Recitation 9

Exam preparation

Exam details

- Monday, June 3
- 10:30am to 12:20pm
- Location: G20

Exam overview

- 100 points in total
- 24 points for 12 multiple choice
- 11 points for 11 true false
- 30 points for Segmentation and camera projection
- 15 points for PCA and LDA
- 20 points for video and deep learning
- maybe 10 points extra credit

The goal of computer vision

- To bridge the gap between pixels and "meaning"

What we see

0	3	2	5	4	7	6	9	8
3	0	1	2	3	4	5	6	7
2	1	0	3	2	5	4	7	6
5	2	3	0	1	2	3	4	5
4	3	2	1	0	3	2	5	4
7	4	5	2	3	0	1	2	3
6	5	4	3	2	1	0	3	2
9	6	7	4	5	2	3	0	1
8	7	6	5	4	3	2	1	0

What a computer sees

Source: S. Narasimhan

Histogram

Slide credit: Dr. Mubarak Shah

Convolution and Cross-correlation

Convolution is an integral that expresses the amount of overlap of one function as it is shifted over another function

$$(f*h)[m,n]=\sum_{k,l} f[k,l]\,h[m-k,n-l]$$

Cross-correlation compares the *similarity of two sets of data*

$$(f \star g)[m,n]=\sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty} f[i,j]\cdot g[i-m,j-n]$$

Convolution with Impulse function

$$f[n,m]=\sum_{k=-\infty}^{\infty}\sum_{l=-\infty}^{\infty} f[k,l]\,\delta_2[n-k,m-l]$$

Properties of systems

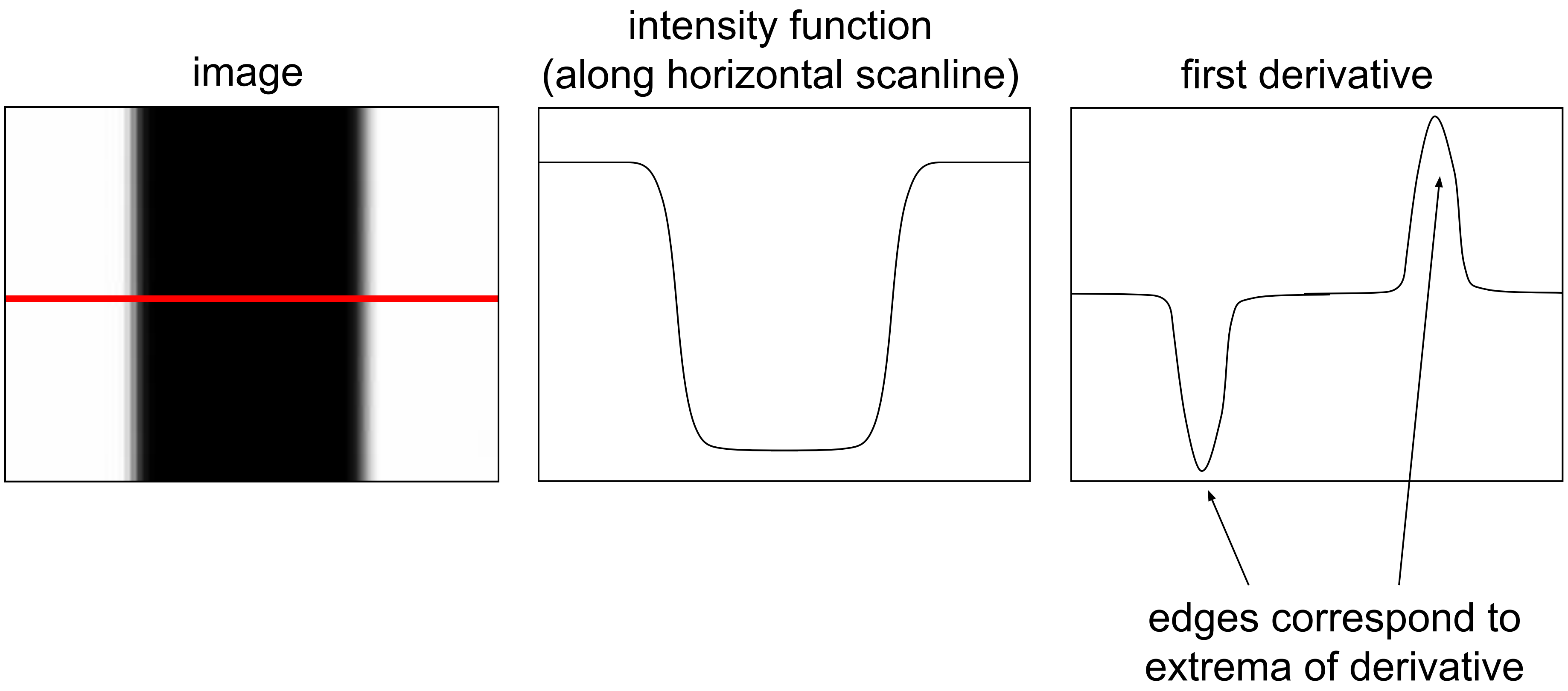
- Amplitude properties:
  - Additivity $$S[f_i[n,m] + f_j[n,m]] = S[f_i[n,m]] + S[f_j[n,m]]$$
  - Homogeneity $$S[\alpha f_i[n,m]] = \alpha S[f_i[n,m]]]$$
  - Superposition $$S[\alpha f_i[n,m] + \beta f_j[n,m]] = \alpha S[f_i[n,m]] + \beta S[f_j[n,m]]$$
  - Stability $$|f[n,m]| \leq k \implies |g[n,m]| \leq ck$$
  - Invertibility $$S^{-1}[S[f_i[n,m]]] = f[n,m]$$

Properties of systems

- Spatial properties
  - Causality $\text{for } n < n_0, m < m_0, \text{ if } f[n,m] = 0 \implies g[n,m] = 0$
  - Shift invariance: $f[n - n_0, m - m_0] \xrightarrow{\mathcal{S}} g[n - n_0, m - m_0]$

Characterizing edges

- An edge is a place of rapid change in the image intensity function



Discrete derivative in 2D

Given function $f(x, y)$

Gradient vector $$\nabla f(x, y) = \begin{bmatrix} \dfrac{\partial f(x, y)}{\partial x} \\ \dfrac{\partial f(x, y)}{\partial y} \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \end{bmatrix}$$

Gradient magnitude $$|\nabla f(x, y)| = \sqrt{f_x^2 + f_y^2}$$

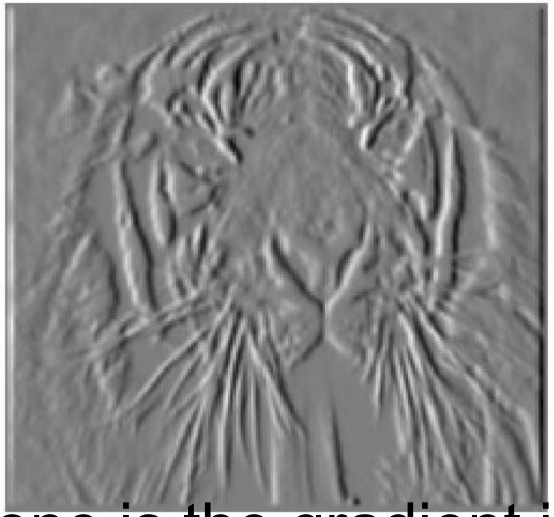
Gradient direction $$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

Finite differences: example

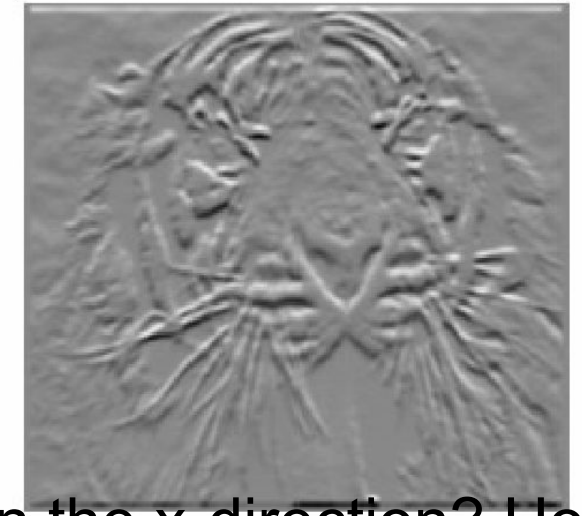
Original Image

Gradient magnitude

x-direction



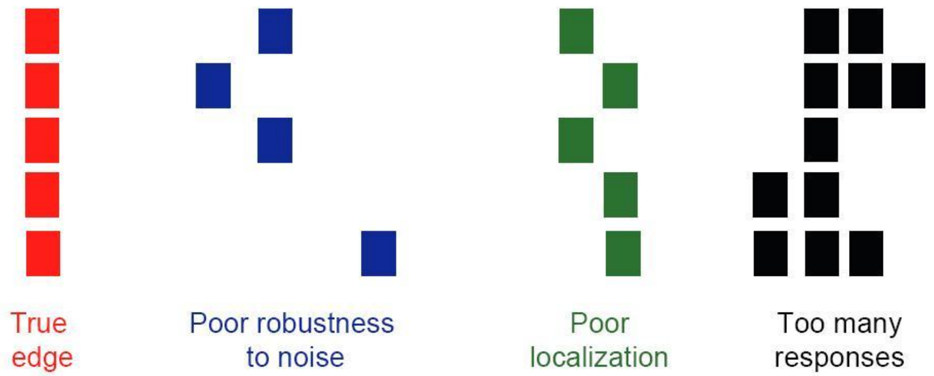
y-direction



- Which one is the gradient in the x-direction? How about y-direction?

Designing an edge detector

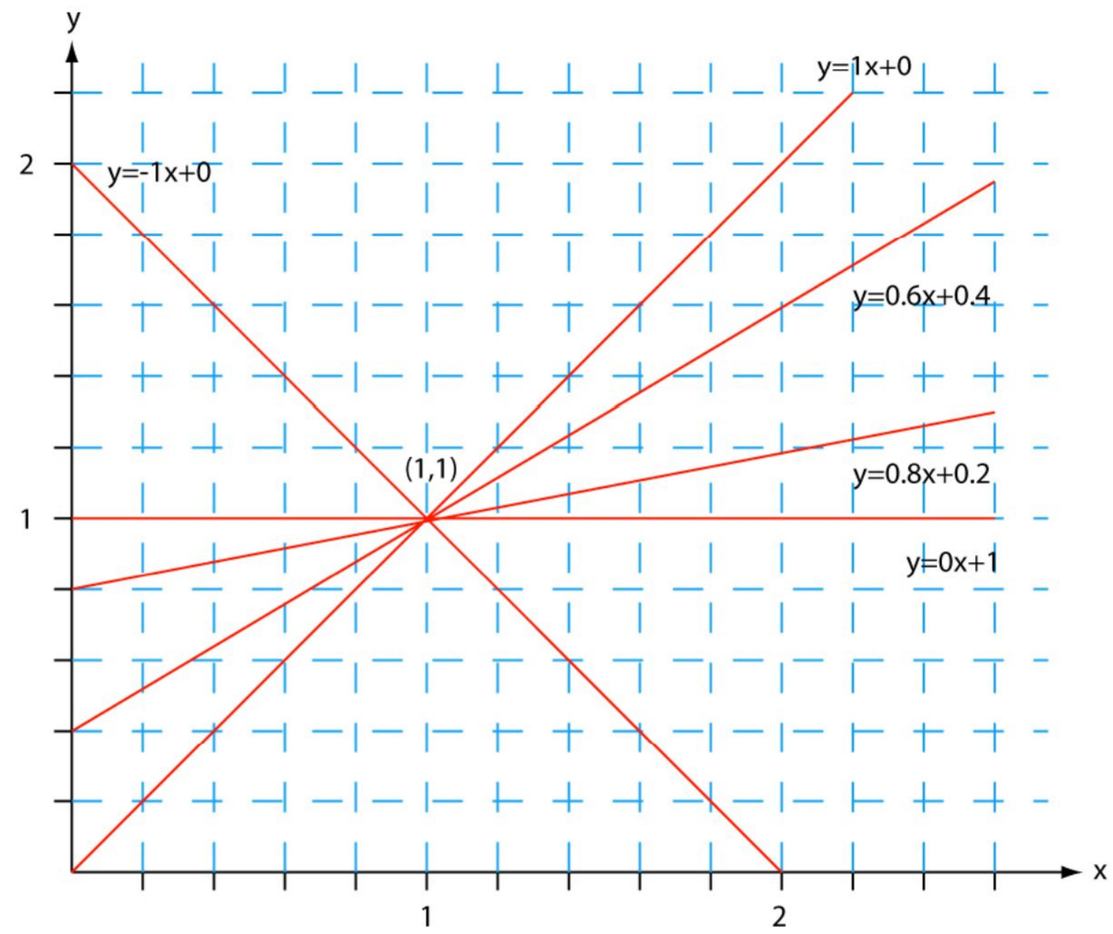
- Criteria for an “optimal” edge detector:
  - **Good detection:** the optimal detector must minimize the probability of false positives (detecting spurious edges caused by noise), as well as that of false negatives (missing real edges)
  - **Good localization:** the edges detected must be as close as possible to the true edges
  - **Single response:** the detector must return one point only for each true edge point; that is, minimize the number of local maxima around the true edge



Canny edge detector

- Suppress Noise
- Compute gradient magnitude and direction
- Apply Non-Maximum Suppression
  - Assures minimal response
- Use hysteresis and connectivity analysis to detect edges

Detecting lines using Hough transform



RANSAC: Pros and Cons

- **Pros**:
  - General method suited for a wide range of model fitting problems
  - Easy to implement and easy to calculate its failure rate
- **Cons**:
  - Only handles a moderate percentage of outliers without cost blowing up
  - Many real problems have high rate of outliers (but sometimes selective choice of random subsets can help)
- A voting strategy, The Hough transform, can handle high percentage of outliers

Requirements for keypoint localization

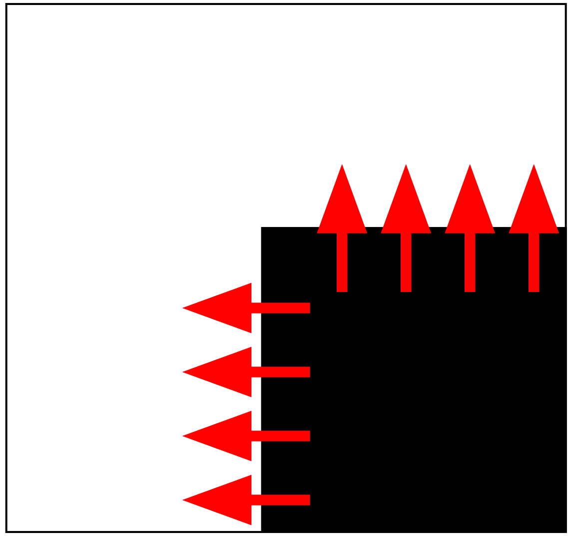
- Region extraction needs to be repeatable and accurate
  - Invariant to translation, rotation, scale changes
  - Robust or covariant to out-of-plane (≈affine) transformations
  - Robust to lighting variations, noise, blur, quantization
- Locality: Features are local, therefore robust to occlusion and clutter.
- Quantity: We need a sufficient number of regions to cover the object.
- Distinctivenes : The regions should contain “interesting” structure.
- Efficiency: Close to real-time performance.

Slide credit: Bastian Leibe

Harris corner detector and second moment matrix

- First, let's consider an axis-aligned corner:

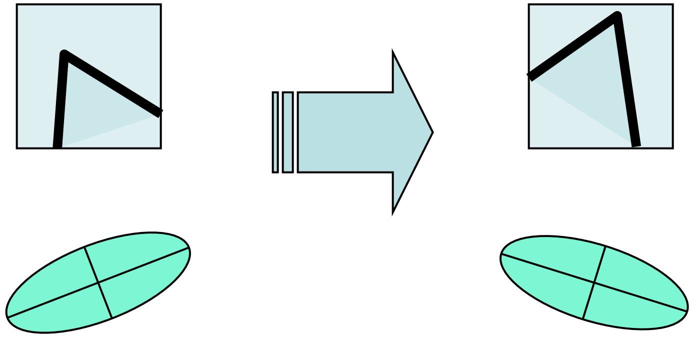
$$M = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

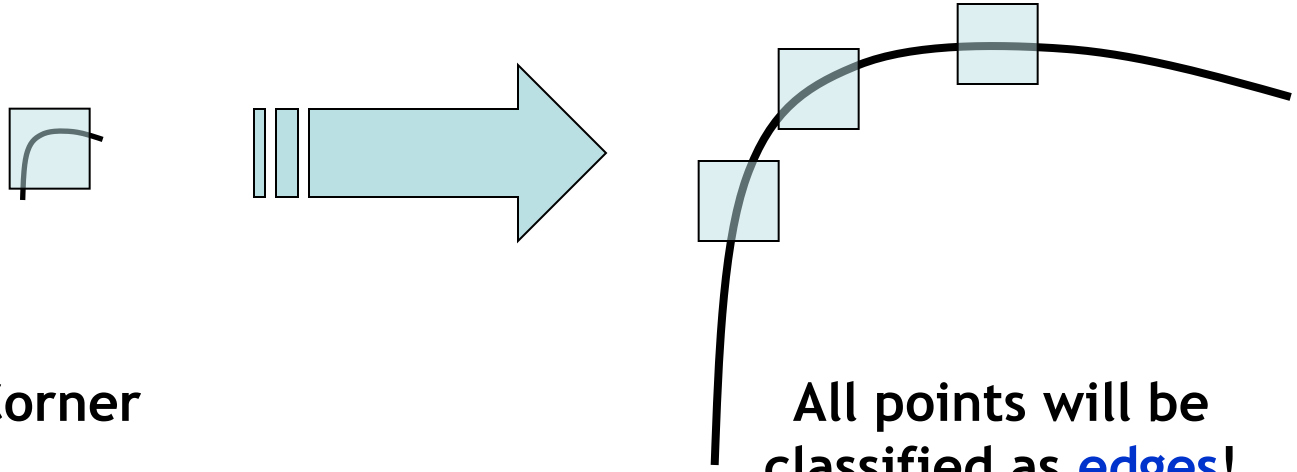


Slide credit: David Jacobs

Harris Detector: Properties

- Translation invariance
- Rotation invariance
- Scale invariance?





Corner

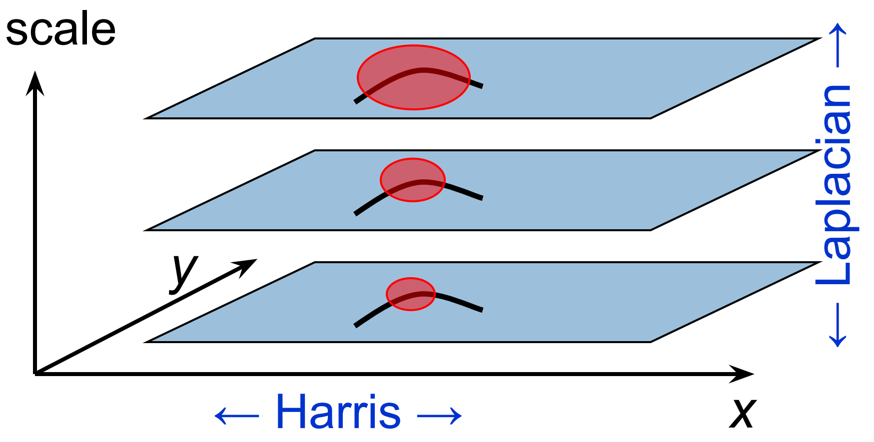
All points will be classified as **edges**!

Not invariant to image scale!

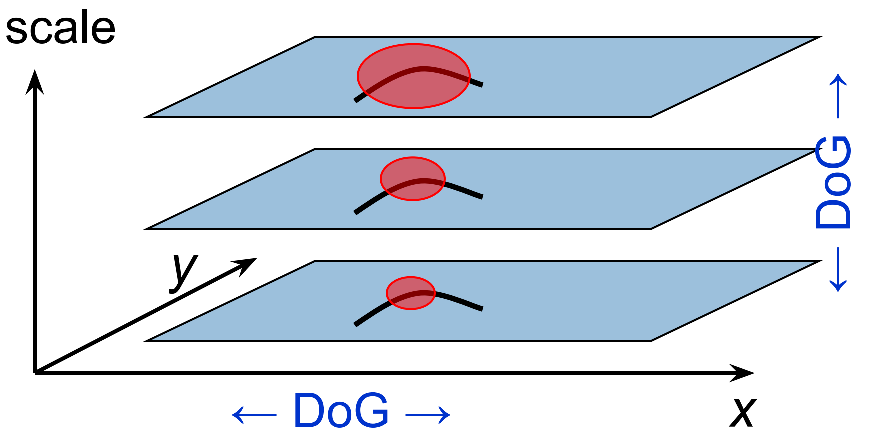
Slide credit: Kristen Grauman

Scale Invariant Detectors

- Harris-Laplacian[1]
  Find local maximum of:
  - Harris corner detector in space (image coordinates)
  - Laplacian in scale

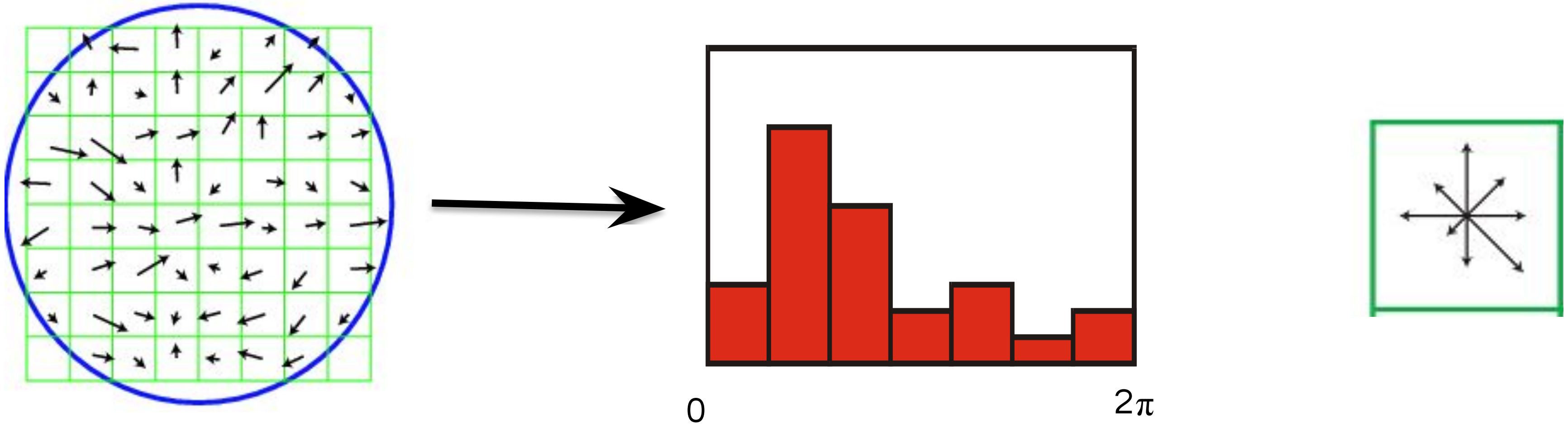


- SIFT (Lowe)[2]
  Find local maximum of:
  - Difference of Gaussians in space and scale



[1] K.Mikolajczyk, C.Schmid. “Indexing Based on Scale Invariant Interest Points”. ICCV 2001
[2] D.Lowe. “Distinctive Image Features from Scale-Invariant Keypoints”. IJCV 2004

SIFT descriptor formation



- Using precise gradient locations is fragile. We'd like to allow some "slop" in the image, and still produce a very similar descriptor
- Create array of orientation histograms (a 4x4 array is shown)
- Put the rotated gradients into their local orientation histograms
  - A gradients's contribution is divided among the nearby histograms based on distance. If it's halfway between two histogram locations, it gives a half contribution to both.
  - Also, scale down gradient contributions for gradients far from the center
- The SIFT authors found that best results were with 8 orientation bins per histogram.

Difference between HoG and SIFT

- HoG is usually used to describe entire images. SIFT is used for key point matching
- SIFT histrograms are oriented towards the dominant gradient. HoG is not.
- HoG gradients are normalized using neighborhood bins.
- SIFT descriptors use varying scales to compute multiple descriptors.

Seam Carving

- Assume m x n m x n’, n’<n   (summarization)

- Basic Idea: remove unimportant pixels from the image
  - Unimportant = pixels with less “energy”

$$E_1(\mathbf{I}) = |\frac{\partial}{\partial x}\mathbf{I}| + |\frac{\partial}{\partial y}\mathbf{I}|.$$

- Intuition for gradient-based energy:
  - Preserve strong contours
  - Human vision more sensitive to edges – so try remove content from smoother areas
  - Simple enough for producing some nice results

Gestalt Factors

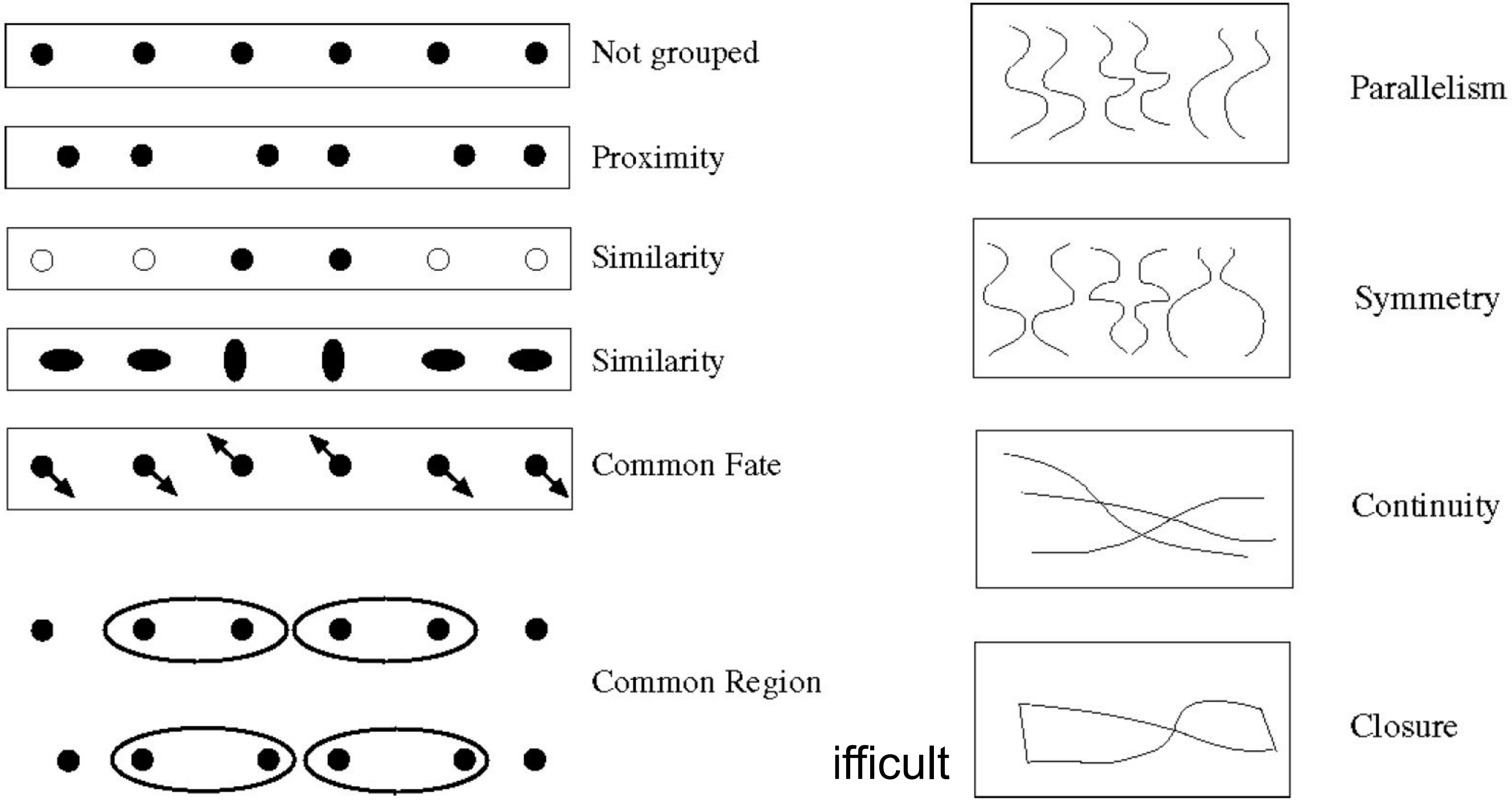


Image source: Forsyth & Ponce

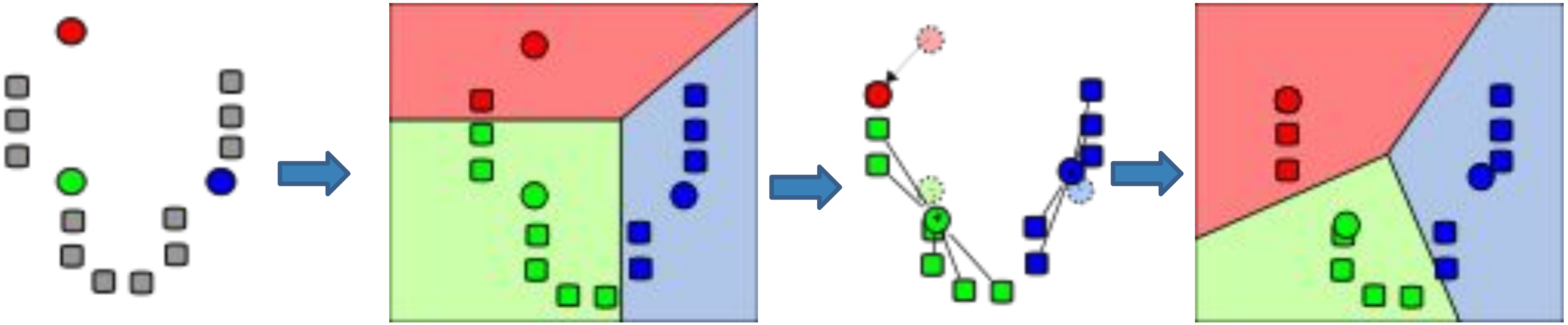
- These fac ifficult

Conclusions: Agglomerative Clustering

Good
- Simple to implement, widespread application.
- Clusters have adaptive shapes.
- Provides a hierarchy of clusters.
- No need to specify number of clusters in advance.

Bad
- May have imbalanced clusters.
- Still have to choose number of clusters or threshold.
- Does not scale well. Runtime of $O(n^3)$.
- Can get stuck at a local optima.

K-means clustering



1. Initialize Cluster Centers | 2. Assign Points to Clusters | 3. Re-compute Means | Repeat (2) and (3)

- Java demo:

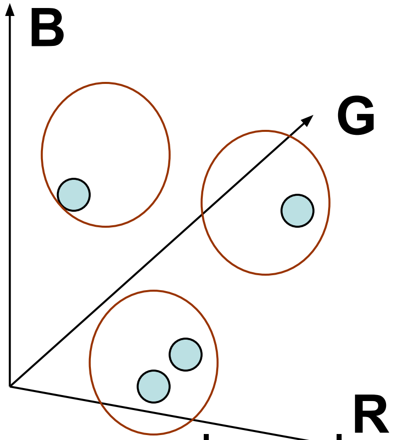
http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/AppletKM.html

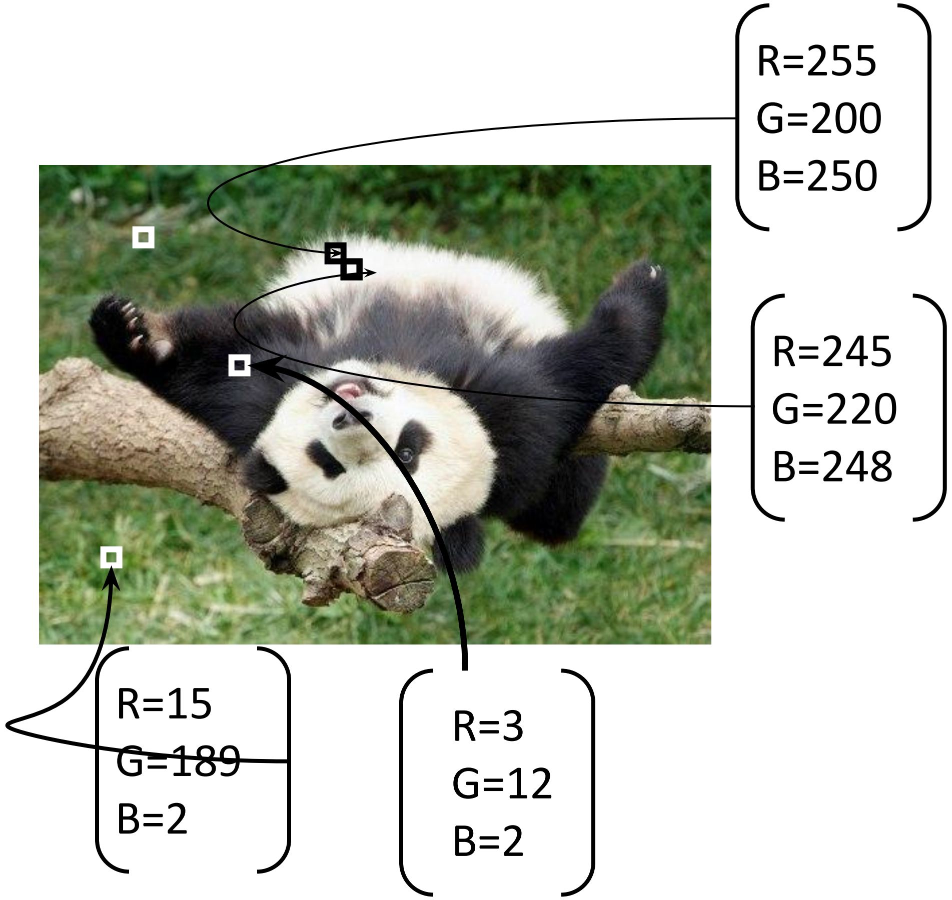
Illustration Source: Wikipedia

Feature Space

- Depending on what we choose as the *feature space*, we can group pixels in different ways.

- Grouping pixels based on color similarity





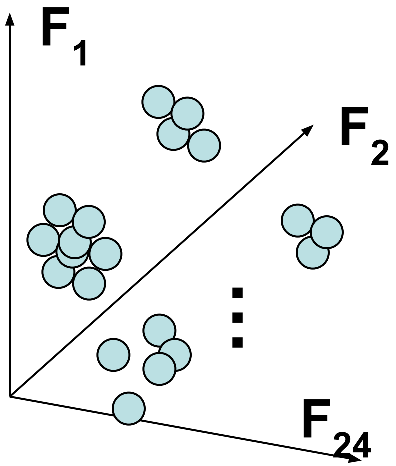
- Feature space: color value (3D)

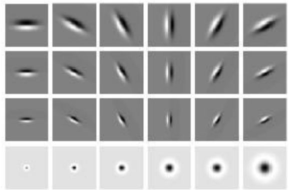
Slide credit: Kristen Grauman

Feature Space

- Depending on what we choose as the *feature space*, we can group pixels in different ways.

- Grouping pixels based on texture similarity





Filter bank of 24 filters

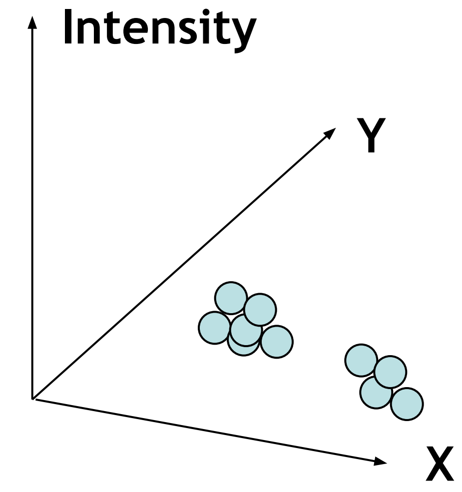
- Feature space: filter bank responses (e.g., 24D)

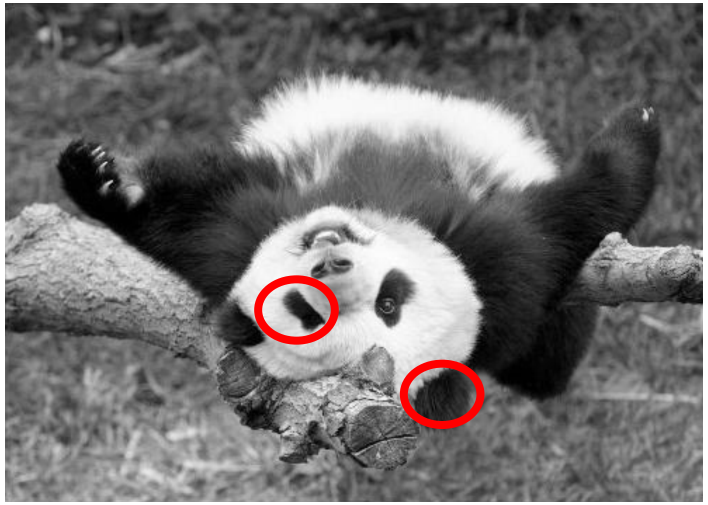
Slide credit: Kristen Grauman

Segmentation as Clustering

- Depending on what we choose as the *feature space*, we can group pixels in different ways.

- Grouping pixels based on *intensity+position* similarity



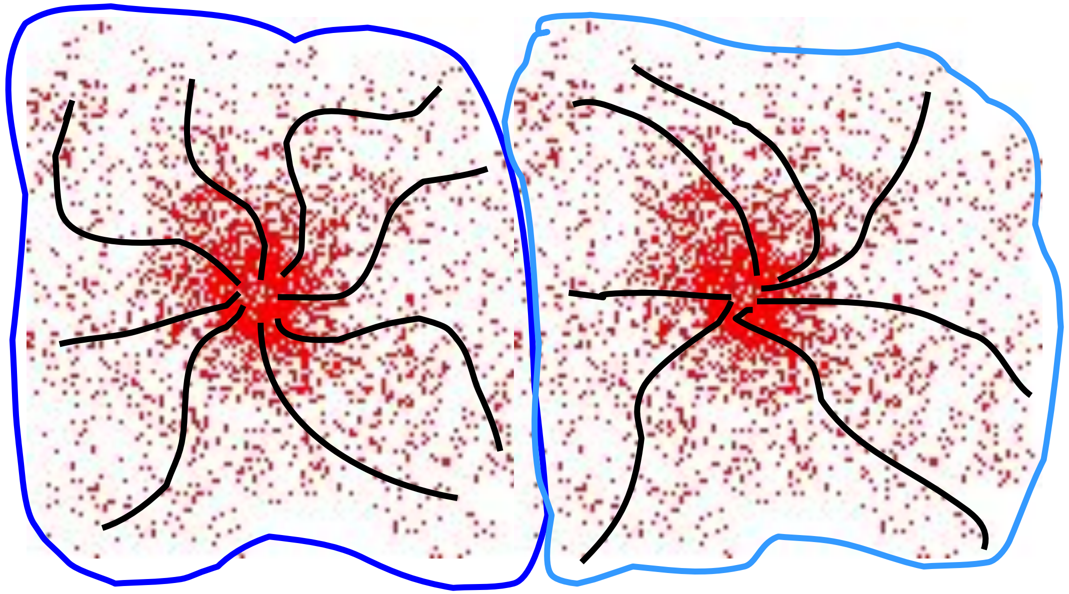


⇒ Way to encode both *similarity* and *proximity.*

Slide credit: Kristen Grauman

Mean-Shift Clustering

- Cluster: all data points in the attraction basin of a mode
- Attraction basin: the region for which all trajectories lead to the same mode



Slide by Y. Ukrainitz & B. Sarel

Summary Mean-Shift

- Pros
  - General, application-independent tool
  - Model-free, does not assume any prior shape (spherical, elliptical, etc.) on data clusters
  - Just a single parameter (window size h)
    - h has a physical meaning (unlike k-means)
  - Finds variable number of modes
  - Robust to outliers
- Cons
  - Output depends on window size
  - Window size (bandwidth) selection is not trivial
  - Computationally (relatively) expensive (~2s/image)
  - Does not scale well with dimension of feature space

Slide credit: Svetlana Lazebnik

12/6/19

The machine learning framework

$$\mathbf{y} = f(\mathbf{x})$$

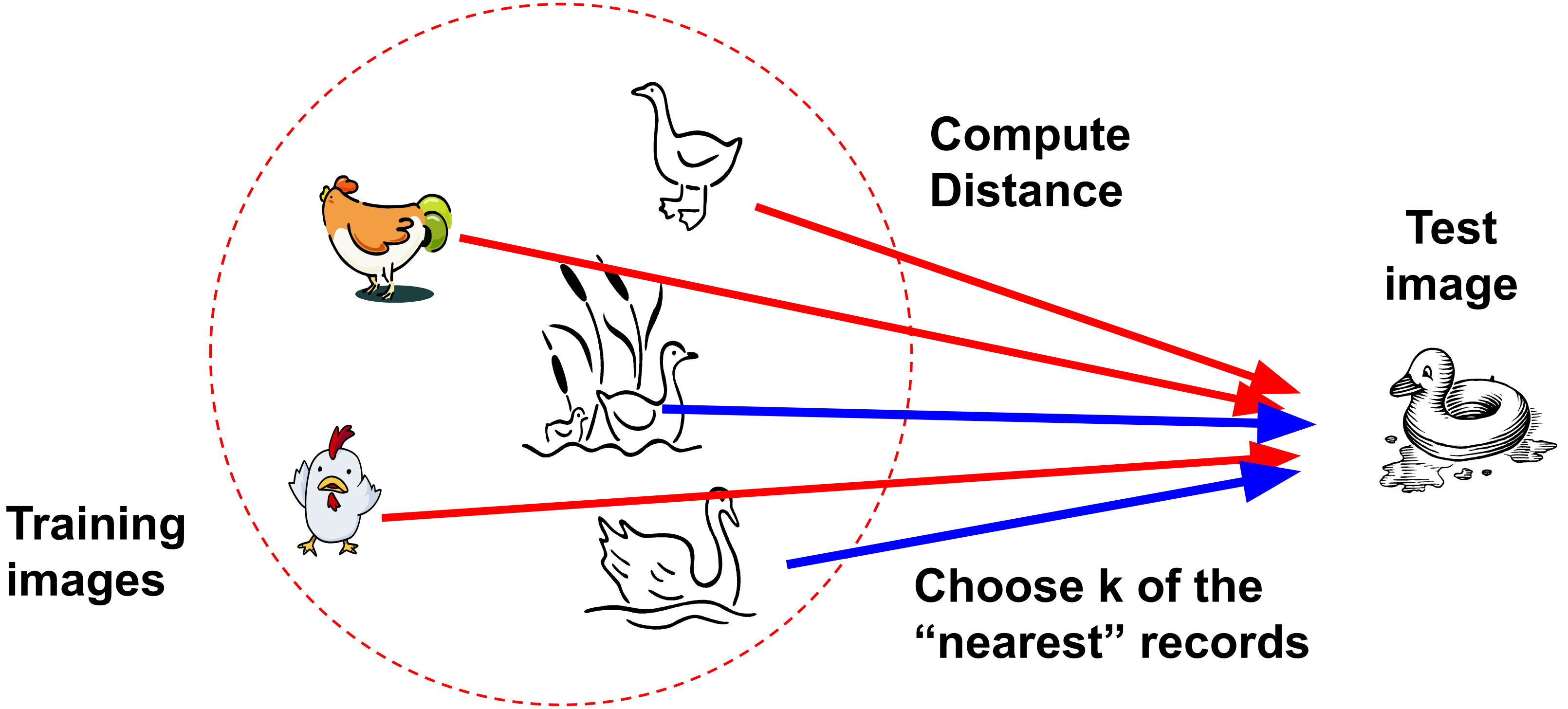
output | prediction function | Image feature

- **Training:** given a *training set* of labeled examples $\{(\mathbf{x}_1,y_1), \ldots, (\mathbf{x}_N,y_N)\}$, estimate the prediction function f by minimizing the prediction error on the training set
- **Testing:** apply f to a never before seen *test example* $\mathbf{x}$ and output the predicted value $y = f(\mathbf{x})$

Slide credit: L. Lazebnik 2/6/19

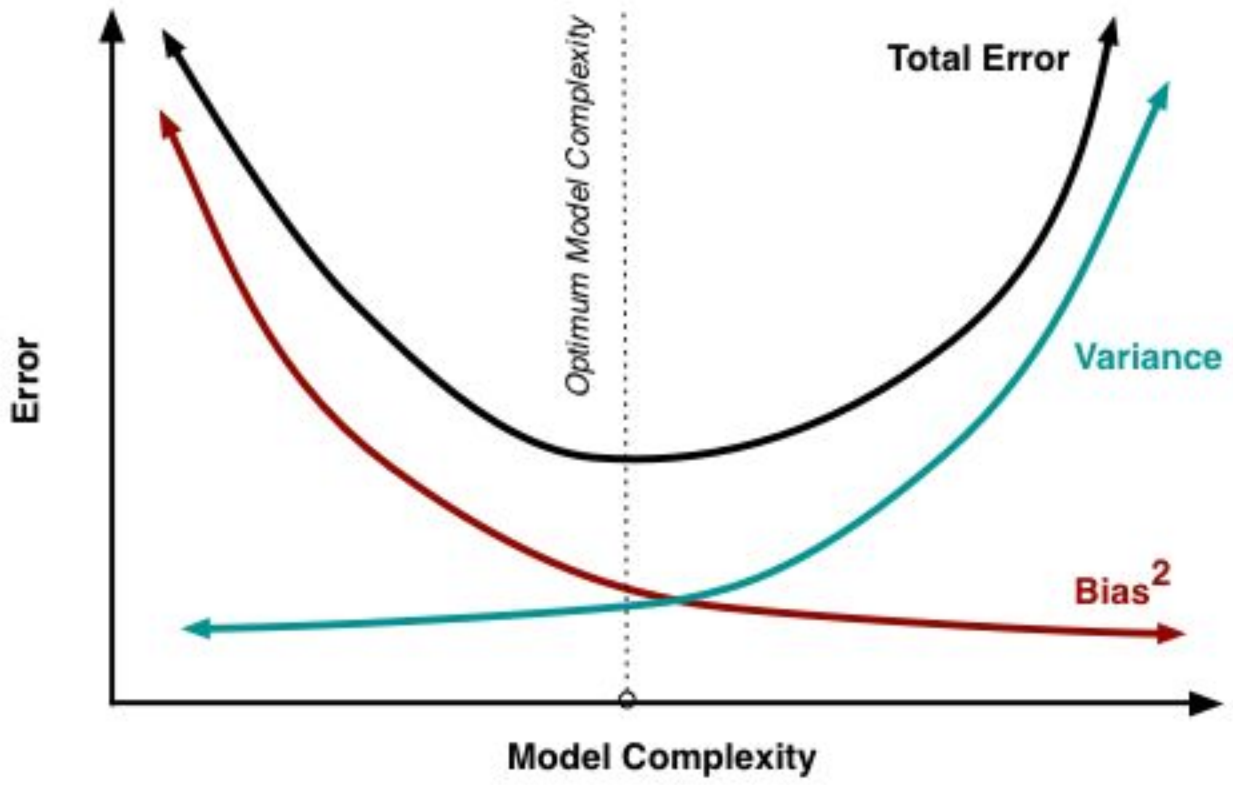
Nearest Neighbor Classifier

- Assign label of nearest training data point to each test data point



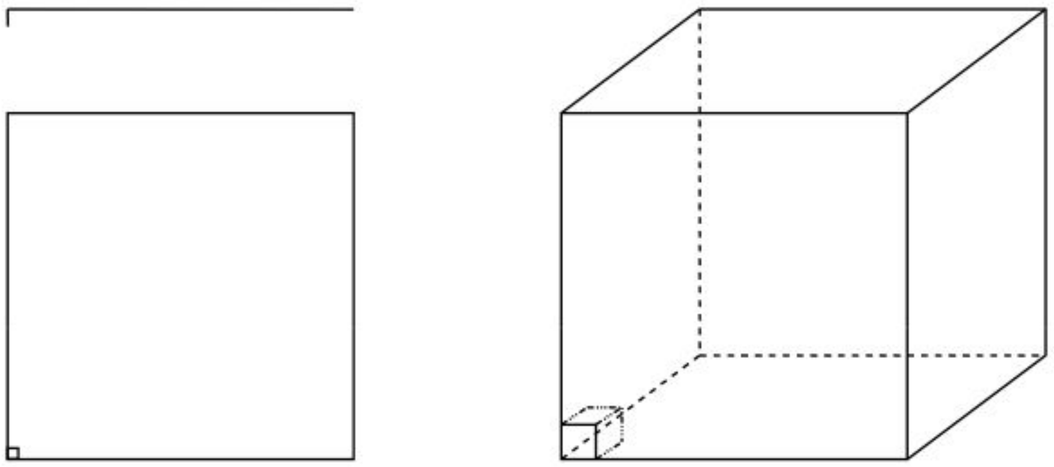
Source: N. Goyal

Bias versus variance trade off

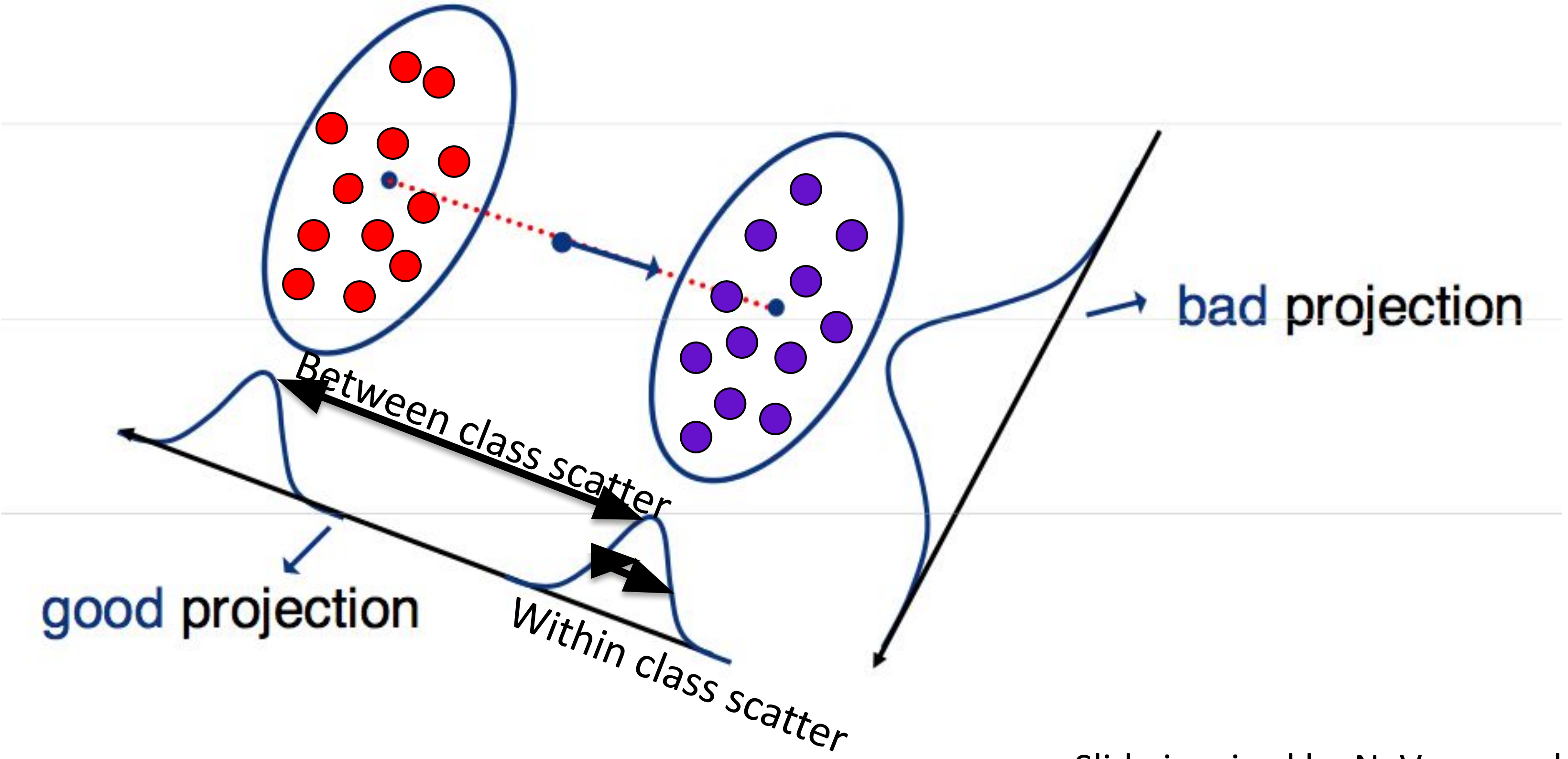


Curse of dimensionality

- Assume 5000 points uniformly distributed in the unit hypercube and we want to apply 5-NN. Suppose our query point is at the origin.
    - In 1-dimension, we must go a distance of 5/5000=0.001 on the average to capture 5 nearest neighbors.
    - In 2 dimensions, we must go $\sqrt{0.001}$ to get a square that contains 0.001 of the volume.
    - In d dimensions, we must go $\left(0.001\right)^{1/d}$



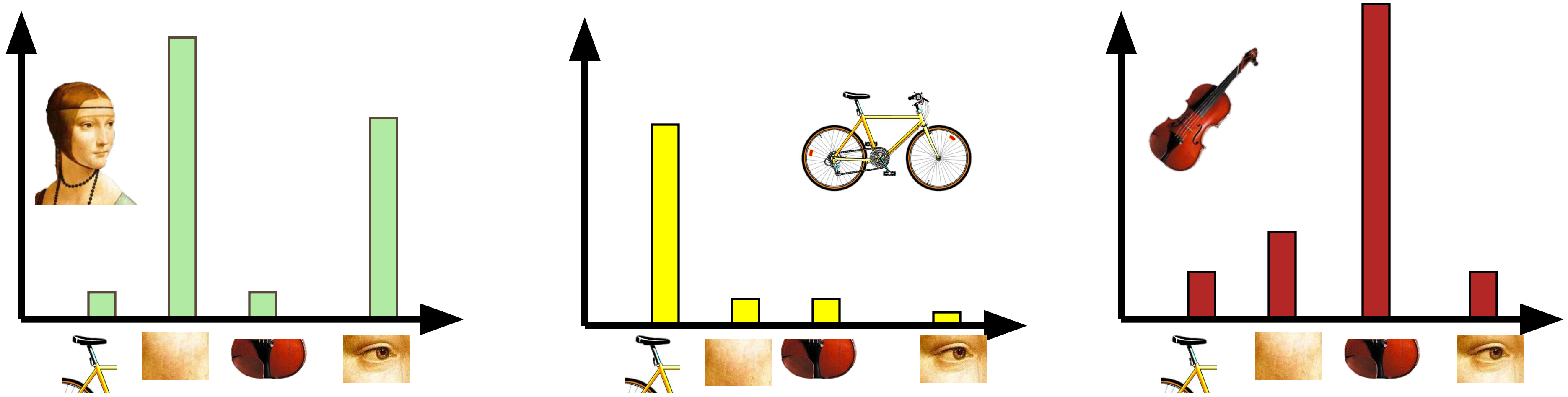
Fischer's Linear Discriminant Analysis



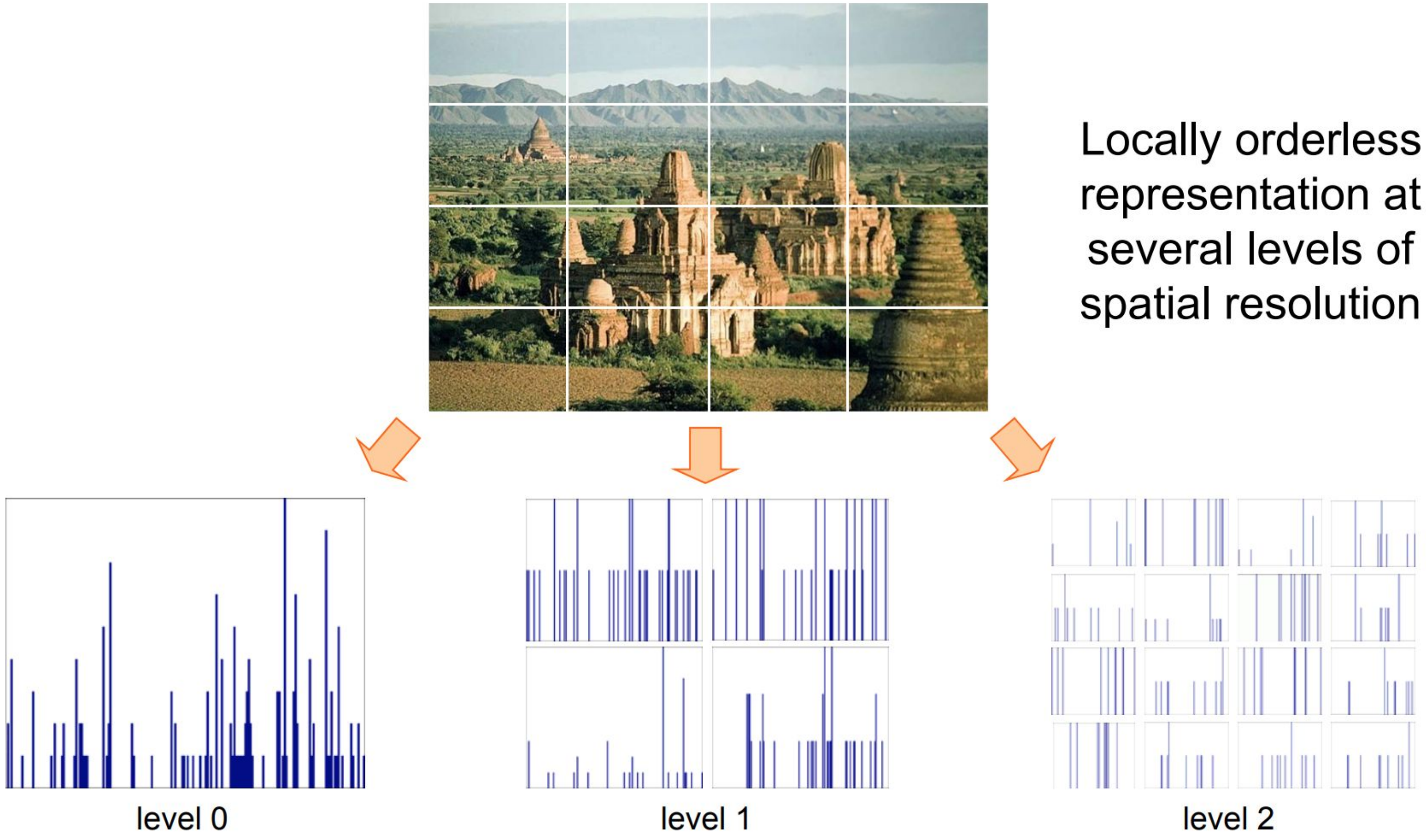
Slide inspired by N. Vasconcelos

Bag of features: outline

1. Extract features
2. Learn "visual vocabulary"
3. Quantize features using visual vocabulary
4. Represent images by frequencies of "visual words"



Bag of words + pyramids

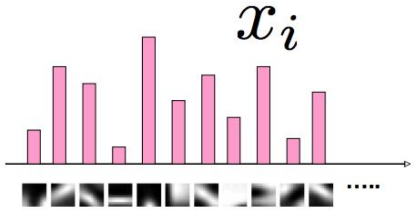


Locally orderless representation at several levels of spatial resolution

Naïve Bayes

- Classify image using histograms of occurrences on visual words:

$$\mathbf{x} =$$



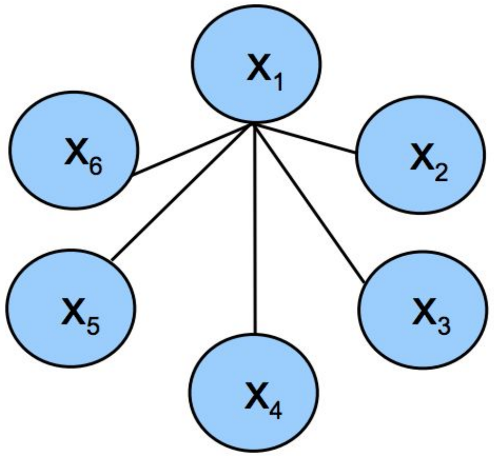
- if only present/absence of a word is taken into account:
- Naïve Bayes classifier assumes that visual words are conditionally independent given object class

$$x_i \in \{0, 1\}$$

Csurka Bray, Dance & Fan, 2004

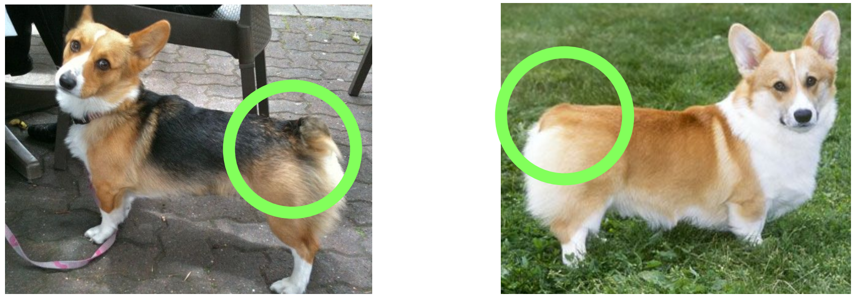
Detecting a person with their parts

- For example, a person can be modelled as having a head, left arm, right arm, etc.
- All parts can be modelled relative to the global person detector



Fine-Grained Recognition

Cardigan Welsh Corgi



Pembroke Welsh Corgi

What breed is this dog?

Key: Find the right features.

Points

2D points: $\mathbf{x} = (x, y) \in \mathcal{R}^2$ or column vector $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

3D points: $\mathbf{x} = (x, y, z) \in \mathcal{R}^3$ (often noted $\mathbf{X}$ or $\mathbf{P}$)

Homogeneous coordinates: append a 1

Why? $\bar{\mathbf{x}} = (x, y, 1)$ $\bar{\mathbf{x}} = (x, y, z, 1)$

Homogeneous coordinates in 2D

2D Projective Space $\mathcal{P}^2 = \mathcal{R}^3 - (0,0,0)$ (same story in 3D with $\mathcal{P}^3$)

- heterogeneous → homogeneous $\begin{bmatrix} x \\ y \end{bmatrix} \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

- homogeneous → heterogeneous $\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow \begin{bmatrix} x/w \\ y/w \end{bmatrix}$

- points differing only by scale are *equivalent*: $(x, y, w) \sim \lambda\,(x, y, w)$

$$\tilde{\mathbf{x}} = (\tilde{x}, \tilde{y}, \tilde{w}) = \tilde{w}(x, y, 1) = \tilde{w}\bar{\mathbf{x}},$$

Camera matrix decomposition

We can decompose the camera matrix like this:

$$\mathbf{P} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \left[\begin{array}{ccc:c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right]$$

(homogeneous) transformation from 2D to 2D, accounting for focal length *f* and origin translation

(homogeneous) perspective projection from 3D to 2D, assuming image plane at z = 1 and shared camera/image origin

Also written as: $\mathbf{P} = \mathbf{K}[\mathbf{I}|\mathbf{0}]$ where $\mathbf{K} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}$

K is called the camera intrinsics

Putting it all together

We can write everything into a single projection: $\mathbf{x}^{\mathbf{I}} \sim \mathbf{K}[\mathbf{I}|\mathbf{0}]\begin{bmatrix} \mathbf{R} & -\mathbf{RC} \\ \mathbf{0} & 1 \end{bmatrix}\mathbf{X}^{\mathbf{W}} = \mathbf{P}\mathbf{X}^{\mathbf{W}}$

The camera matrix now looks like:

$$\mathbf{P} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} [\mathbf{I} \mid \mathbf{0}] \begin{bmatrix} \mathbf{R} & -\mathbf{RC} \\ \mathbf{0} & 1 \end{bmatrix}$$

intrinsic parameters (3 x 3): correspond to camera internals (image-to-image transformation)

perspective projection (3 x 4): maps 3D to 2D points (camera-to-image transformation)

extrinsic parameters (4 x 4): correspond to camera externals (world-to-camera transformation)

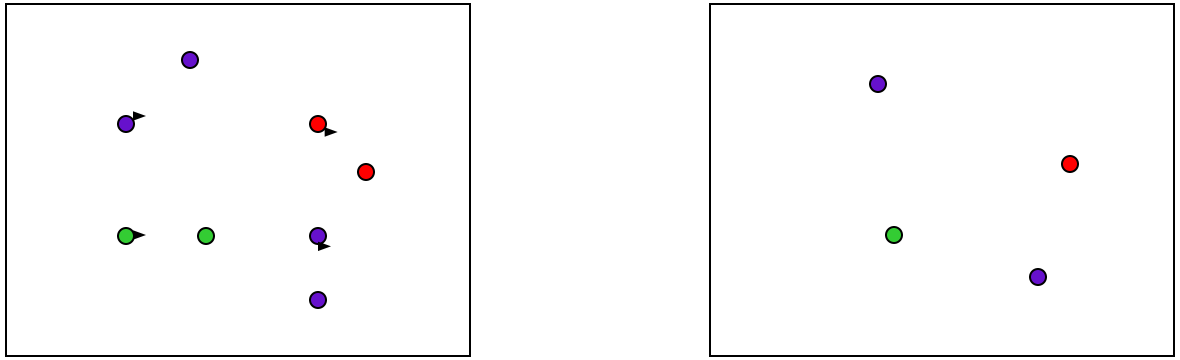
General pinhole camera matrix

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}] \quad \text{where} \quad \mathbf{t} = -\mathbf{RC}$$

$$\mathbf{P} = \underbrace{\begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}}_{\text{intrinsic parameters}} \underbrace{\left[\begin{array}{ccc:c} r_1 & r_2 & r_3 & t_1 \\ r_4 & r_5 & r_6 & t_2 \\ r_7 & r_8 & r_9 & t_3 \end{array}\right]}_{\text{extrinsic parameters}}$$

$$\mathbf{R} = \underbrace{\begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix}}_{\text{3D rotation}} \qquad \mathbf{t} = \underbrace{\begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}}_{\text{3D translation}}$$

Estimating optical flow



$I(x,y,t-1)$ $\qquad$ $I(x,y,t)$

- Given two subsequent frames, estimate the apparent motion field u(x,y), v(x,y) between them

- Key assumptions
  - **Brightness constancy:** projection of the same point looks the same in every frame
  - **Small motion:** points do not move very far
  - **Spatial coherence:** points move like their neighbors

Source: Silvio Savarese

Lucas Kande optical flow

- Optimal (u, v) satisfies Lucas-Kanade equation

$$\underbrace{\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}}_{A^T A} \begin{bmatrix} u \\ v \end{bmatrix} = - \underbrace{\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}}_{A^T b}$$

When is This Solvable?

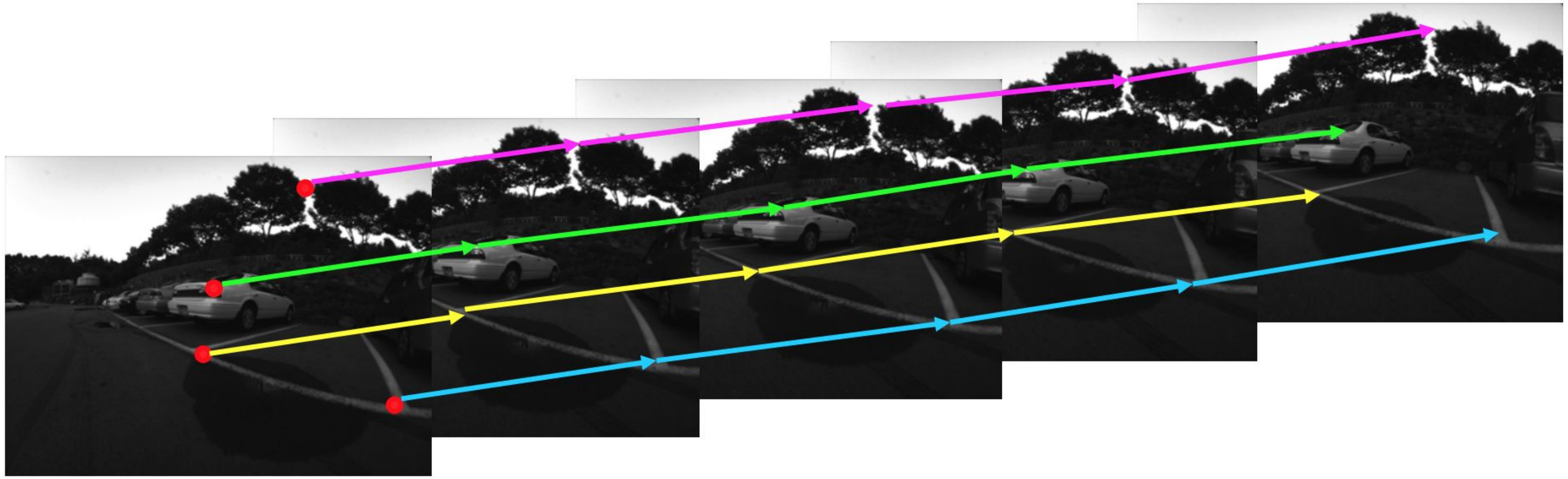
- $\mathbf{A^TA}$ should be invertible
- $\mathbf{A^TA}$ should not be too small due to noise
  - eigenvalues λ_1 and λ_2 of $\mathbf{A^TA}$ should not be too small
- $\mathbf{A^TA}$ should be well-conditioned
  - λ_1 / λ_2 should not be too large (λ_1 = larger eigenvalue)

Does this remind anything to you?

Source: Silvio Savarese

Tracking

Feature point tracking



Slide credit: Yonsei Univ.

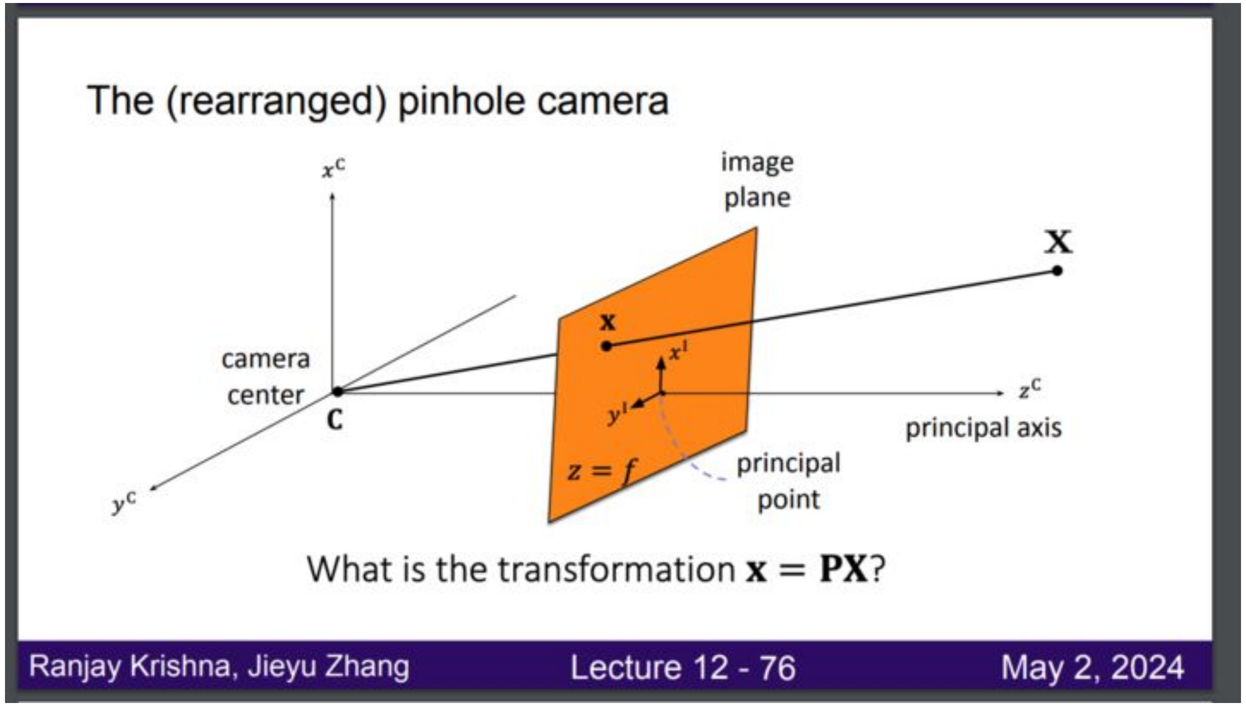
What should you take away from this class?

- A broad understanding of computer vision as a field.
- Learning to use common software packages: jupyter, numpy, scipy.
- Converting ideas into mathematical equations.
- Converting mathematical equations into code.
- Learning to communicate ideas and algorithms in formal writing.

Exam

- Pay attention to formulation and definition

  eg, what is <u>principal point</u> and <u>principal axis</u>?



Exam

- Pay attention to calculation

say what's eigenvectors and eigenvalues of the matrix A

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}.$$

PCA by SVD

$$C = \frac{1}{n} U \Sigma^2 U^T$$

- Note that U is (d x d) and orthonormal, and Σ^2 is diagonal.
  This is just the eigenvalue decomposition of C
- This means that we can calculate the eigenvectors of C using the eigenvectors of X_c
- It follows that
  - The eigenvectors of C are the columns of U
  - The eigenvalues of C are the diagonal entries of Σ^2: λ_i^2

Exam

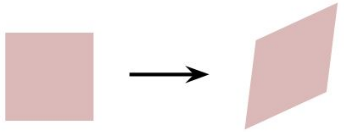
- Pay attention to property and property comparison between methods
- Pay attention to why and when we use or not use a method
- Pay attention to assumptions, when they hold or not hold

Affine transformation = similarity + no restrictions on scaling

Properties of affine transformations:

- arbitrary 6 Degrees Of Freedom
- lines map to lines
- parallel lines map to parallel lines
- ratios are preserved

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

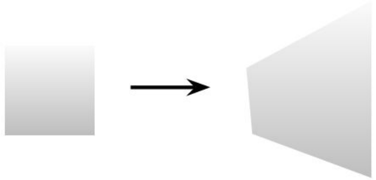


Projective transformation (homography)

Properties of projective transformations:

- 8 degrees of freedom
- lines map to lines
- parallel lines do not necessarily map to parallel lines
- ratios are not necessarily preserved

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



Good Luck