Lecture 20:
Final Review

Juan Carlos Niebles and Ranjay Krishna

Stanford Vision and Learning Lab

Exam details

- Monday, December 11
- 12:15 to 3:15pm
- Location: 320-105
- Practice exam available online.

- Make-up exam for those with conflicts
  - You should have received an email about this.

Class overview

- 68% assignments
- 5% class notes
- 7% extra credit
- 20% final exam

- Don’t be worried about the final.
- Optimal strategy:
  - Go over all the materials we covered in class.
  - Read through the entire final first and work on the problems you find easy first.

Exam overview

- 100 points in total
- 15 points for multiple choice
- 20 points for true false
- 25 points for filters, edges, corners, RANSAC, Hough transform
- 15 points for segmentation and seam carving
- 15 points for recognition and detection
- 10 points for optical flow and tracking

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

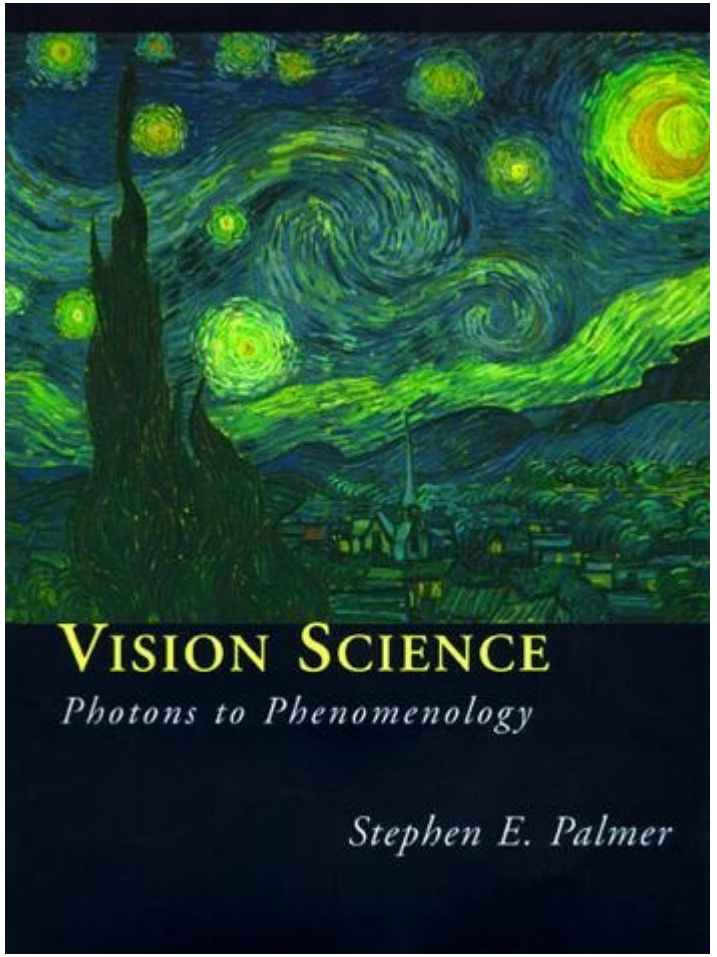
What a computer sees

Source: S. Narasimhan

What is color?

- The result of interaction between physical light in the environment and our visual system.

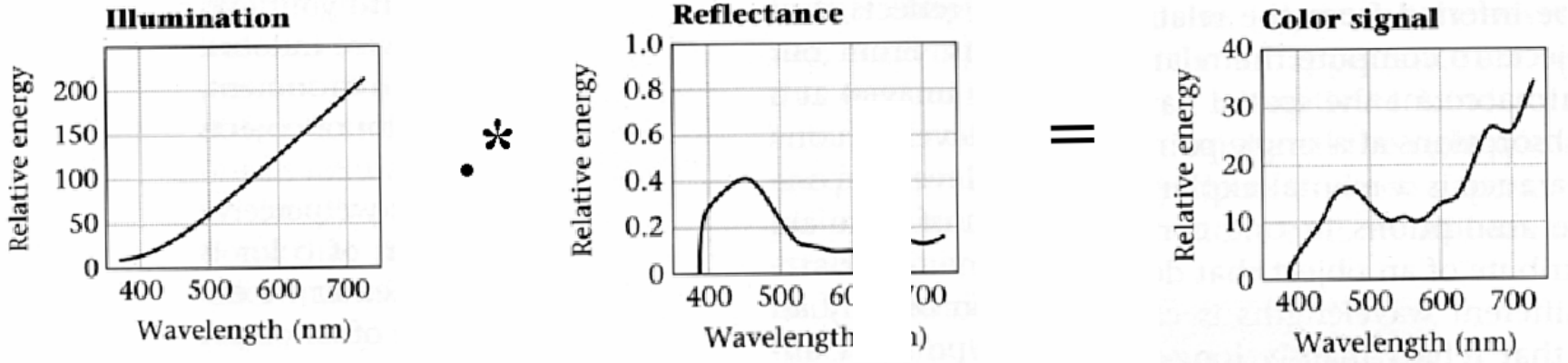
- A *psychological property* of our visual experiences when we look at objects and lights, *not a physical property* of those objects or lights.



Slide credit: Lana Lazebnik

Interaction of light and surfaces

- Reflected color is the result of interaction of light source spectrum with surface reflectance
- Spectral radiometry
  - All definitions and units are now “per unit wavelength”
  - All terms are now “spectral”



From Foundation of Vision by Brian Wandell, Sinauer Associates, 1995

White balance

- Von Kries adaptation
  - Multiply each channel by a gain factor
  - A more general transformation would correspond to an arbitrary 3x3 matrix
- Best way: gray card
  - Take a picture of a neutral object (white or gray)
  - Deduce the weight of each channel
    - If the object is recoded as r_w, g_w, b_w use weights $1/r_w$, $1/g_w$, $1/b_w$

White balance

- Without gray cards: we need to “guess” which pixels correspond to white objects
- Gray world assumption
  - The image average r_{ave}, g_{ave}, b_{ave} is gray
  - Use weights $1/r_{ave}$, $1/g_{ave}$, $1/b_{ave}$
- Brightest pixel assumption (non-staurated)
  - Highlights usually have the color of the light source
  - Use weights inversely proportional to the values of the brightest pixels
- Gamut mapping
  - Gamut: convex hull of all pixel colors in an image
  - Find the transformation that matches the gamut of the image to the gamut of a “typical” image under white light
- Use image statistics, learning techniques

Norms

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|$$

$$\|x\|_\infty = \max_i |x_i|$$

ℓ_p

$$\|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}.$$

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

$$\|A\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} A_{ij}^2} = \sqrt{\mathrm{tr}(A^T A)}.$$

Linear algebra

- Singular value decompositon
- Eigenvalues and eigenvectors
- Spectral theory
- Symmetric and singular matrices
- Matrix gradient
- The Hessian
- Matrix rank

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

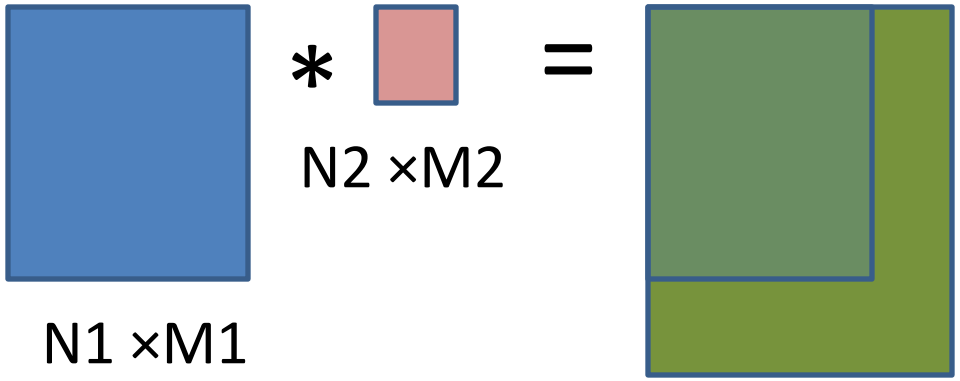
$$\text{for } n < n_0, m < m_0, \text{ if } f[n,m] = 0 \implies g[n,m] = 0$$

  - Shift invariance:

$$f[n - n_0, m - m_0] \xrightarrow{\mathcal{S}} g[n - n_0, m - m_0]$$

Image support and edge effect

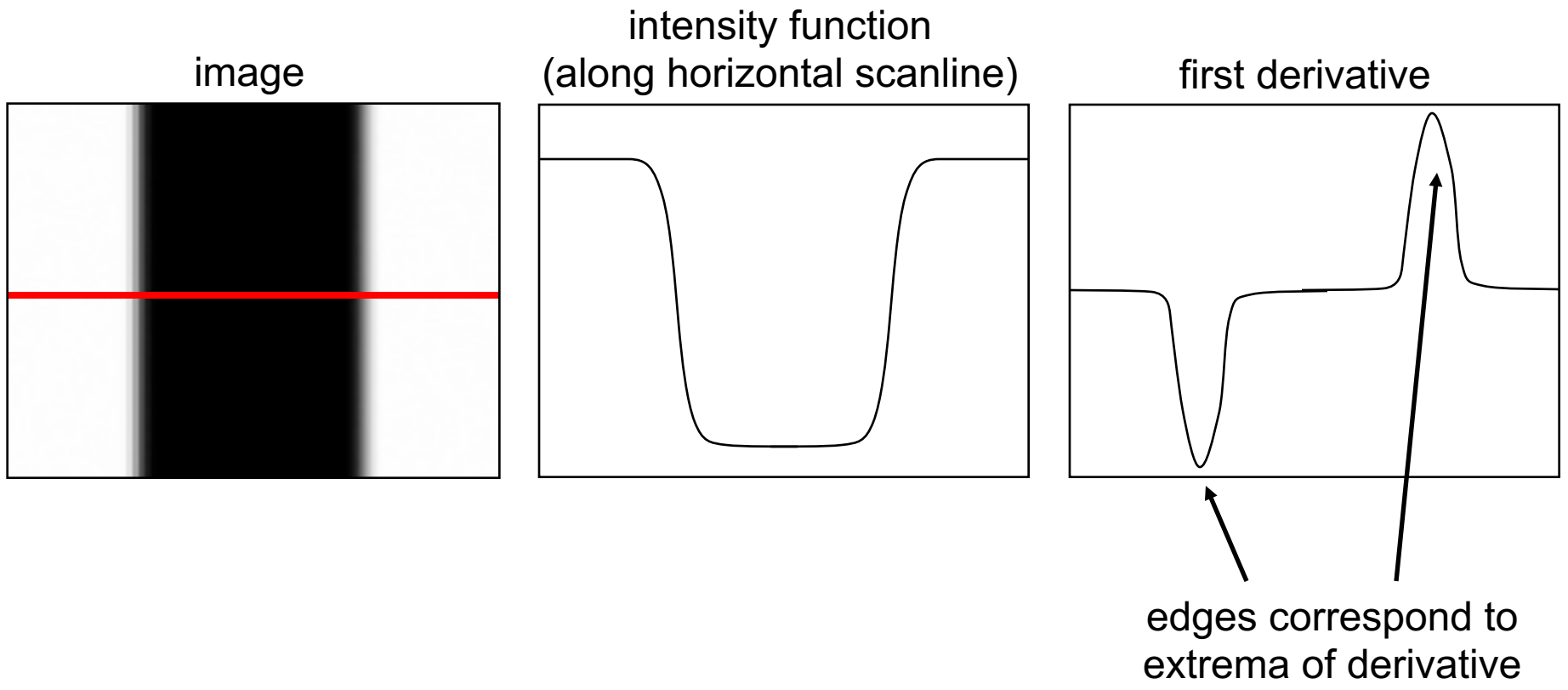
- A computer will only convolve **finite support signals.**
  - That is: images that are zero for n,m outside some rectangular region
- numpy's convolution performs 2D DS convolution of finite-support signals.



(N1 + N2 – 1) × (M1 +M2 – 1)

Characterizing edges

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



Discrete derivate in 2D

Given function $f(x, y)$

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

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

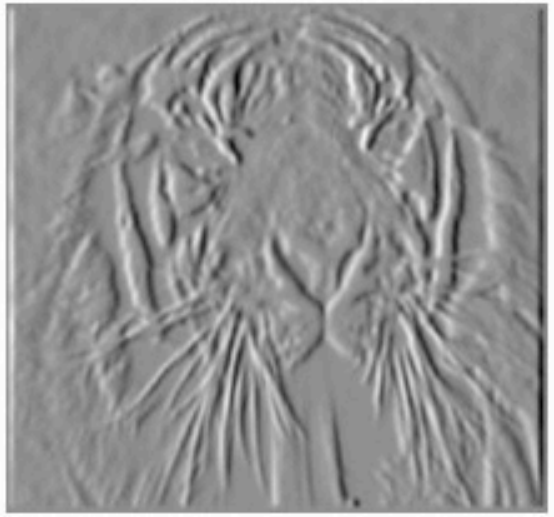
Gradient direction $$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

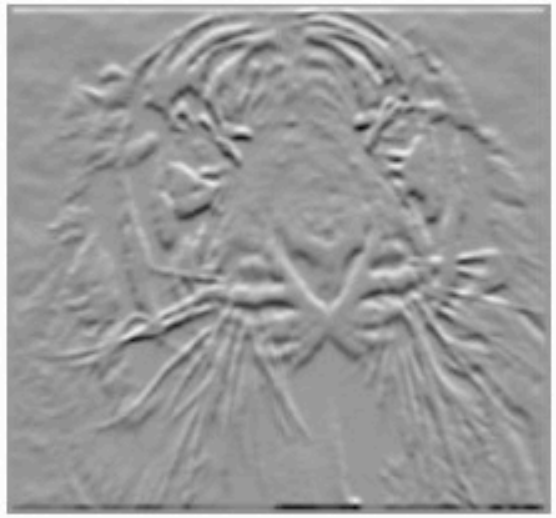
Finite differences: example

Original Image

Gradient magnitude

x-direction



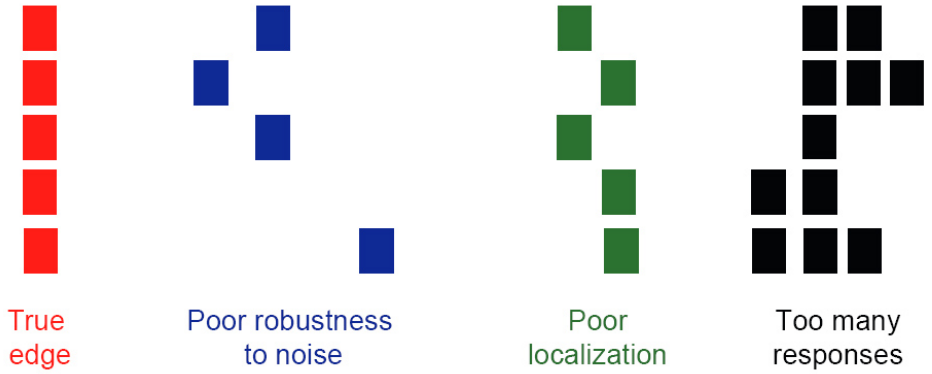


y-direction

- 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



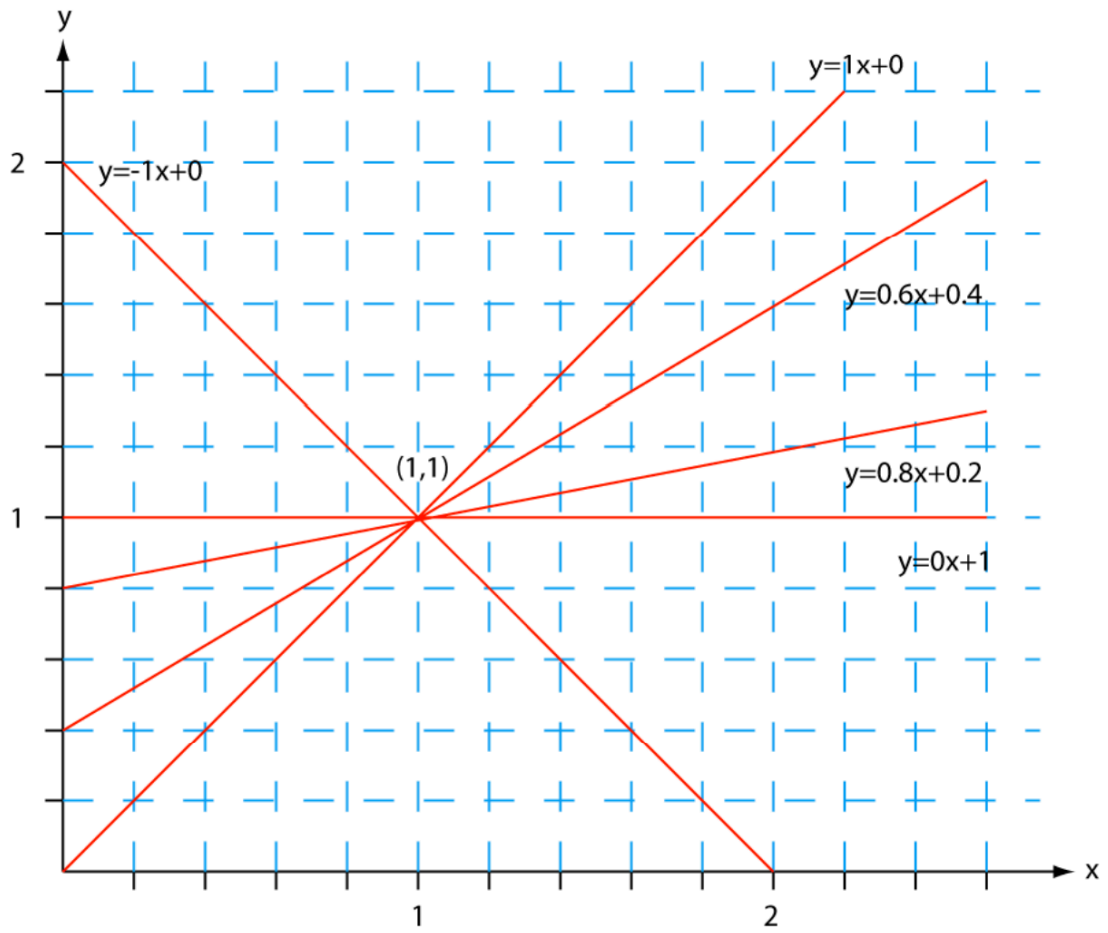
Canny edge detector

- 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

- Two points (x_1, y_1) and$(x_2\ y_2)$ define a line in the (x, y) plane.
- These two points give rise to two different lines in (a,b) space.
- In (a,b) space these lines will intersect in a point (a' b')
- All points on the line defined by (x_1, y_1) and $(x_2\ , y_2)$ in (x, y) space will parameterize lines that intersect in (a', b') in (a,b) space.

Detecting lines using Hough transform



Algorithm 15.4: RANSAC: fitting lines using random sample consensus

Determine:
    n — the smallest number of points required
    k — the number of iterations required
    t — the threshold used to identify a point that fits well
    d — the number of nearby points required
      to assert a model fits well
Until k iterations have occurred
    Draw a sample of n points from the data
      uniformly and at random
    Fit to that set of n points
    For each data point outside the sample
      Test the distance from the point to the line
        against t; if the distance from the point to the line
        is less than t, the point is close
    end
    If there are d or more points close to the line
      then there is a good fit. Refit the line using all
      these points.
end
Use the best fit from this collection, using the
  fitting error as a criterion

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

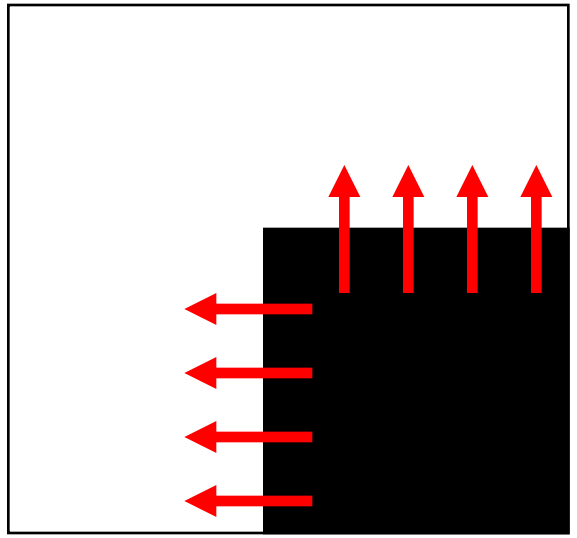
- Region extraction needs to be repeatable and accurate
  - Invariant to translation, rotation, scale changes
  - Robust or covariant to out-of-plane (≈affine) transformations
  - Robust to lighting variations, noise, blur, quantization
- Locality: Features are local, therefore robust to occlusion and 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:

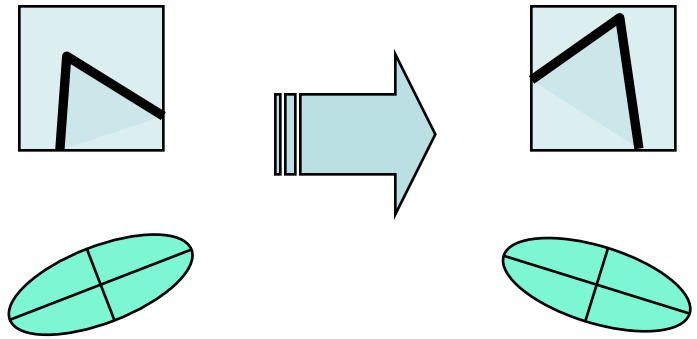
$$M = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

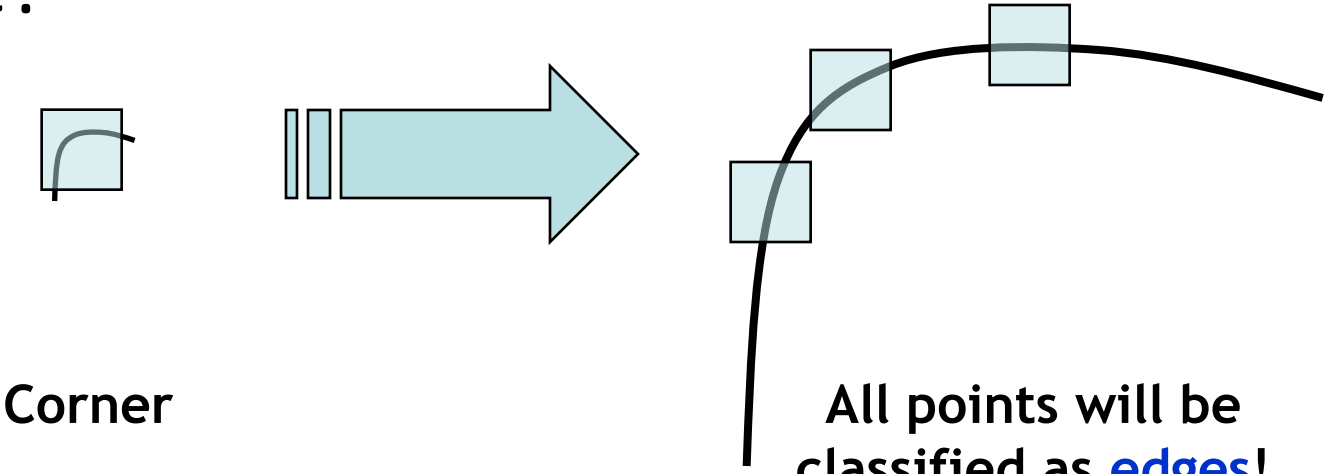


Slide credit: David Jacobs

Harris Detector: Properties

- Translation invariance
- Rotation invariance
- Scale invariance?





Not invariant to image scale!

Slide credit: Kristen Grauman

Scale Invariant Detection

- Functions for determining scale $f = \text{Kernel} * \text{Image}$

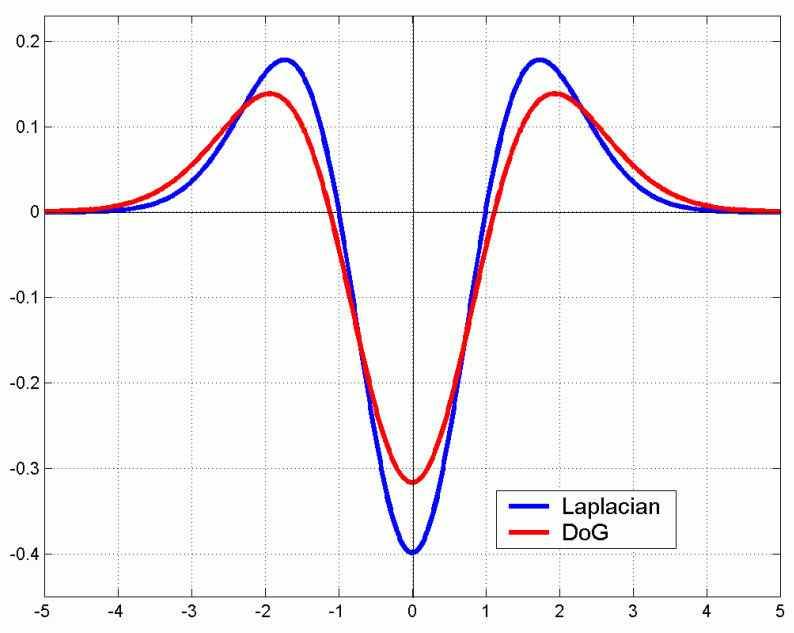
Kernels:

$$L = \sigma^2 \left(G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma) \right)$$

(Laplacian)

$$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$$

(Difference of Gaussians)



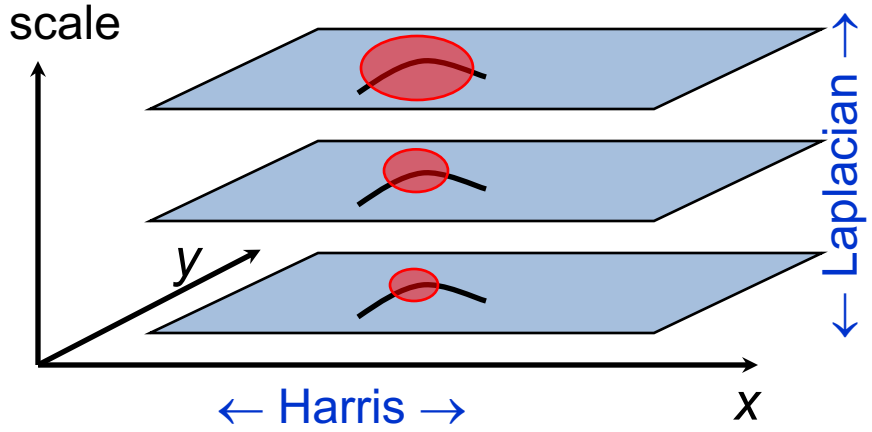
where Gaussian

$$G(x, y, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2 + y^2}{2\sigma^2}}$$

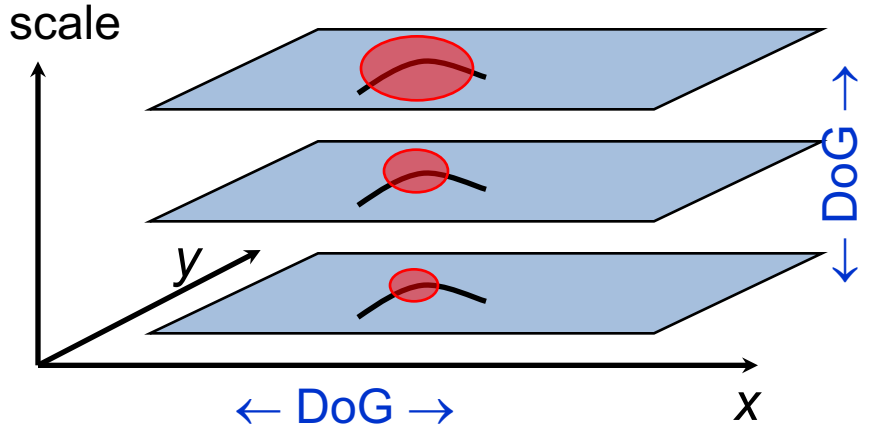
Note: both kernels are invariant to *scale* and *rotation*

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



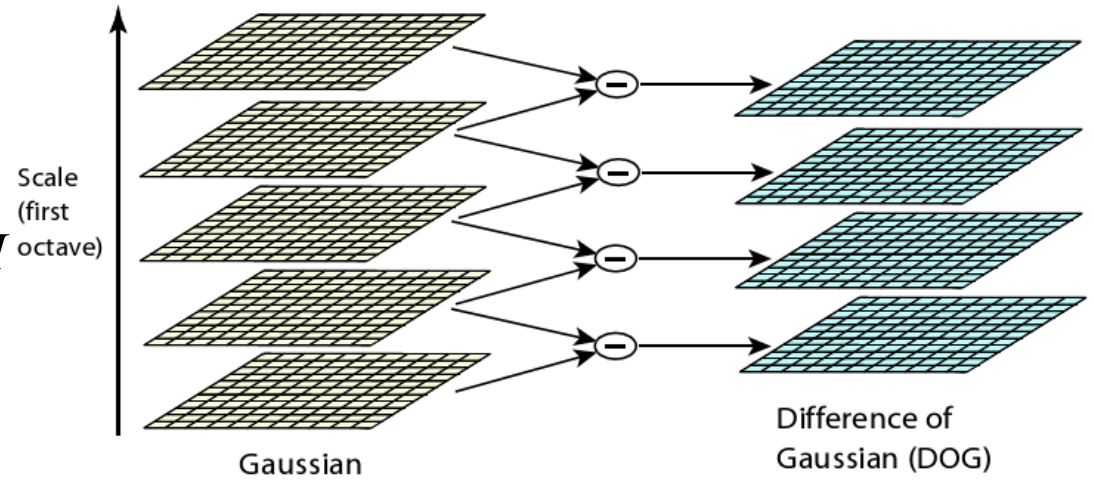
[1] K.Mikolajczyk, C.Schmid. "Indexing Based on Scale Invariant Interest Points". ICCV 2001
[2] D.Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". IJCV 2004

Difference-of-Gaussians

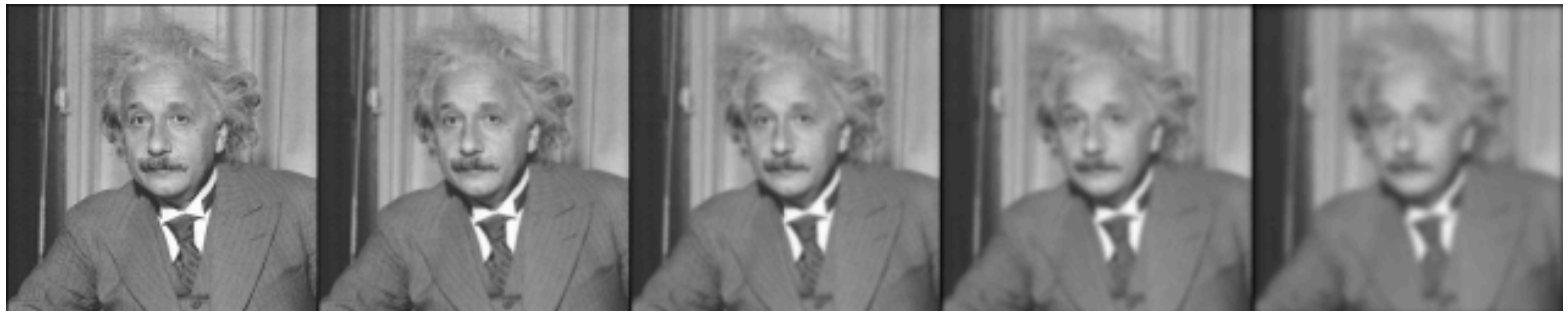
$G(k^2\sigma) * I$

$G(k\sigma) * I$

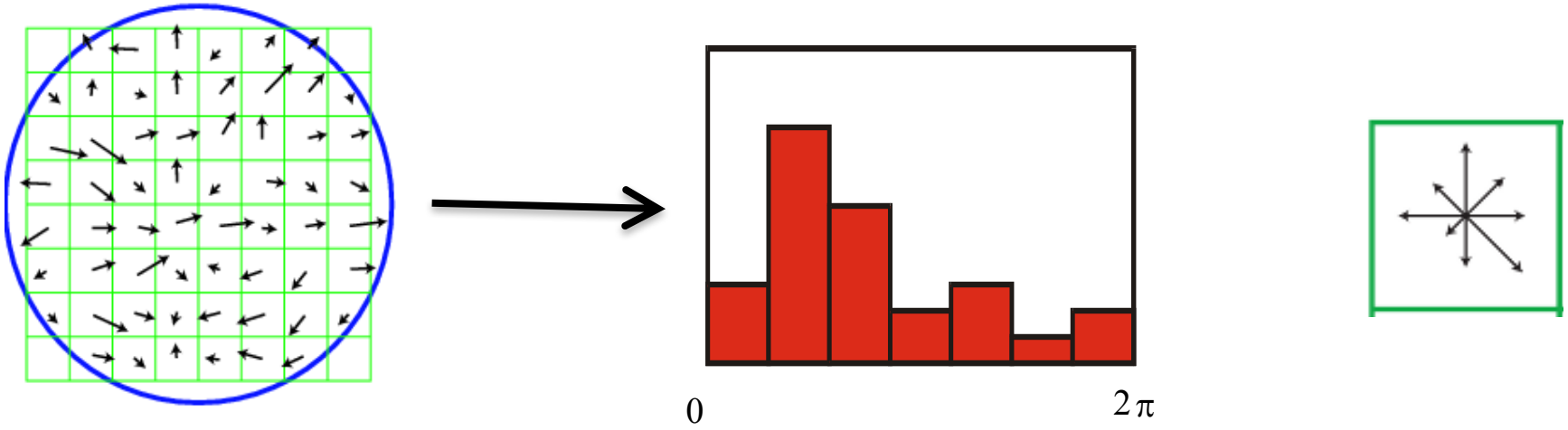
$G(\sigma) * I$



$D(\sigma) \equiv (G(k\sigma) - G(\sigma)) * I$



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.

SIFT descriptor formation

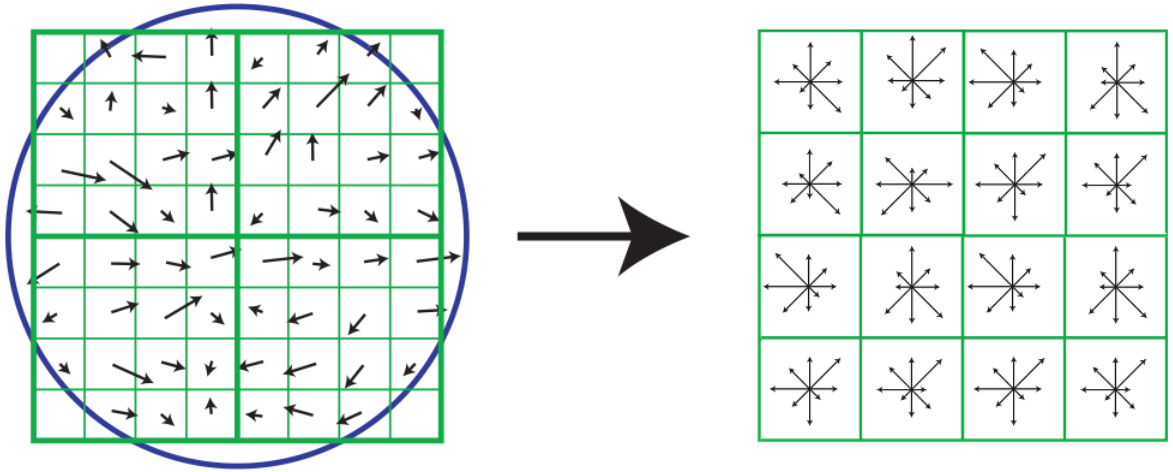
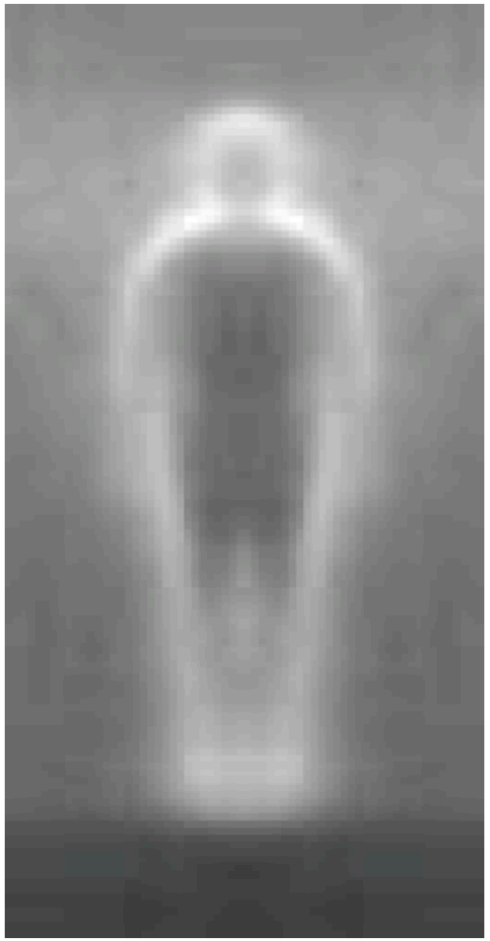


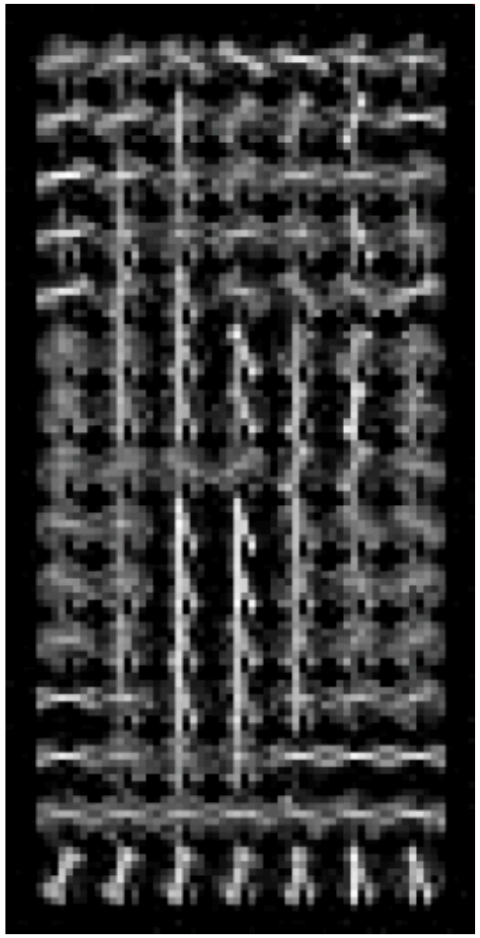
Image gradients

Keypoint descriptor

- 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, and a 4x4 histogram array.

Histogram of Oriented Gradients

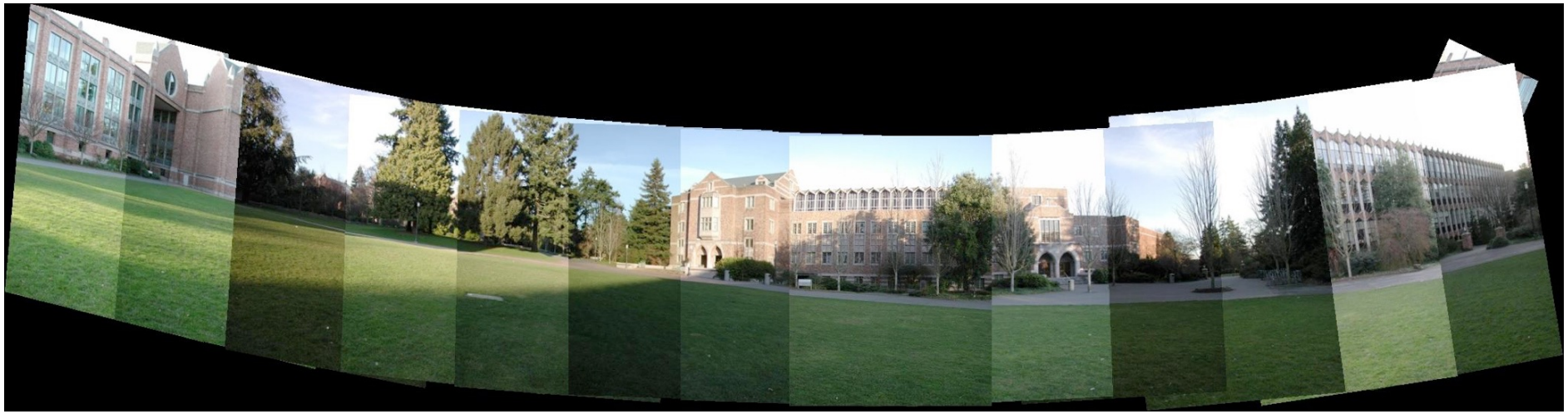




Difference between HoG and SIFT

- 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.

Panorama



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

Preserved Energy

Average Pixel Energy ↑

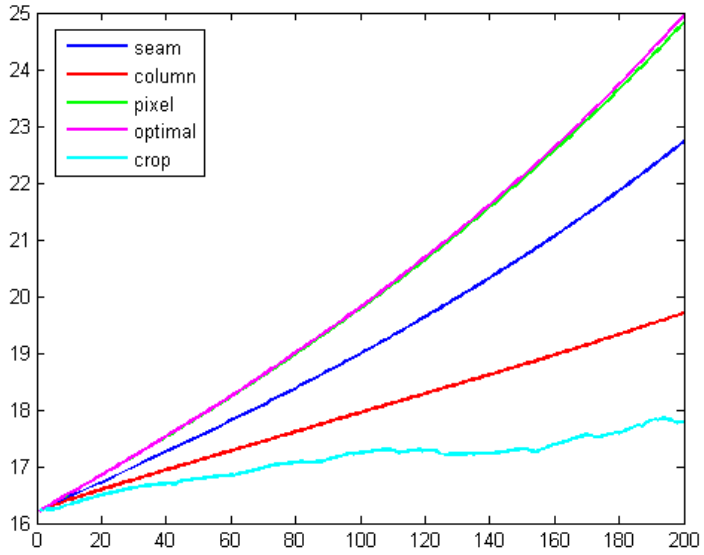


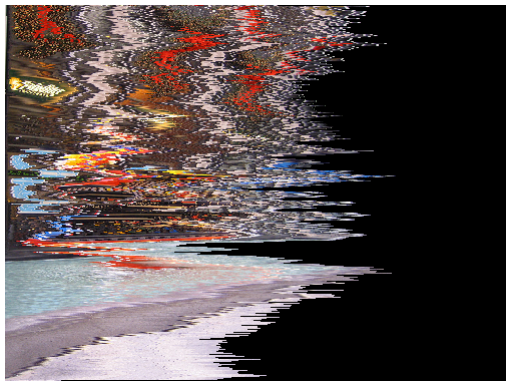
Image Reduction →

crop

column

seam

pixel



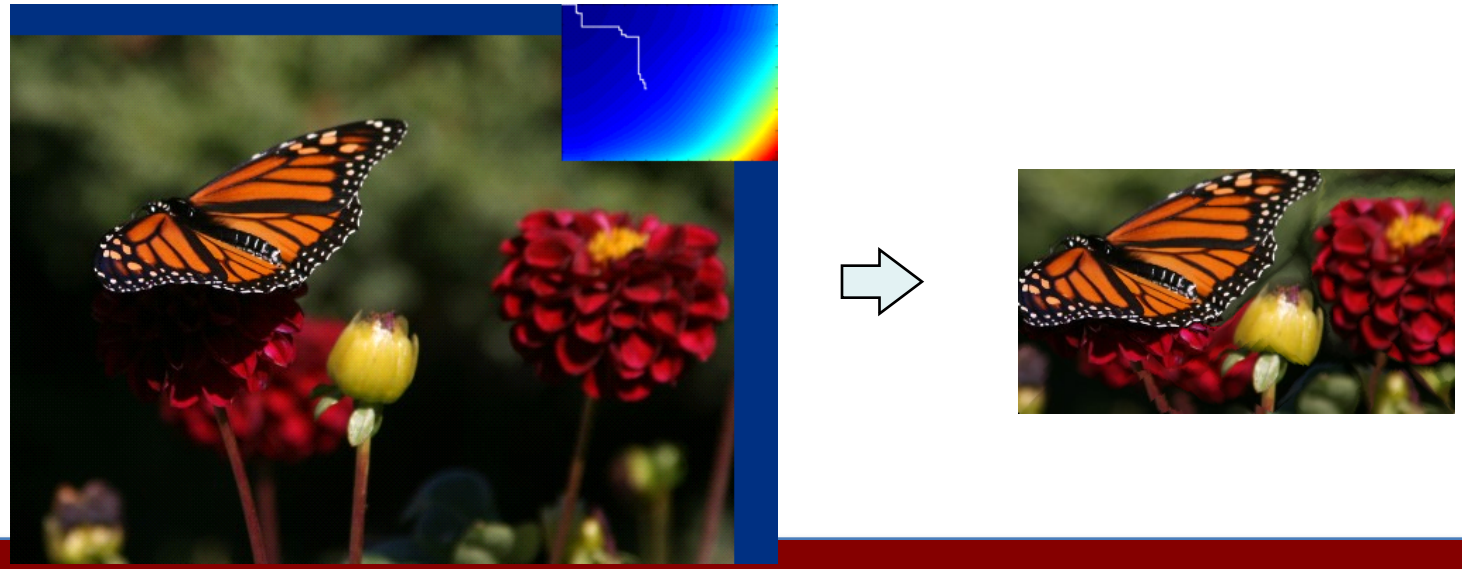
optimal

Retargeting in Both Dimensions

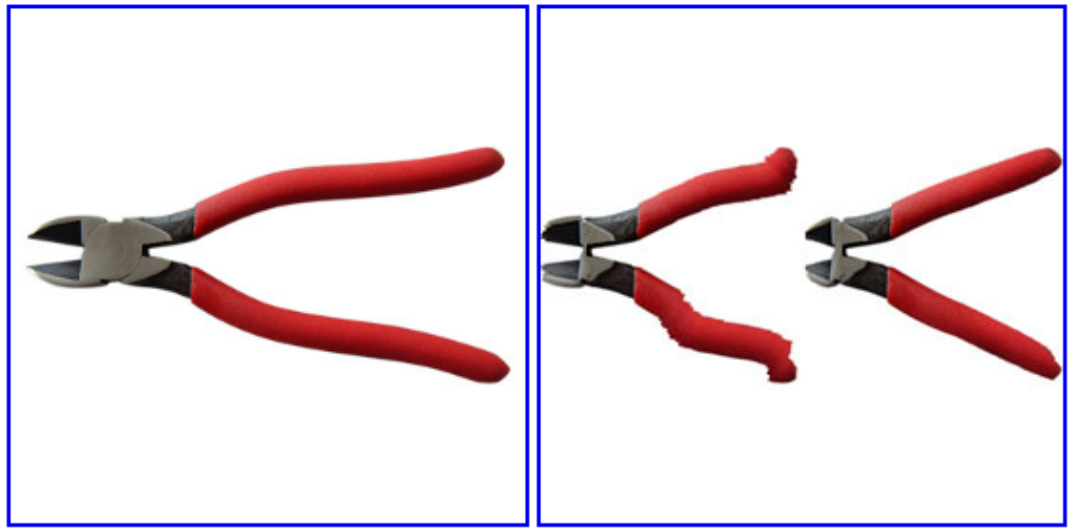
- The recursion relation:

$$\mathbf{T}(r,c) = min(\mathbf{T}(r-1,c) + E(\mathbf{s^x}(\mathbf{I_{n-r-1\times m-c}})), \mathbf{T}(r,c-1) + E(\mathbf{s^y}(\mathbf{I_{n-r\times m-c-1}})))$$

$$\min_{\mathbf{s^x},\mathbf{s^y},\alpha} \sum_{i=1}^{\kappa} E(\alpha_i \mathbf{s_i^x} + (1-\alpha_i)\mathbf{s_i^y})$$



Backward versus forward energy

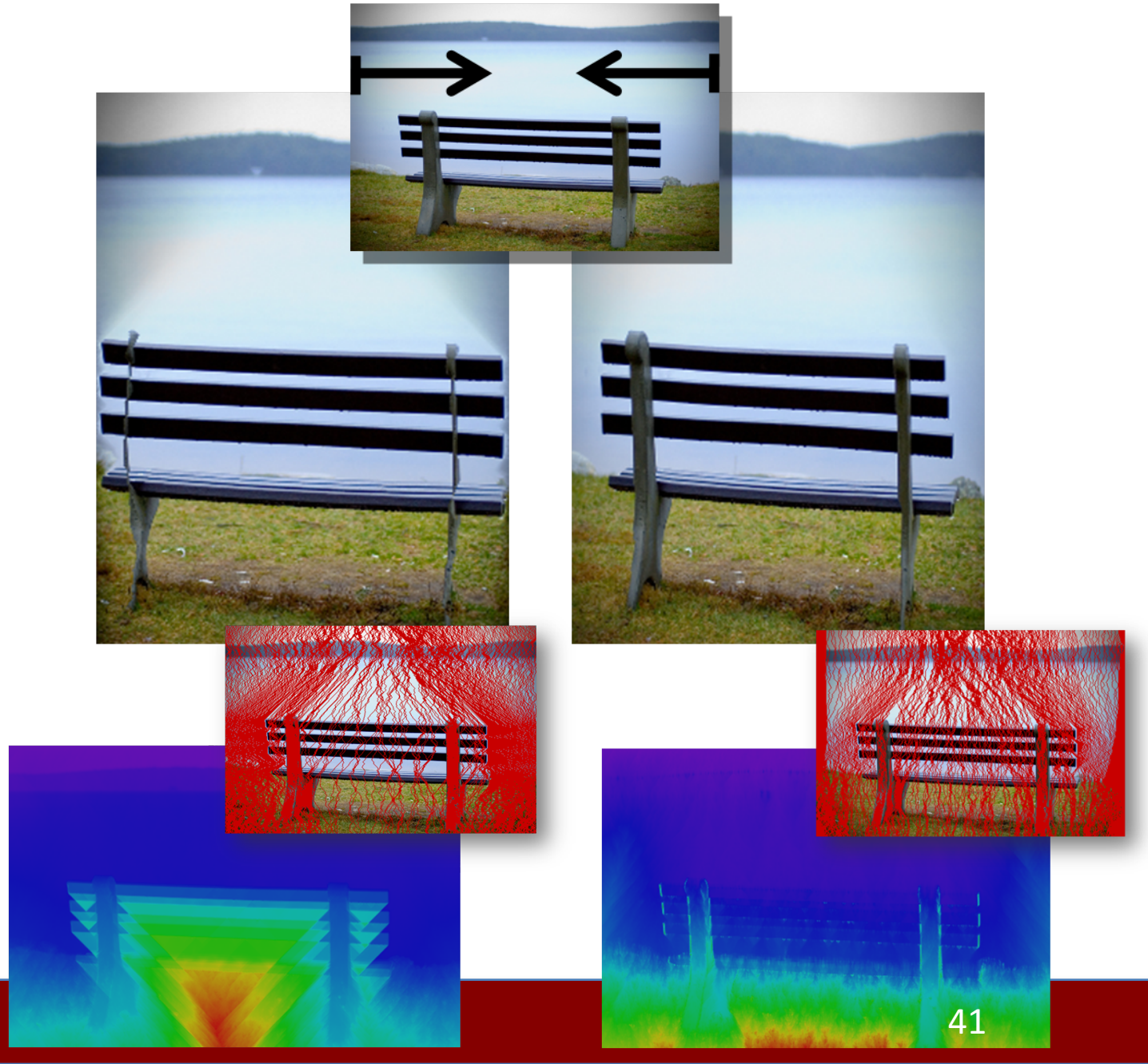


Input Backward Forward

Input

Backward

Forward

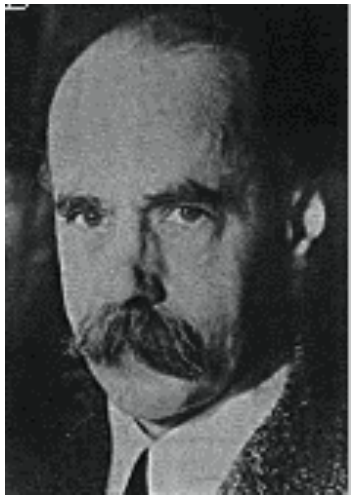


Gestalt Theory

- Gestalt: whole or group
  - Whole is greater than sum of its parts
  - Relationships among parts can yield new properties/features
- Psychologists identified series of factors that predispose set of elements to be grouped (by human visual system)

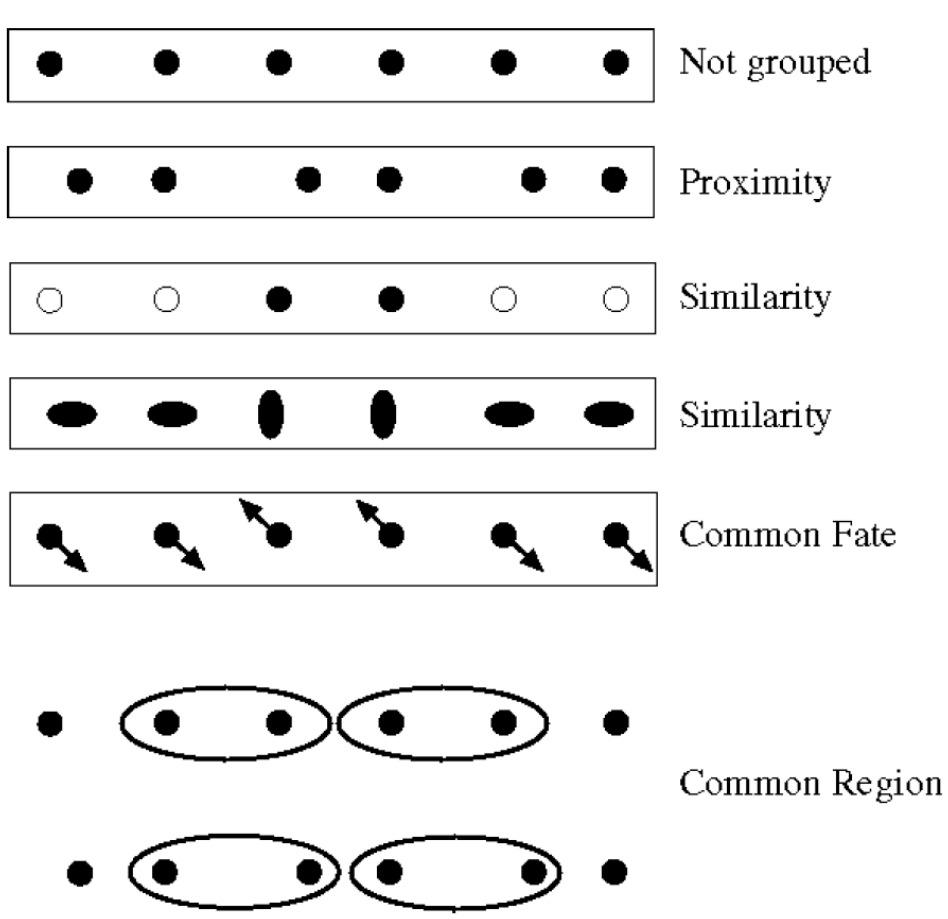
"I stand at the window and see a house, trees, sky. Theoretically I might say there were 327 brightnesses and nuances of colour. Do I have "327"? No. I have sky, house, and trees."

Max Wertheimer
(1880-1943)



Untersuchungen zur Lehre von der Gestalt,
***Psychologische Forschung*, Vol. 4, pp. 301-350, 1923**
http://psy.ed.asu.edu/~classics/Wertheimer/Forms/forms.htm

Gestalt Factors



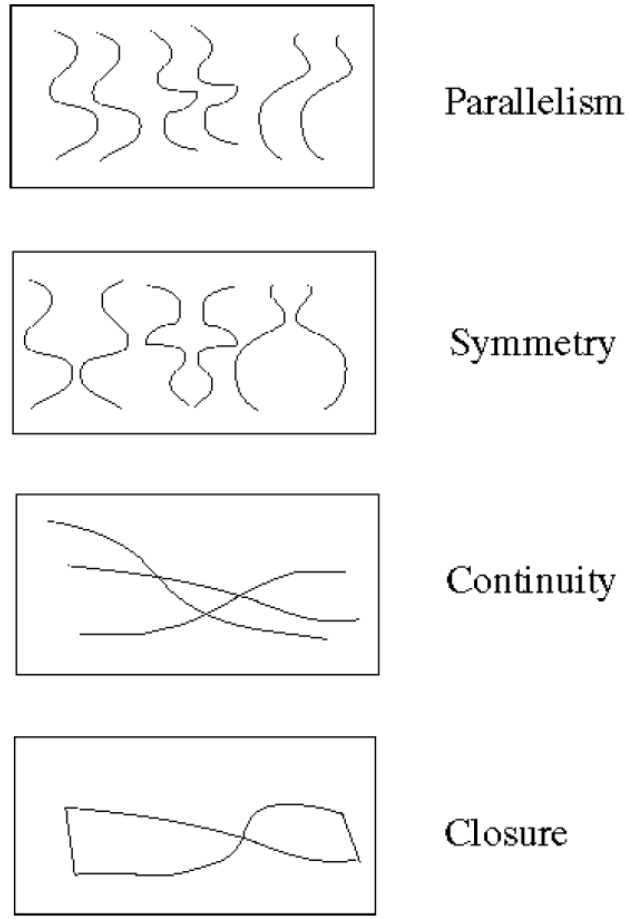


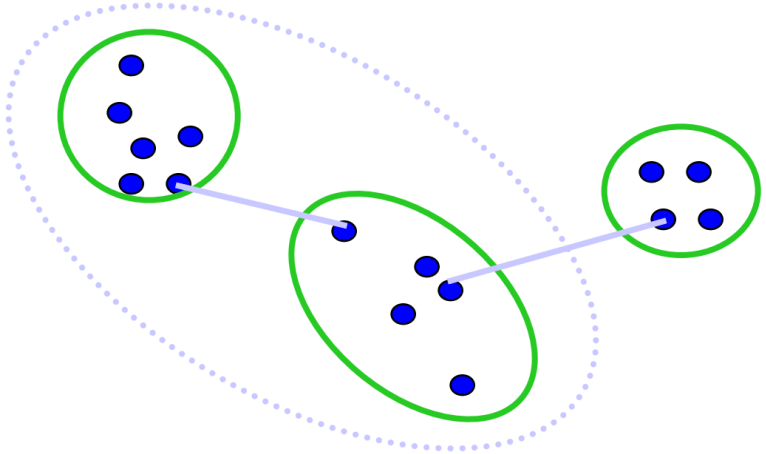
Image source: Forsyth & Ponce

- These factors make intuitive sense, but are very difficult to translate into algorithms.

Different measures of nearest clusters

Single Link

- $d(C_i, C_j) = \min_{x \in C_i, x' \in C_j} d(x, x')$. This is known as *single-linkage*. It is equivalent to the minimum spanning tree algorithm. One can set a threshold and stop clustering once the distance between clusters is above the threshold. Single-linkage tends to produce long and skinny clusters.

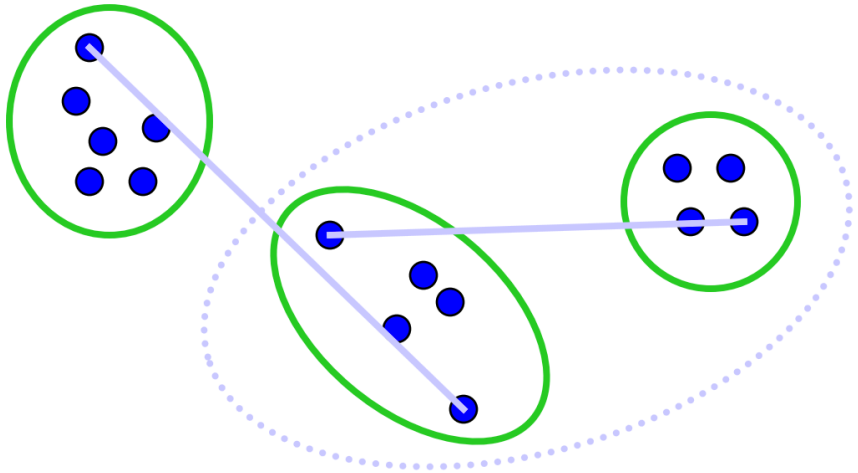


Long, skinny clusters

Different measures of nearest clusters

Complete Link

- $d(C_i, C_j) = \max_{x \in C_i, x' \in C_j} d(x, x')$. This is known as *complete-linkage*. Clusters tend to be compact and roughly equal in diameter.



Tight clusters

Conclusions: Agglomerative Clustering

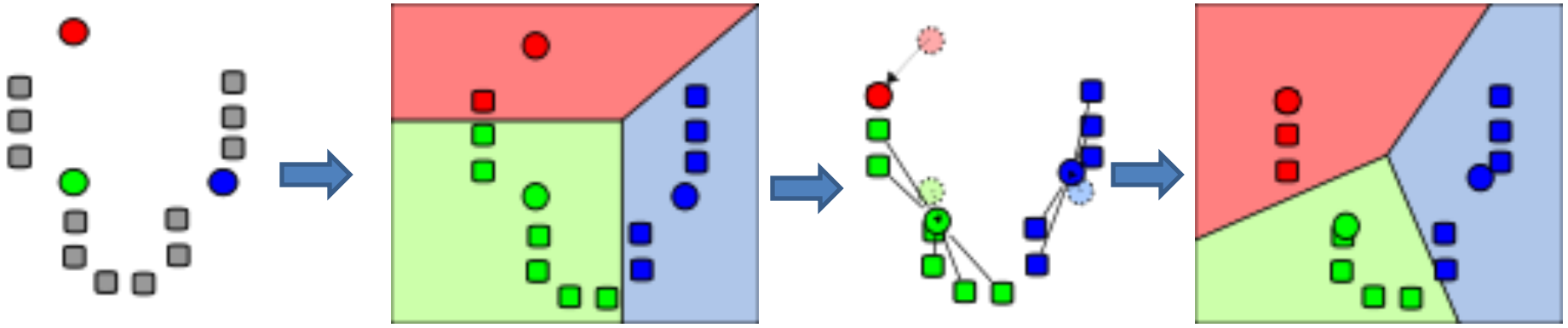
Good

- 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)

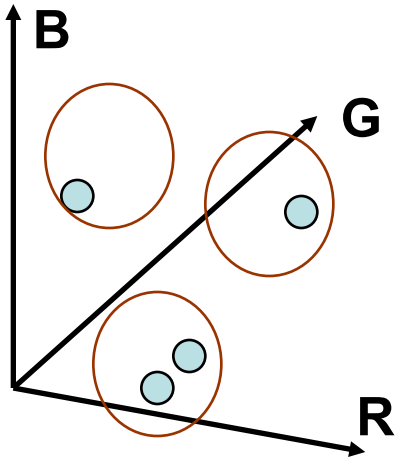
- Java demo:

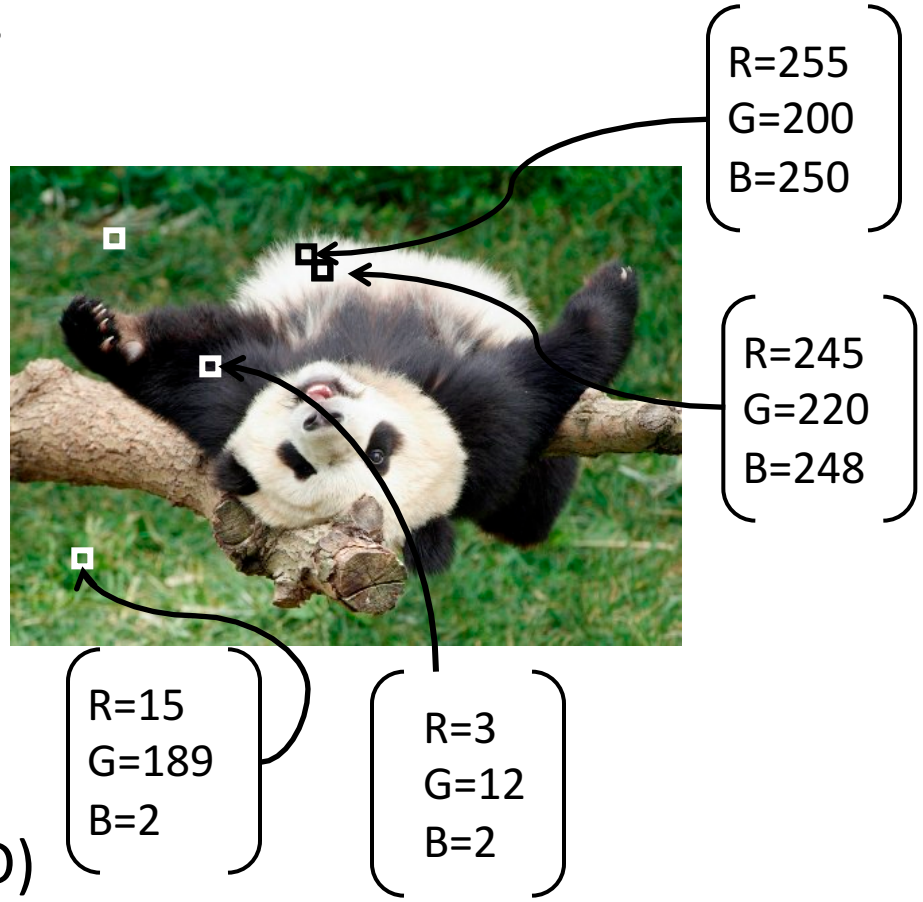
http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/AppletKM.html

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



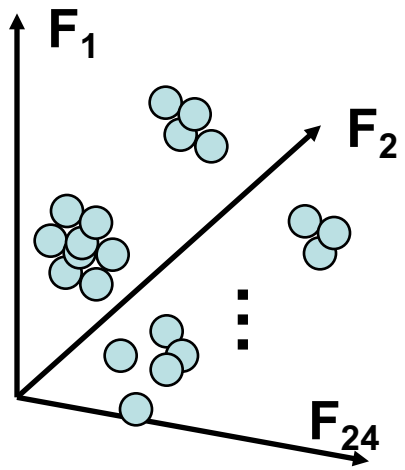


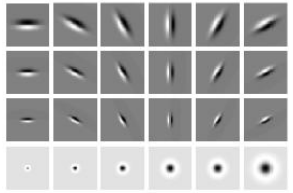
- Feature space: color value (3D)

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

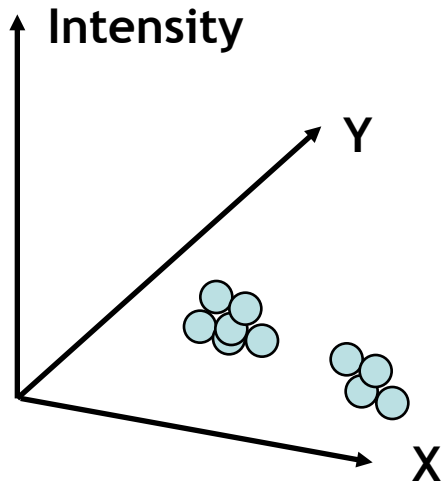
- Feature space: filter bank responses (e.g., 24D)

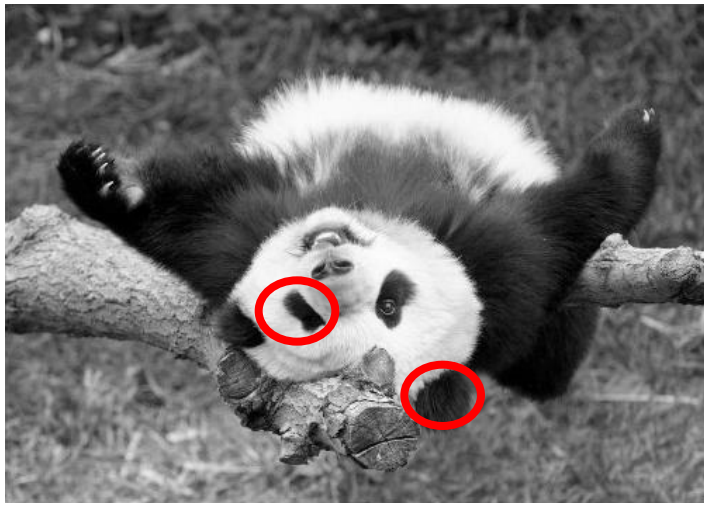
Slide credit: Kristen Grauman

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



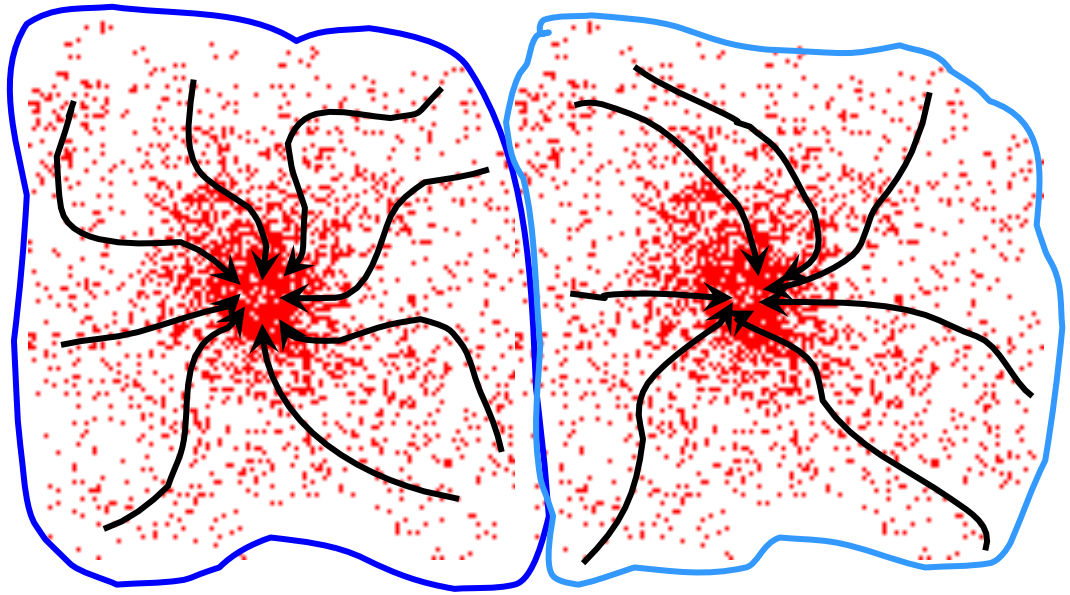


$\Rightarrow$ Way to encode both *similarity* and *proximity.*

Slide credit: Kristen Grauman

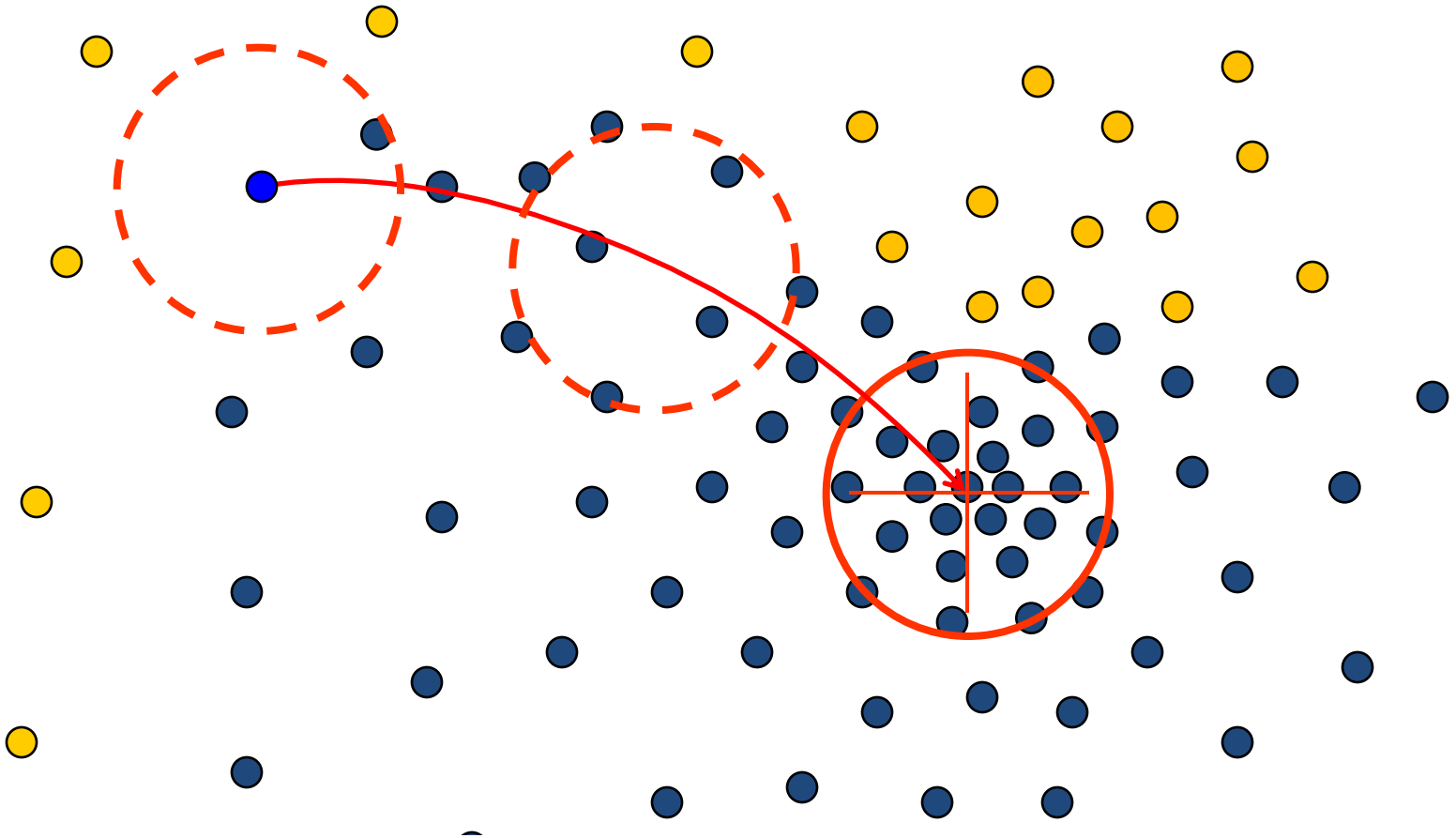
Mean-Shift Clustering

- 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

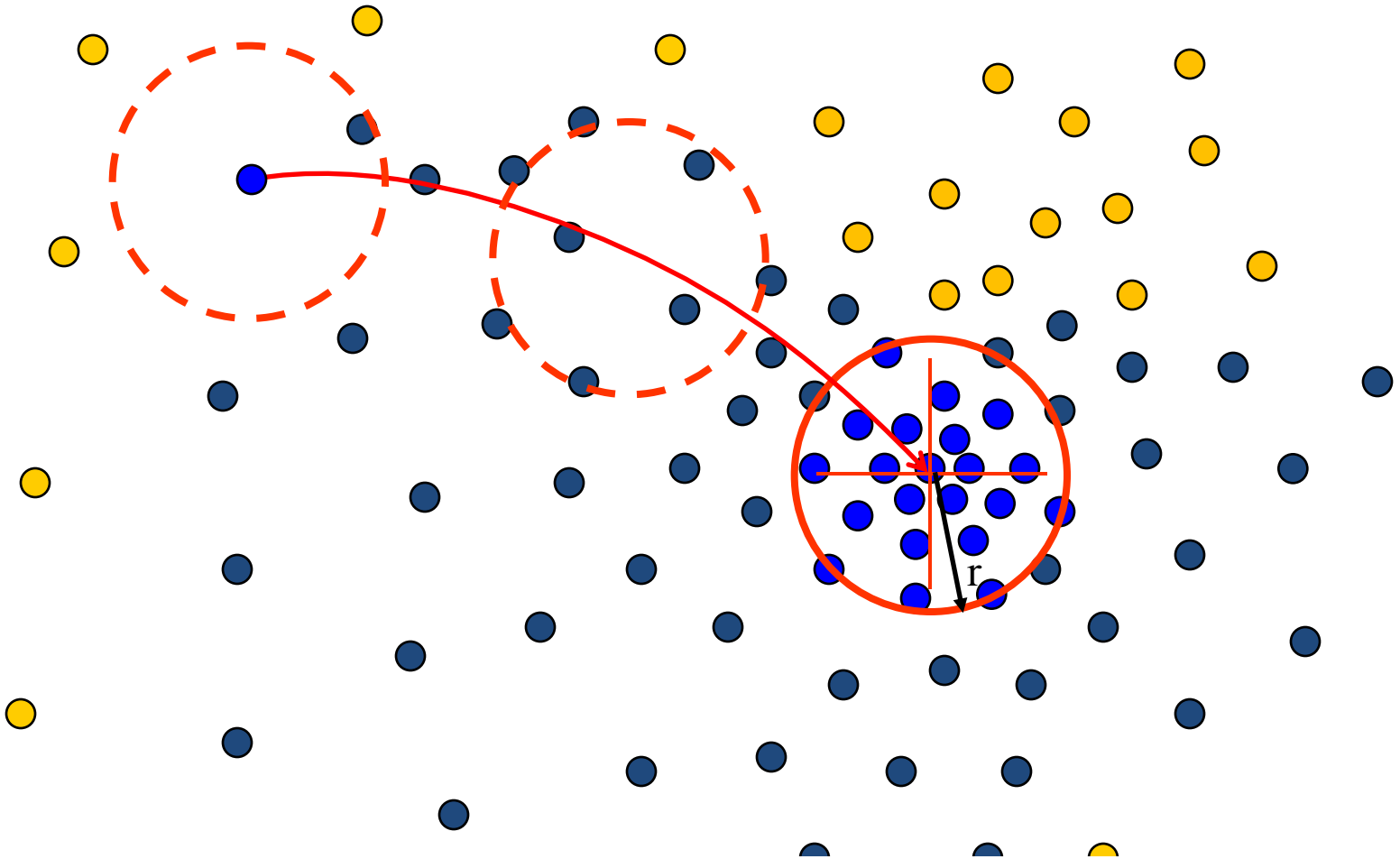
Problem: Computational Complexity



- Need to shift many windows…
- Many computations will be redundant.

Slide credit: Bastian Leibe

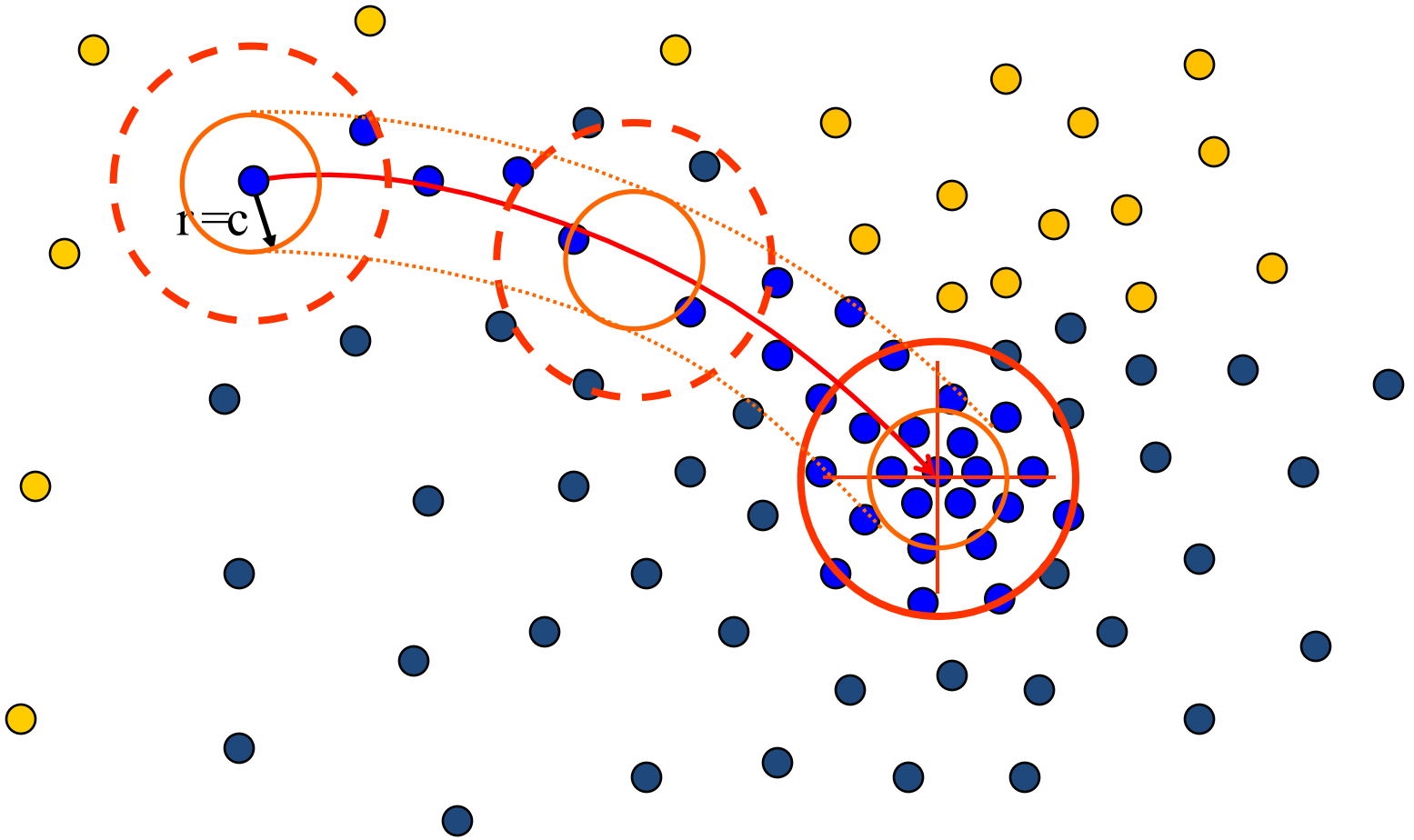
Speedups: Basin of Attraction



1. Assign all points within radius r of end point to the mode.

Slide credit: Bastian Leibe

Speedups



2. Assign all points within radius r/c of the search path to the mode -> reduce the number of data points to search.

Slide credit: Bastian Leibe

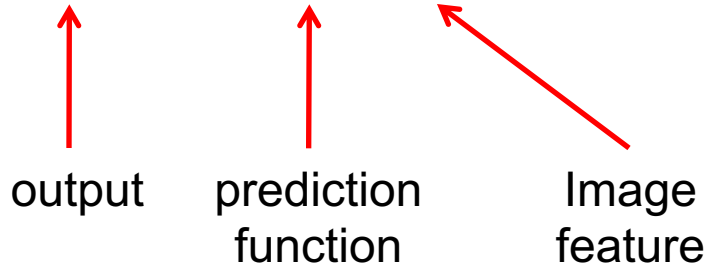
Summary Mean-Shift

- 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

The machine learning framework

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

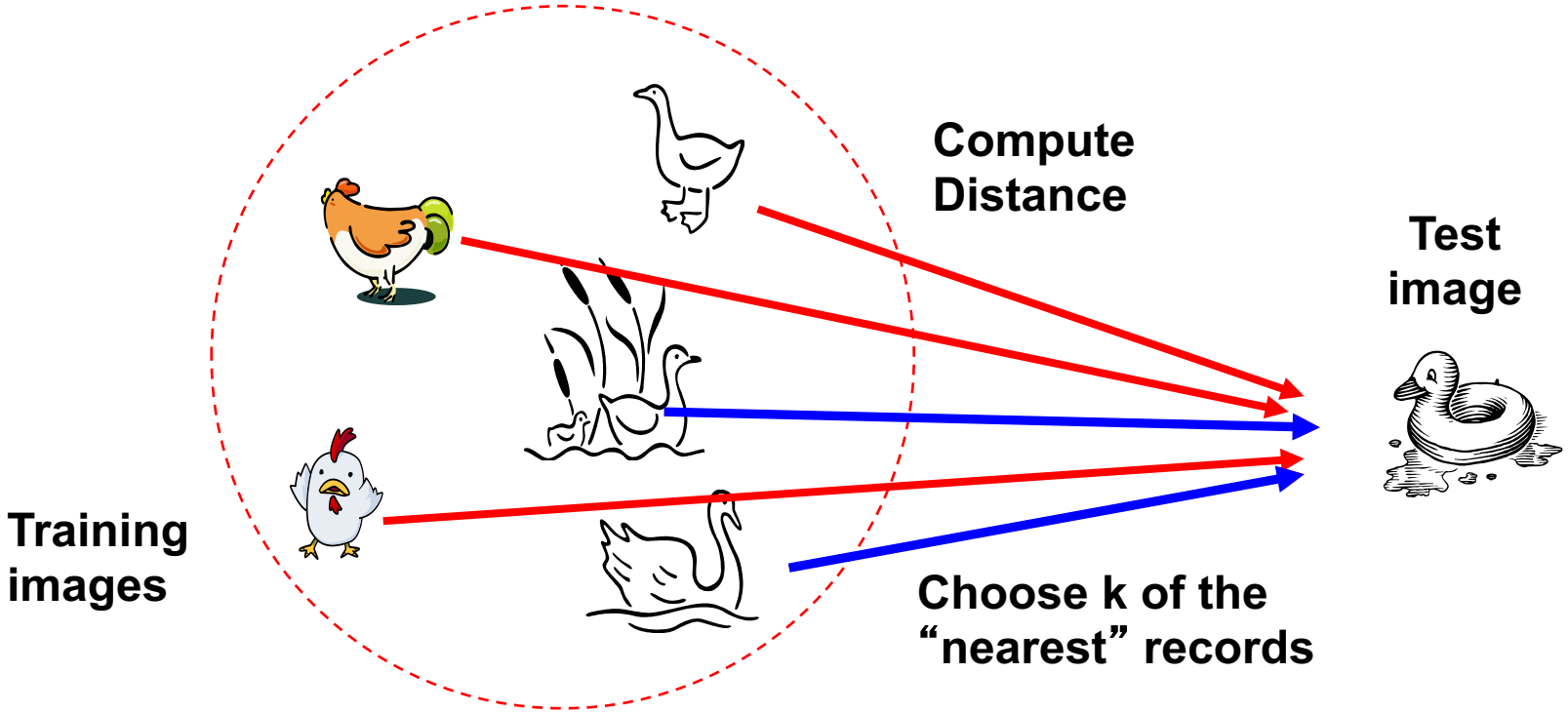


- **Training:** given a *training set* of labeled examples $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, estimate the prediction function f by minimizing the prediction error on the training set
- **Testing:** apply f to a never before seen *test example* $\mathbf{x}$ and output the predicted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

Slide credit: L. Lazebnik

Nearest Neighbor Classifier

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

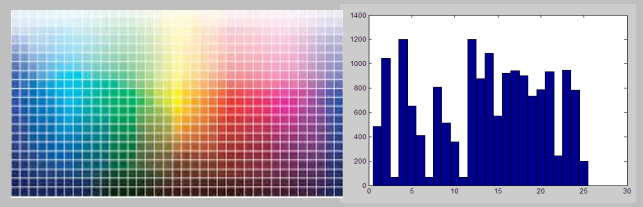


Source: N. Goyal

Image features

Input image

Color: Quantize RGB values

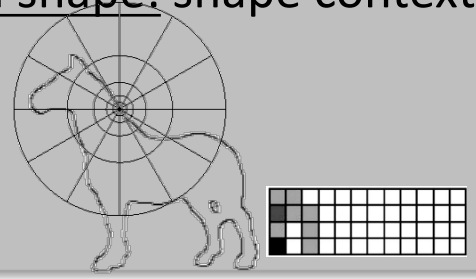


Invariance?
- Translation
- Scale
- Rotation
- Occlusion

Global shape: PCA space

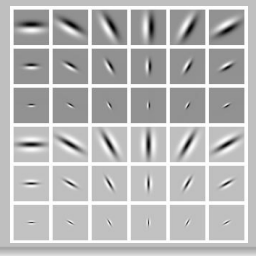
Invariance?
- Translation
- ? Scale
- Rotation
- Occlusion

Local shape: shape context



Invariance?
- Translation
- Scale
- ? Rotation (in-planar)
- Occlusion

Texture: Filter banks



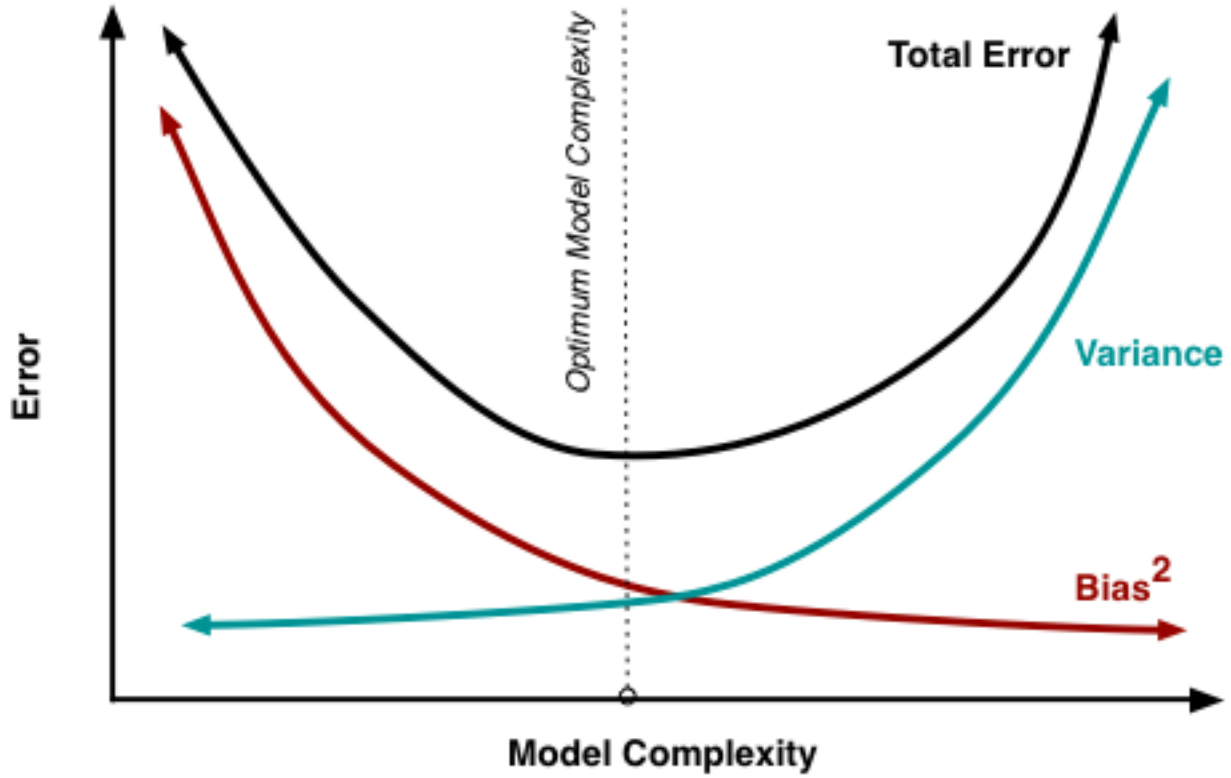
Invariance?
- Translation
- ? Scale
- ? Rotation (in-planar)
- Occlusion

Bias versus variance

- Components of generalization error
  - **Bias:** how much the average model over all training sets differ from the true model?
    - Error due to inaccurate assumptions/simplifications made by the model
  - **Variance:** how much models estimated from different training sets differ from each other
- **Underfitting:** model is too "simple" to represent all the relevant class characteristics
  - High bias and low variance
  - High training error and high test error
- **Overfitting:** model is too "complex" and fits irrelevant characteristics (noise) in the data
  - Low bias and high variance
  - Low training error and high test error

Slide credit: L. Lazebnik

Bias versus variance trade off



K-NN: issues to keep in mind

- Choosing the value of k:
  - If too small, sensitive to noise points
  - If too large, neighborhood may include points from other classes
  - Solution: cross validate!
- Can produce counter-intuitive results (using Euclidean measure)
  - Solution: normalize the vectors to unit length
- Curse of Dimensionality
  - Solution: no good one

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 $(0.001)^{1/d}$

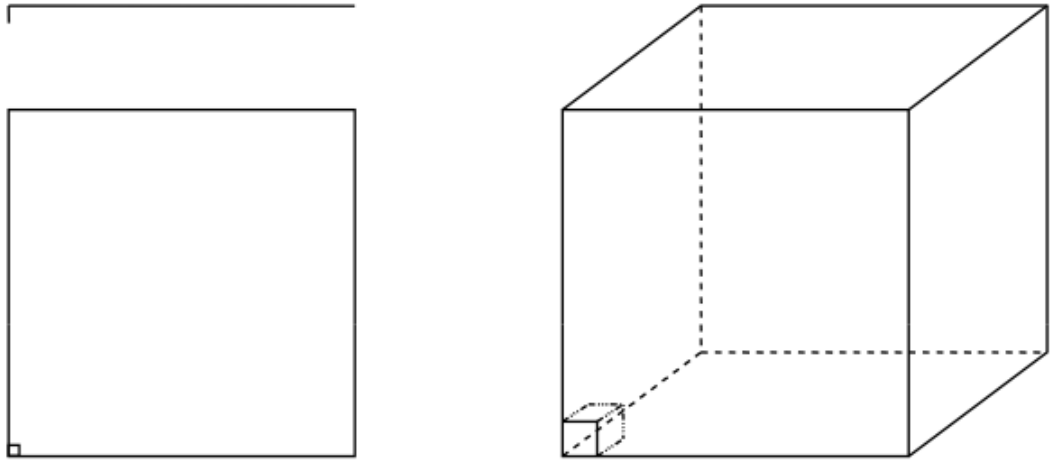
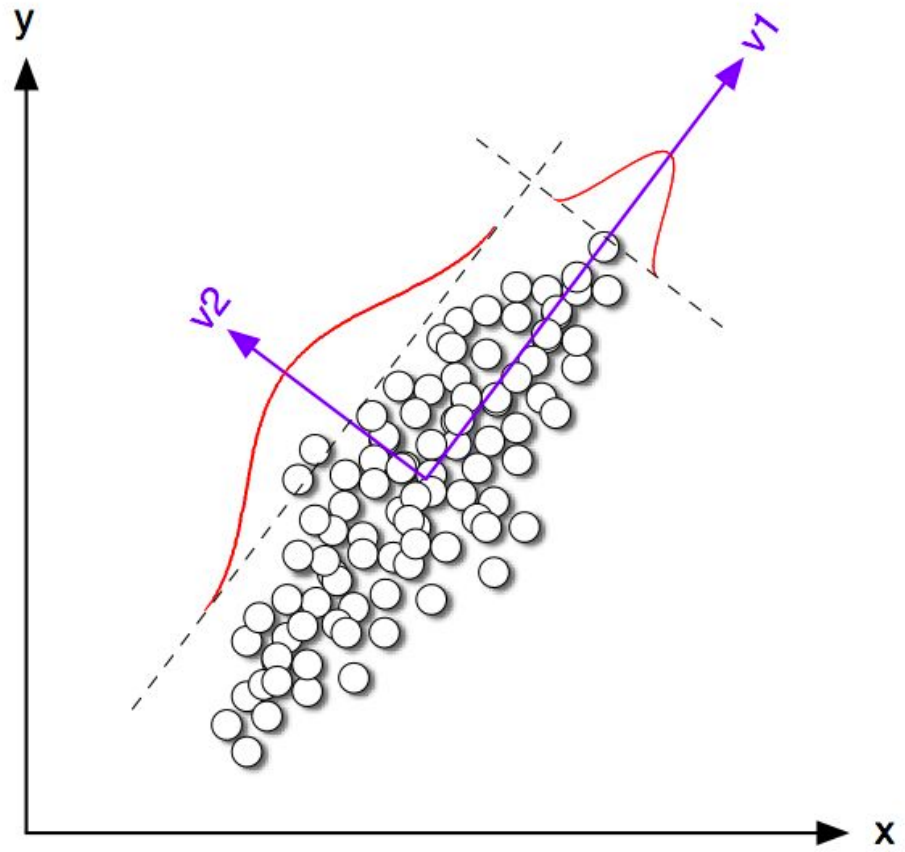


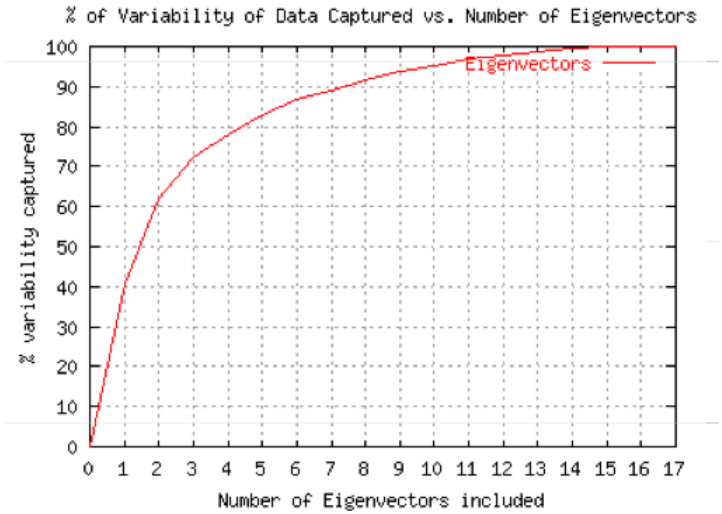
Image compression through PCA

Covariance between the two axis is high. Can we reduce the number of dimensions to just 1?



Rule of thumb for finding the number of PCA components

- A natural measure is to pick the eigenvectors that explain p% of the data variability
  - Can be done by plotting the ratio r_k as a function of k



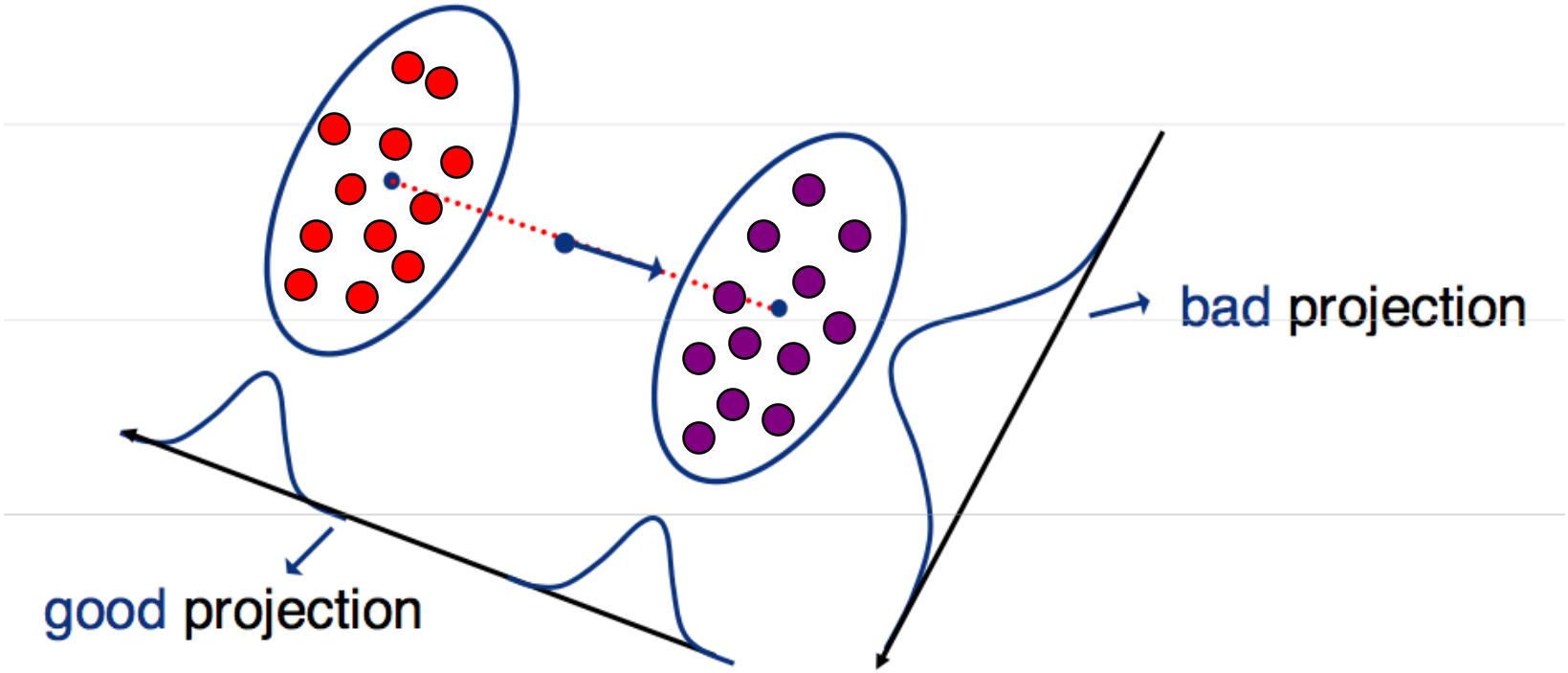
$$r_k = \frac{\sum_{i=1}^{k} \lambda_i^2}{\sum_{i=1}^{n} \lambda_i^2}$$

  - E.g. we need 3 eigenvectors to cover 70% of the variability of this dataset

Slide inspired by N. Vasconcelos

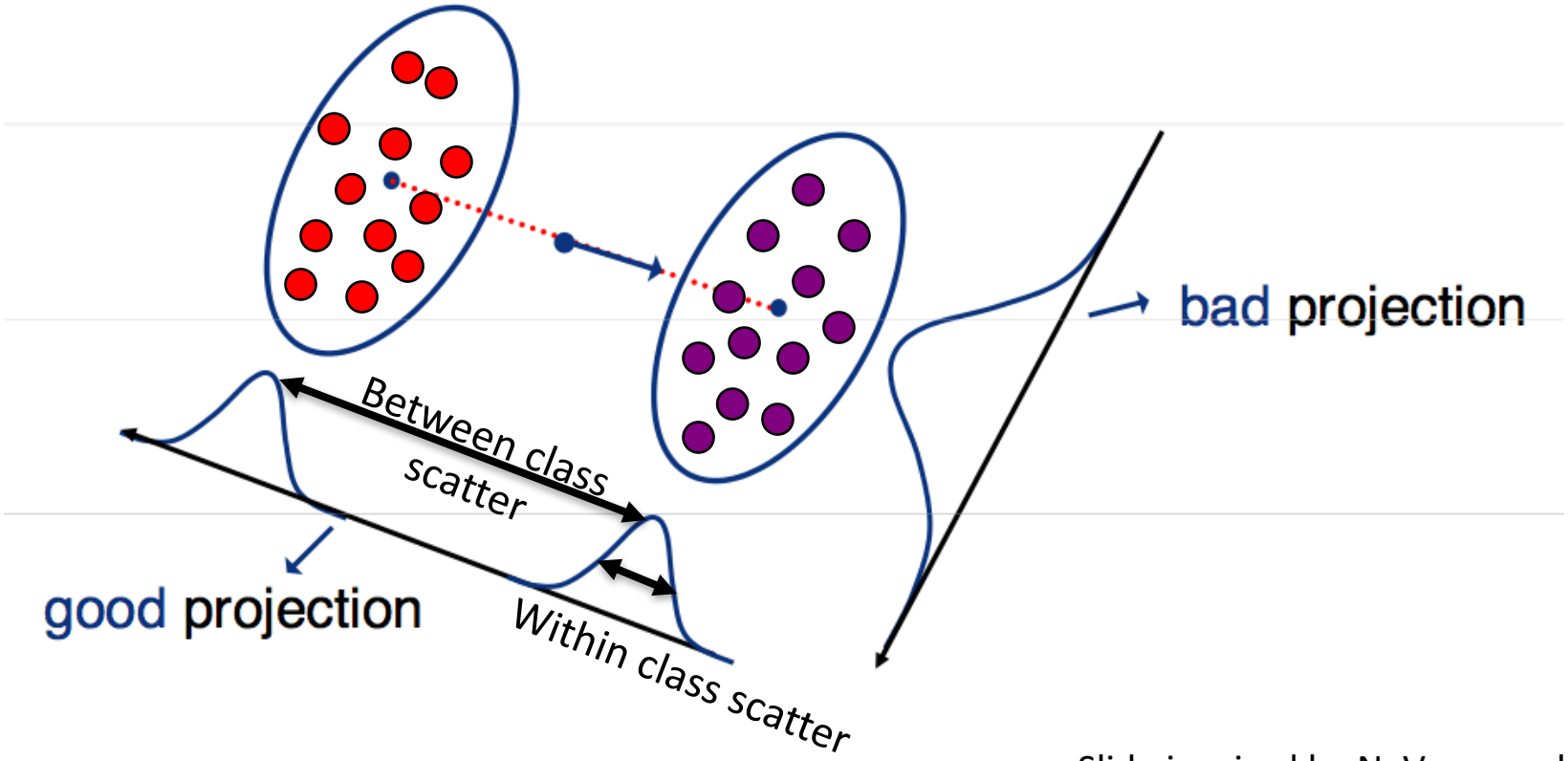
Fischer's Linear Discriminant Analysis

- Goal: find the best separation between two classes



Slide inspired by N. Vasconcelos

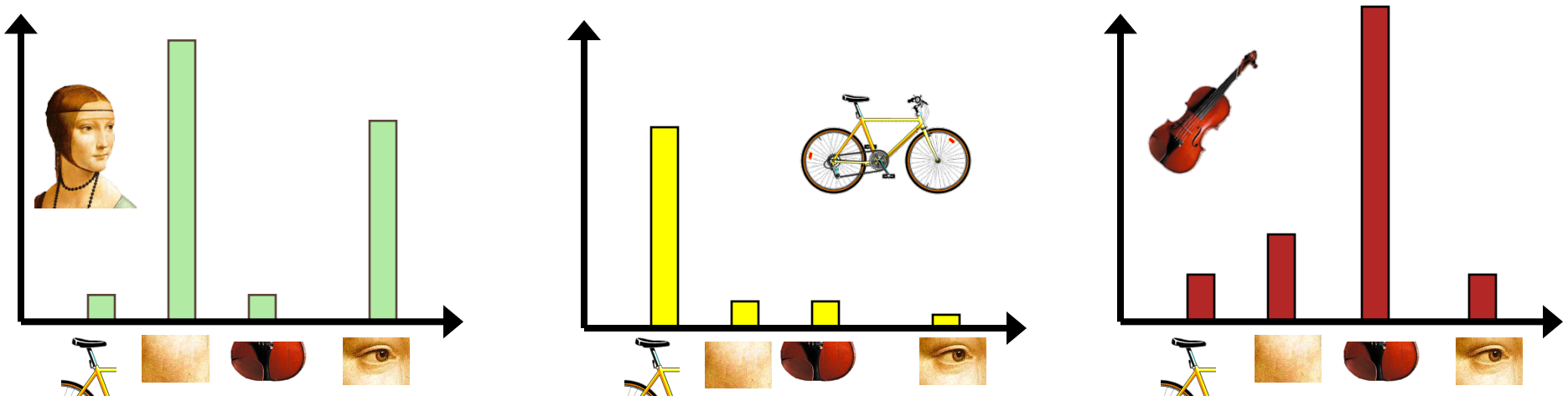
Fischer's Linear Discriminant Analysis



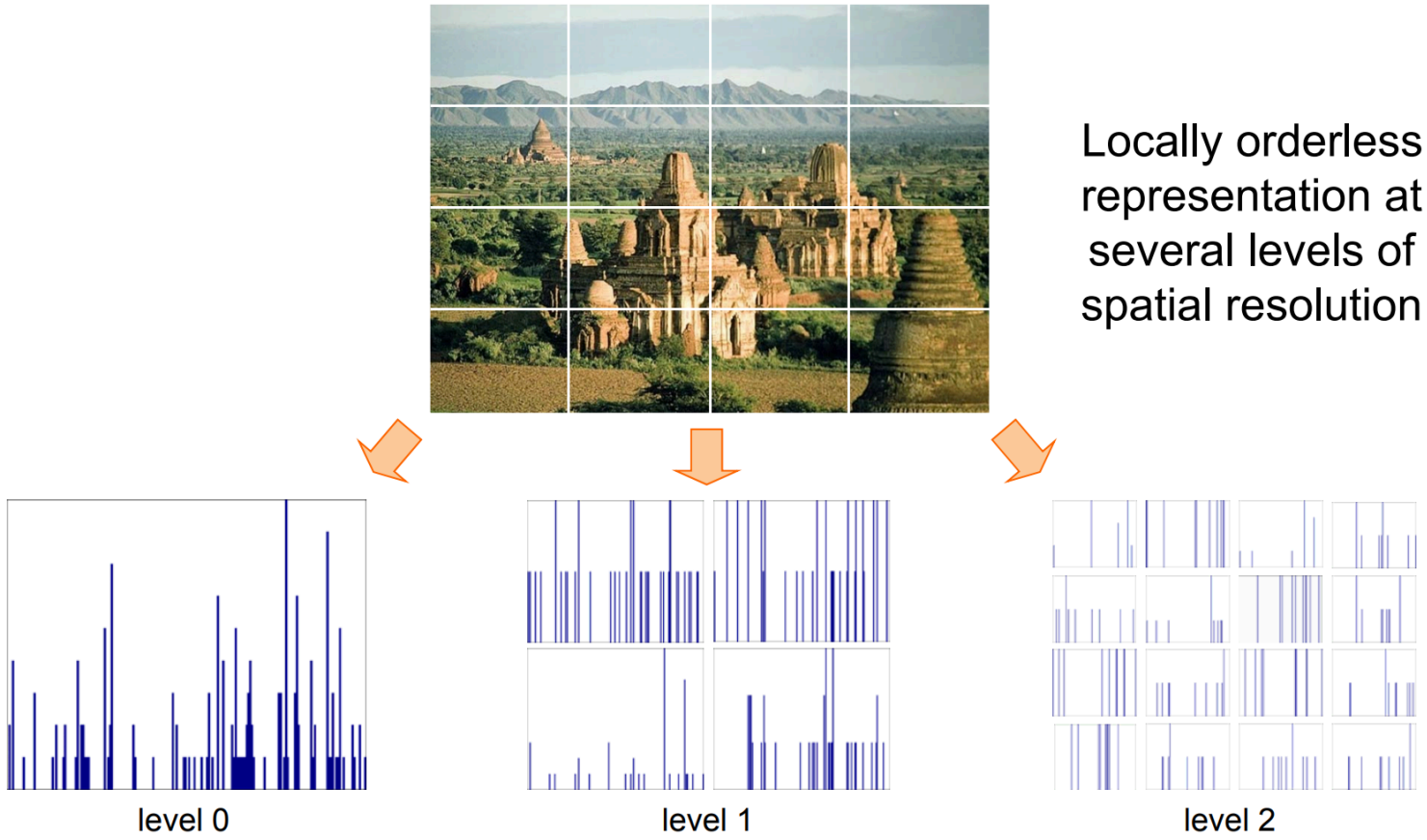
Slide inspired by N. Vasconcelos

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”

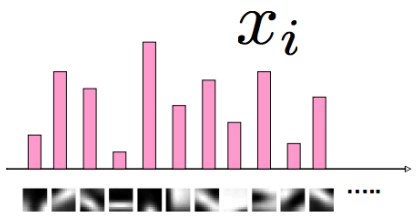


Bag of words + pyramids



Naïve Bayes

- Classify image using histograms of occurrences on visual words:

$$\mathbf{x} =$$ 

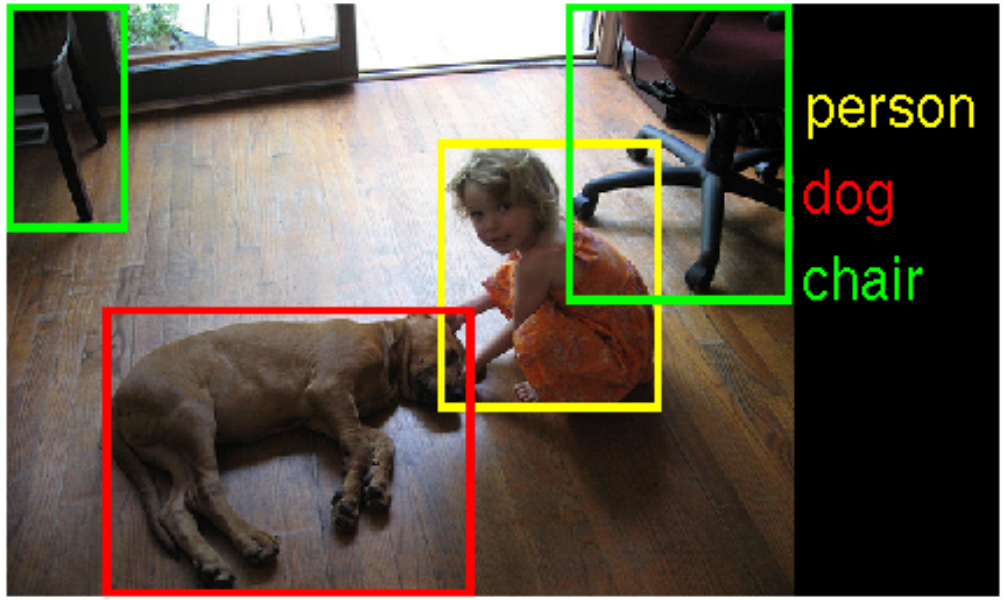
- if only present/absence of a word is taken into account: $x_i \in \{0, 1\}$
- Naïve Bayes classifier assumes that visual words are conditionally independent given object class

Csurka Bray, Dance & Fan, 2004

Object Detection

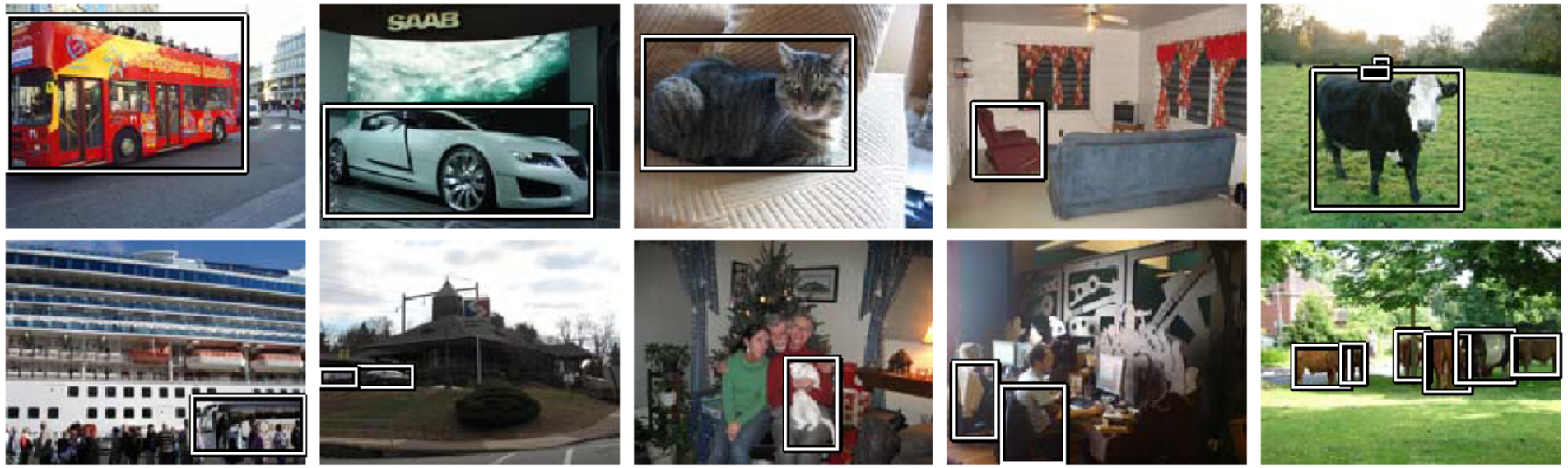
- **Problem**: Detecting and localizing generic objects from various categories, such as cars, people, etc.

- Challenges:
  - Illumination,
  - viewpoint,
  - deformations,
  - Intra-class variability



Object Detection Benchmarks

- PASCAL VOC Challenge



- 20 categories
- Annual classification, detection, segmentation, … challenges

	Predicted 1	Predicted 0
True 1	true positive	false negative
True 0	false positive	true negative

	Predicted 1	Predicted 0
True 1	TP	FN
True 0	FP	TN

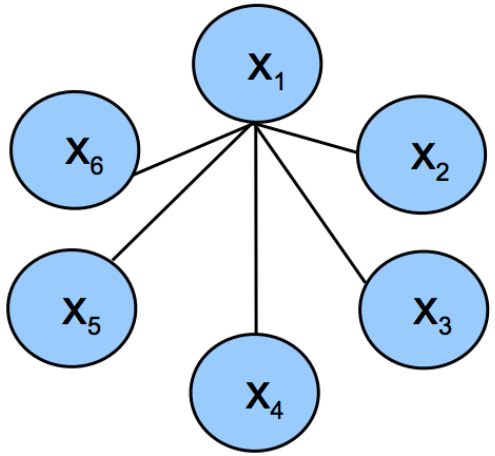
	Predicted 1	Predicted 0
True 1	hits	misses
True 0	false alarms	correct rejections

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

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



DPM - discussion

- Approach
  - Manually selected set of parts - Specific detector trained for each part
  - Spatial model trained on part activations
  - Evaluate joint likelihood of part activations
- Advantages
  - Parts have intuitive meaning.
  - Standard detection approaches can be used for each part.
  - Works well for specific categories.
- Disadvantages
  - Parts need to be selected manually
  - Semantically motivated parts sometimes don’t have a simple appearance distribution
  - No guarantee that some important part hasn’t been missed
- When switching to another category, the model has to be rebuilt from scratch.

From Pascal VOC to ImageNet

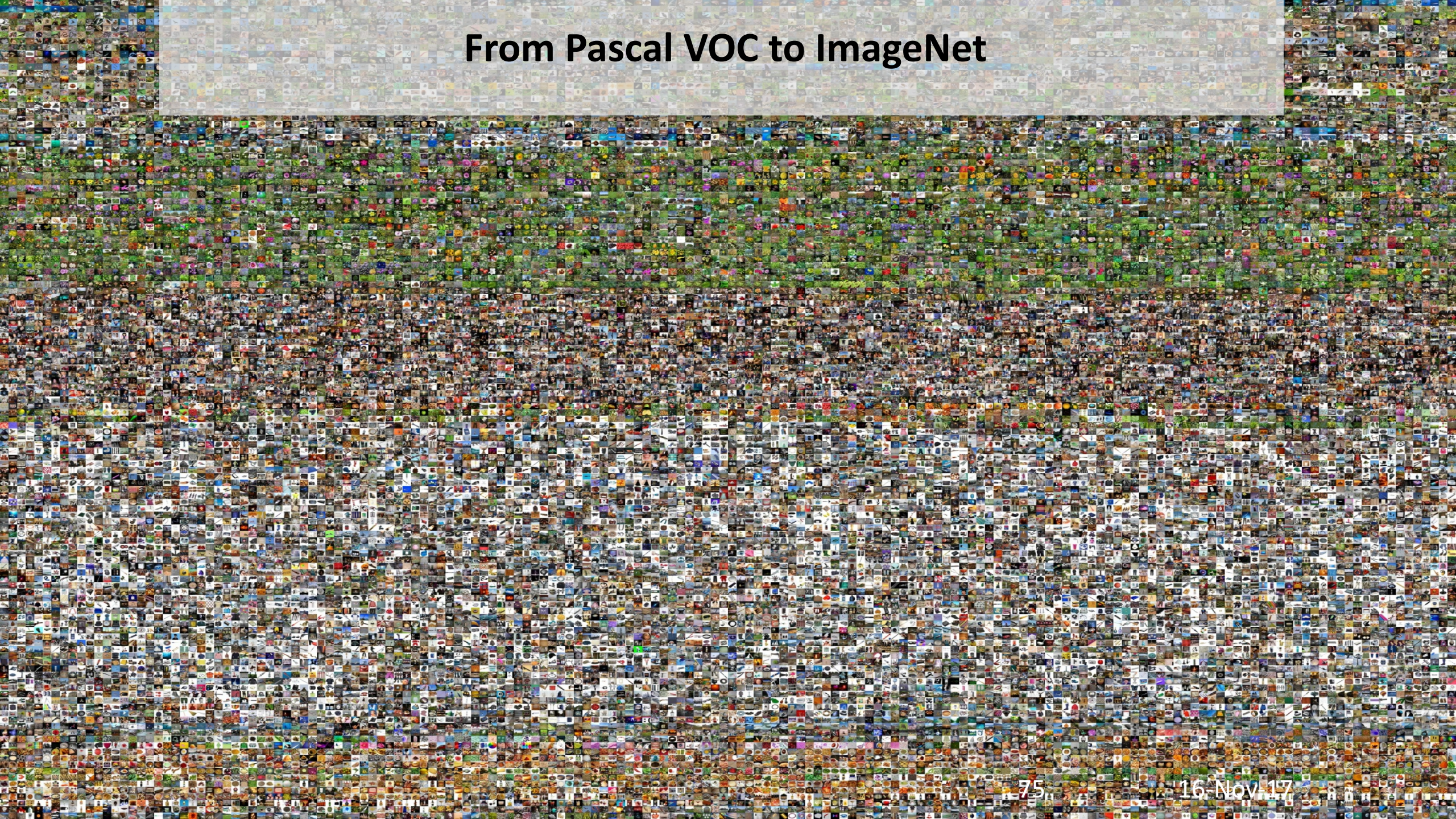
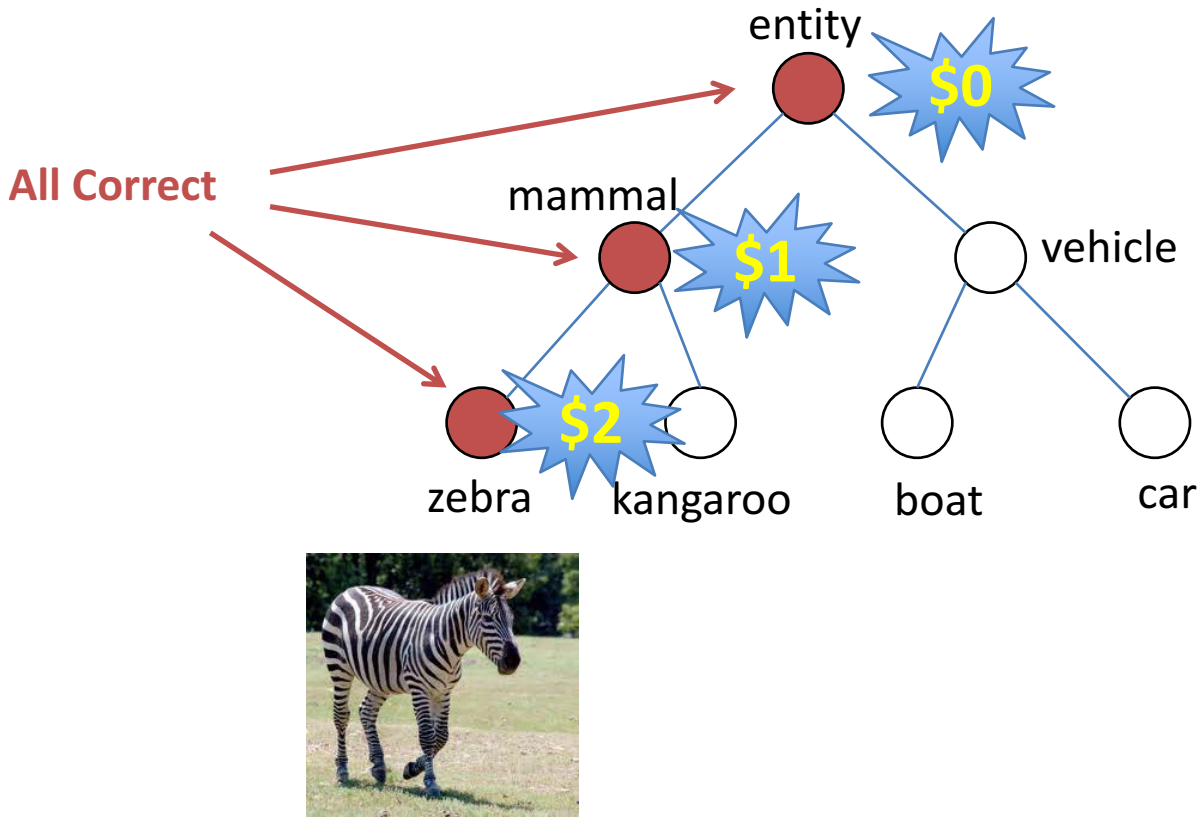
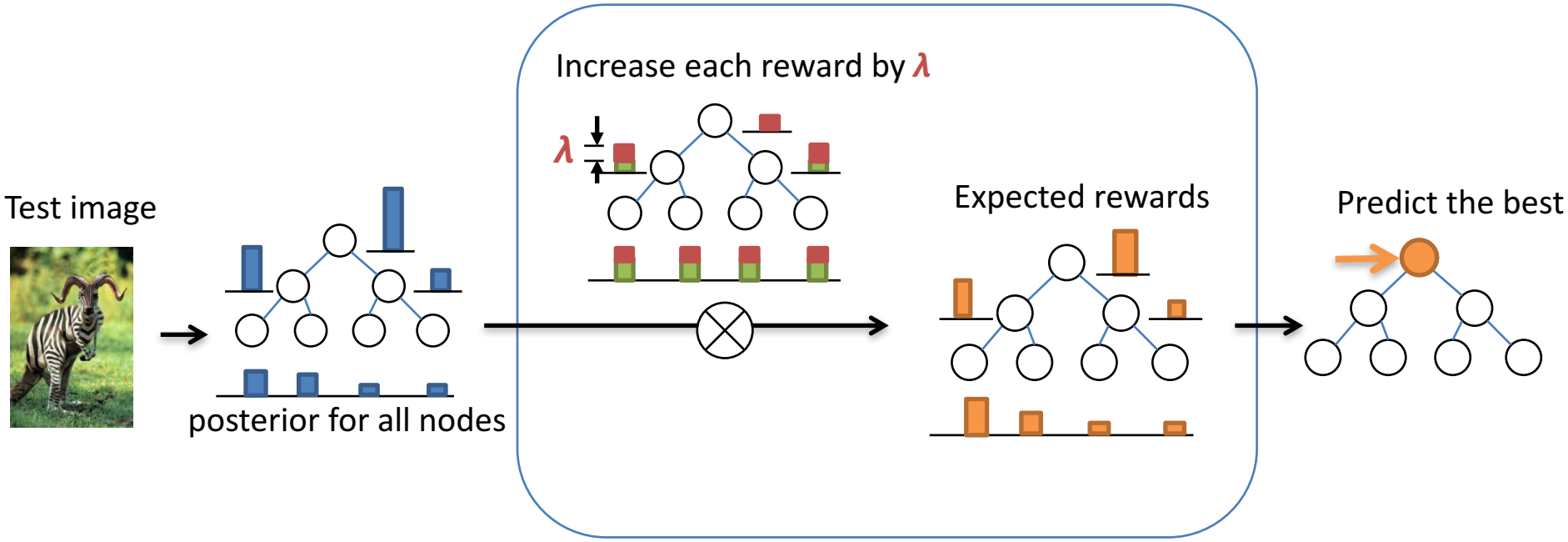


Image classification using Semantic hierarchy

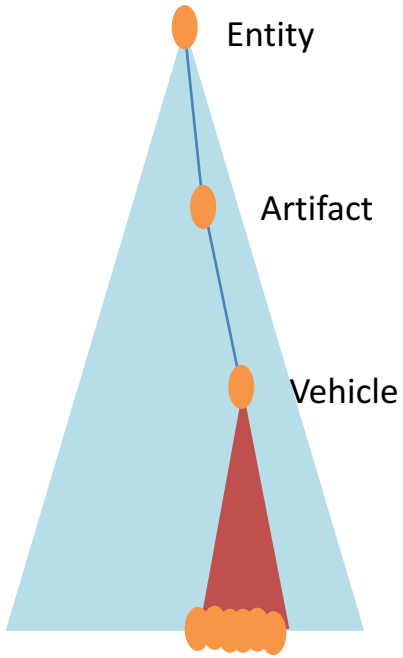


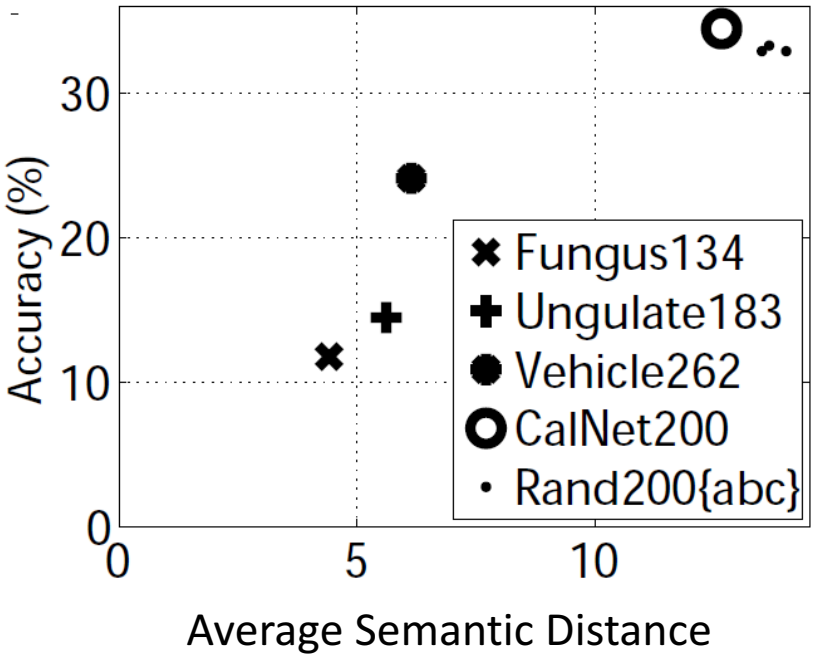
A global, fixed scalar parameter $\lambda \geq 0$

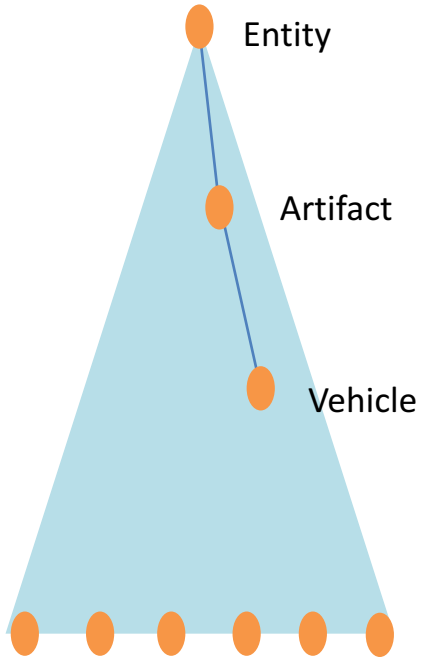


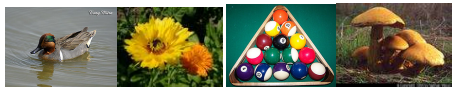
Deng, Krause, Berg, Fei-Fei, CVPR2012

Fine-grained categories are a lot harder









Finer **Coarser**

Fine-Grained Recognition

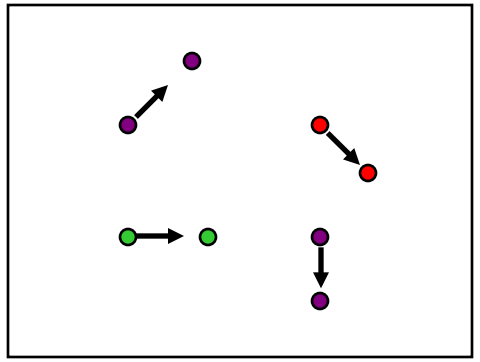
What breed is this dog?

Key: Find the right features.

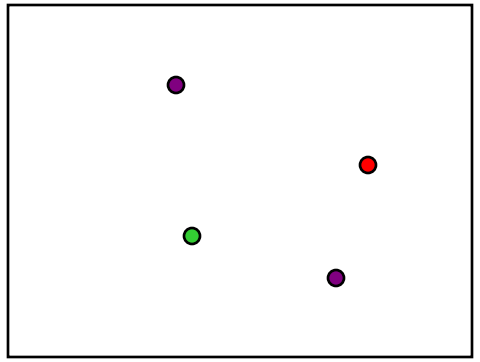
The Bubbles game

?

Estimating optical flow



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



$I(x,y,t)$

- 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

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

$A^T A$ $\qquad$ $A^T b$

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

Errors in Lukas-Kanade

What are the potential causes of errors in this procedure?

- Suppose A^TA is easily invertible
- Suppose there is not much noise in the image

- When our assumptions are violated
  - Brightness constancy is **not** satisfied
  - The motion is **not** small
  - A point does **not** move like its neighbors
    - window size is too large
    - what is the ideal window size?

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Horn-Schunk method for optical flow

- The flow is formulated as a global energy function which is should be minimized:

$$E = \iint \left[(I_x u + I_y v + I_t)^2 + \alpha^2 (\|\nabla u\|^2 + \|\nabla v\|^2) \right] \mathrm{d}x \mathrm{d}y$$

- The second part is the smoothness constraint. It's trying to make sure that the changes between frames are small.

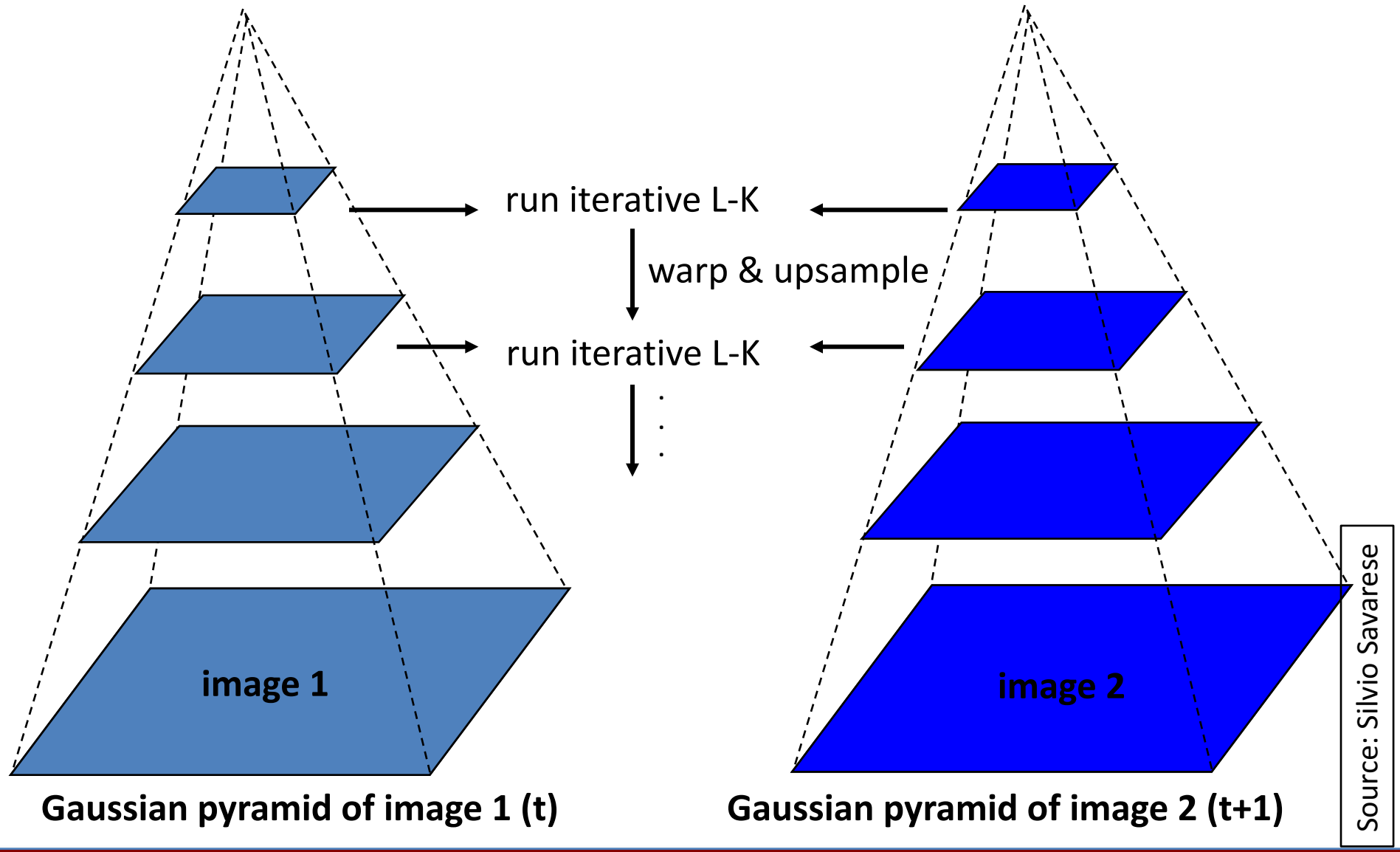
Iterative Horn-Schunk

- But since the solution depends on the neighboring values of the flow field, it must be repeated once the neighbors have been updated.
- So instead, we can iteratively solve for u and v using:

$$u^{k+1} = \bar{u}^k - \frac{I_x(I_x\bar{u}^k + I_y\bar{v}^k + I_t)}{\alpha^2 + I_x^2 + I_y^2}$$

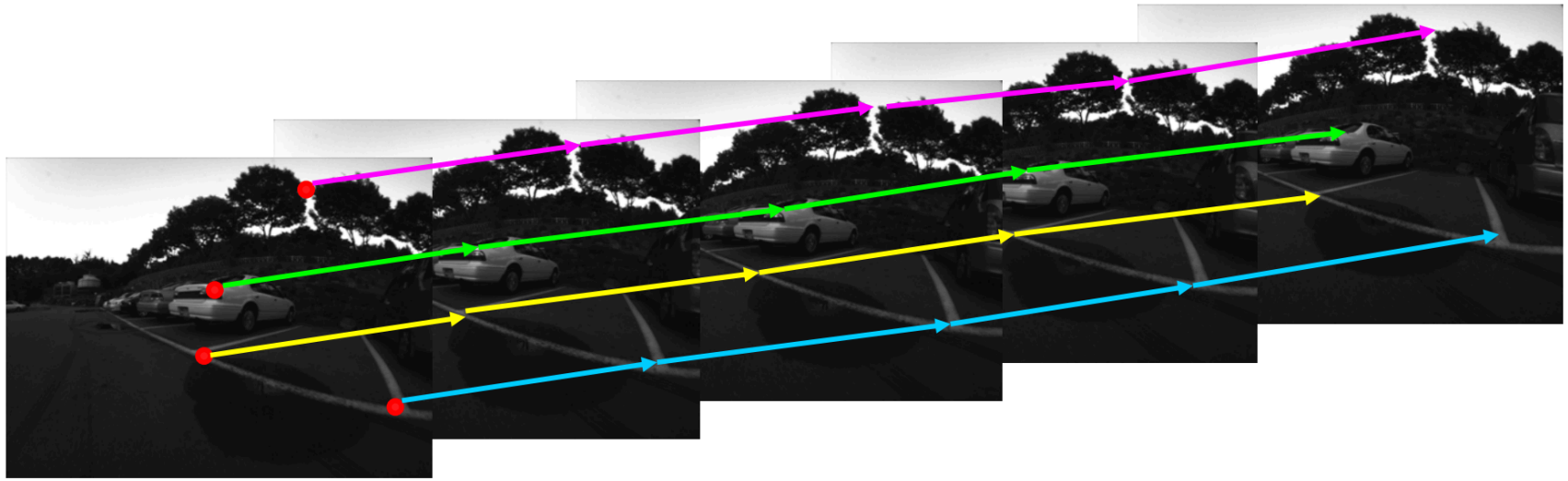
$$v^{k+1} = \bar{v}^k - \frac{I_y(I_x\bar{u}^k + I_y\bar{v}^k + I_t)}{\alpha^2 + I_x^2 + I_y^2}$$

Coarse-to-fine optical flow estimation



Tracking

Feature point tracking

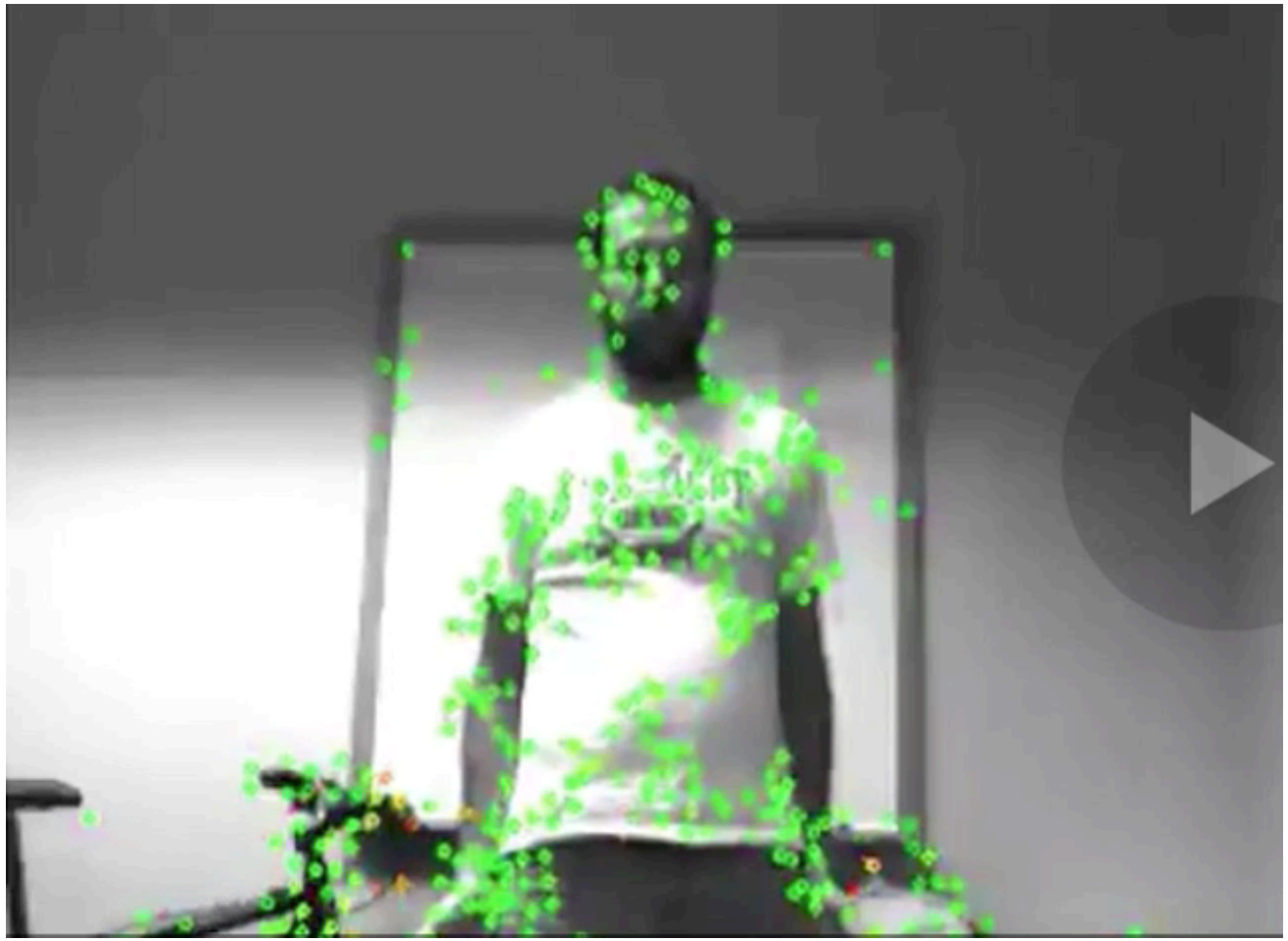


Slide credit: Yonsei Univ.

Challenges in Feature tracking

- Figure out which features can be tracked
  - Efficiently track across frames
- Some points may change appearance over time
  - e.g., due to rotation, moving into shadows, etc.
- Drift: small errors can accumulate as appearance model is updated
- Points may appear or disappear.
  - need to be able to add/delete tracked points.

Tracking movement



Video credit: Kanade

Affine motion

- Affine motion includes scaling + rotation + translation.
- $x' = a_1x + a_2y + b1$
  $y' = a_3x + a_4y + b_2$
- $W = \begin{bmatrix} a_1 & a_2 & b_1 \\ a_3 & a_4 & b_2 \end{bmatrix}$
- $\boldsymbol{p} = [a_1 \quad a_2 \quad b_1 \quad a_3 \quad a_4 \quad b_2]^T$
- $\frac{\partial W}{\partial \boldsymbol{p}}(\boldsymbol{x};\boldsymbol{p}) = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix}$

Overall KLT tracker algorithm

Given the features from Harris detector:

1. Initialize $\boldsymbol{p_0}$ and $\Delta\boldsymbol{p}$.
2. Compute the initial templates $T(x)$ for each feature.
3. Transform the features in the image I with $W(\boldsymbol{x};\boldsymbol{p_0})$.
4. Measure the error: $I(W(\boldsymbol{x};\boldsymbol{p_0})) - T(x)$.
5. Compute the image gradients $\nabla I = [I_x \quad I_y]$.
6. Evaluate the Jacobian $\frac{\partial W}{\partial \boldsymbol{p}}$.
7. Compute steepest descent $\nabla I \frac{\partial W}{\partial \boldsymbol{p}}$.
8. Compute Inverse Hessian H^{-1}
9. Calculate the change in parameters $\Delta\boldsymbol{p}$
10. Update parameters $\boldsymbol{p_0} = \boldsymbol{p_0} + \Delta\boldsymbol{p}$
11. Repeat 2 to 10 until $\Delta\boldsymbol{p}$ is small.

Challenges to consider

- Implementation issues
- Window size
  - Small window more sensitive to noise and may miss larger motions (without pyramid)
  - Large window more likely to cross an occlusion boundary (and it's slower)
  - 15x15 to 31x31 seems typical
- Weighting the window
  - Common to apply weights so that center matters more (e.g., with Gaussian)

What should you take away from this class?

- A broad understanding of computer vision as a field.
- Learning to use common softwares like github, jupyter notebooks.
- Writing math equations using latex
- Converting between math equations and python code.
- Learning to communicate ideas and algorithms in formal writing.

Next classes to take

- Continue learning more about computer vision
  - CS 231A – camera models and 3D vision
  - CS 231N – Convolutional neural networks
  - CS 331B – Representation learning
- Take more machine learning and AI classes:
  - CS 229 – machine learning fundamentals (must take)
  - CS 221 – similar to this class as it covers a breath of AI tasks
  - CS 228 – probabilistic graphical models
  - CS 362a – convex optimization (to truly understand machine learning)

Come work with us

- Come TA for us next year
  - It will be significantly less work next year
- Join the vision and learning lab as a researcher
  - From self driving cars to never ending learning AI systems.
  - Publish papers with us!
  - We expect 10-15 hours a week of research commitment.

How to find research opportunities?

- Find papers you are interested in from vision conferences:
  - Ex, CVPR2017, ICCV2017
- Find who in vision works on similar projects.
  - Link to people in the vision lab.
- Take the classes you need to be resourceful.
- Email us with ideas you want to work with.
- For undergraduates: apply to be a CURIS research summer internship.

Feedback – it matters a lot

- Feel free to email me with feedback.
  - Highly appreciated.
- Fill out anonymous class evaluations on axess.stanford.edu
- Fill out reviews on carta.stanford.edu

Feedback – it matters a lot!

- Assignments
  - Which assignments were too easy? Too hard?
- Topics
  - Which topics did you enjoy the most and would recommend we keep and which ones did we not cover that you would like to include?
- Class notes and extra credit
  - Did you find them to be educational?
- Lectures
  - Were they engaging? How can we improve them?