samples; 3. count number of inliers that approximately fit the model; 4. Repeat N times; 5. choose model that has the largest set of inliers.

Lecture 8: Camera Model and Image Formation

Pinhole camera model. ① *idea*: allow only a small amount of light to pass through to get an inverted image on the imaging plane ② *math*: $r = (x, y, z), r' = (x', y', z'), \frac{r'}{f'} = \frac{r}{z} \xrightarrow{\text{Similar Triangles Principle}} \frac{x'}{f'} = \frac{x}{z}, \frac{y'}{f'} = \frac{y}{z}$, where f is focal length, $\Rightarrow p' = (x', y') = (f \frac{x}{z}, f \frac{y}{z})$.

Lens. a lens focuses light onto the film: some objects are "in focus", other points project to a "circle of confusion" in the image.

Camera parameters. ① *coordinate system*: 1. World coordinate system 2. Camera coordinate system ② *goal*: project a point in world coordinates into camera **Step 1, World \rightarrow Camera**.