Object Recognition

DD2423 Image Analysis and Computer Vision

Mårten Björkman

Computational Vision and Active Perception
School of Computer Science and Communication

December 1, 2015

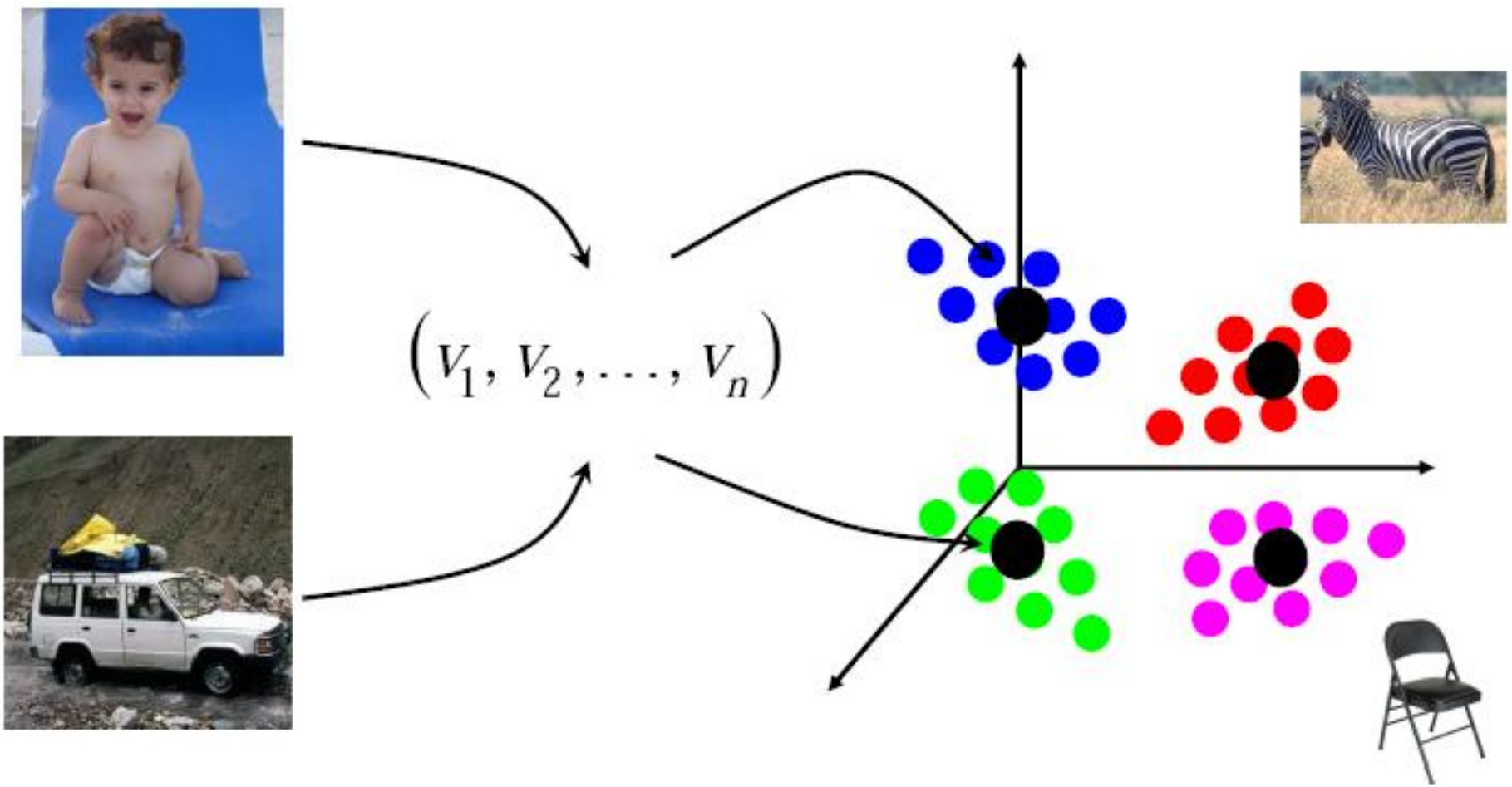
Goals

- Recognition - Is this my cup?
  - Enough for many applications! Prior knowledge available.
- Classification - What is this?
  - What do all cups have in common?
  - How to group objects into classes?

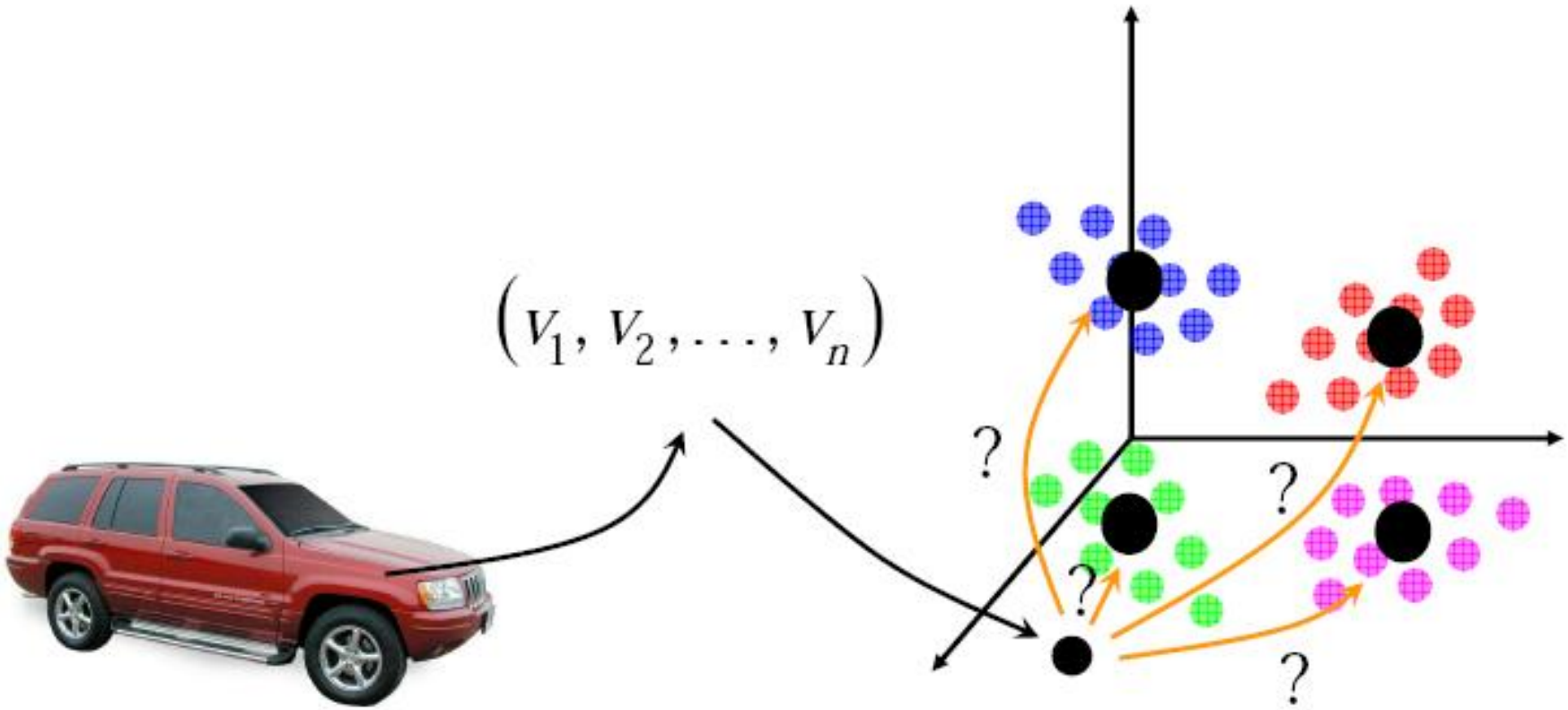
Common for all methods

- Two or more hypotheses.
  - Is this a cup or not?
  - Is this a book, cup, monkey, pen or ...?
- Need for representation.
  - As small as possible, otherwise redundant.
  - A database of objects can be very large.
- Need for metric, such that representations of
  - ... the same class are close.
  - ... different classes are distant.

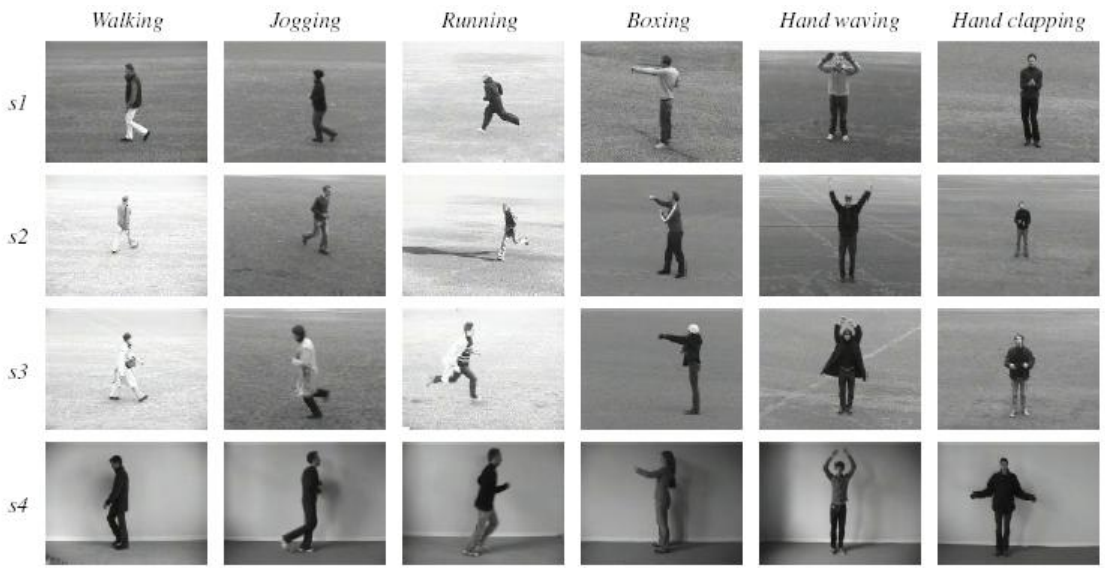
Recognition



Recognition



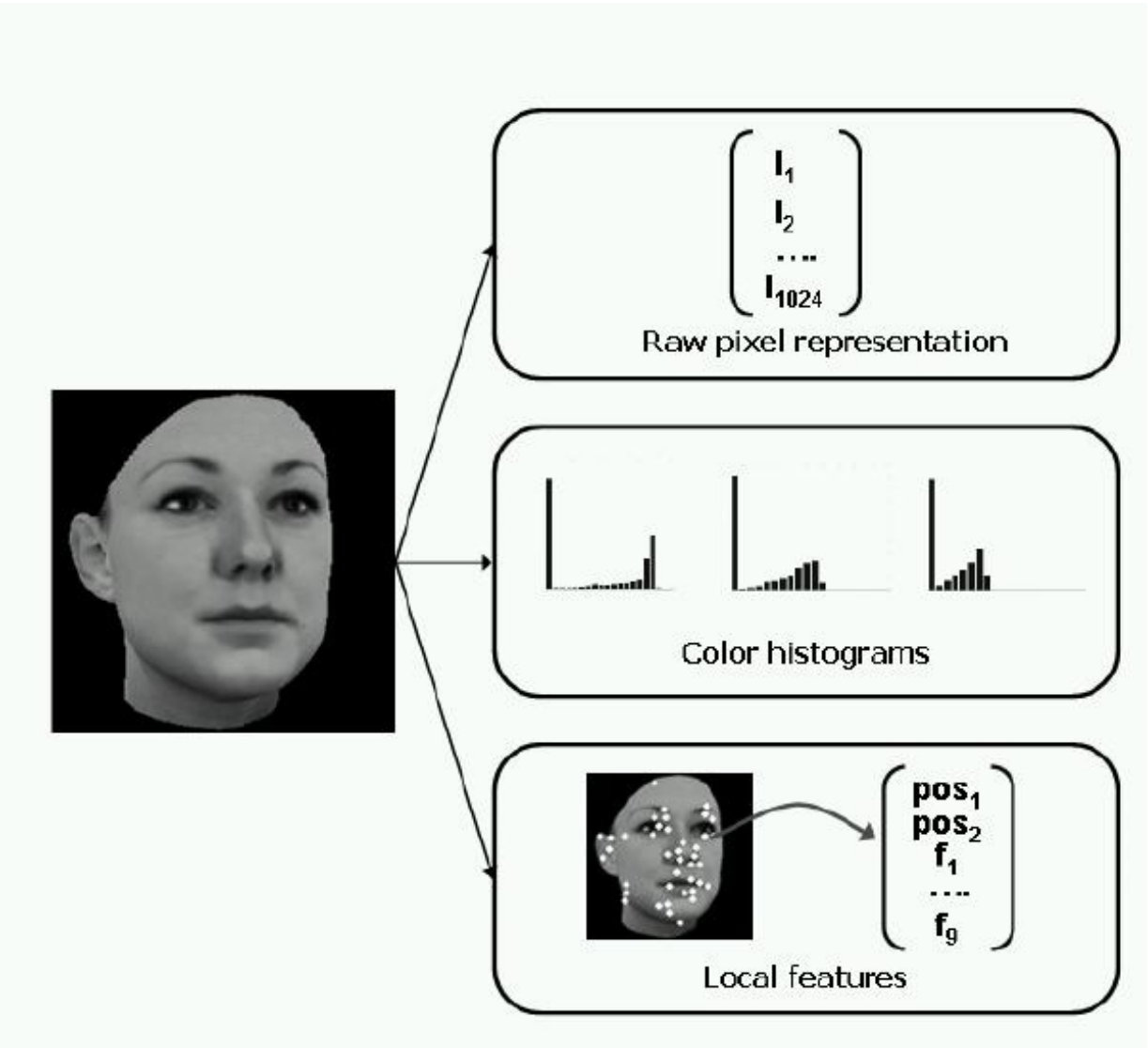
Recognition (example)



	Walk	Jog	Run	Box	Hclp	Hwav
Walk	**83.8**	16.2	0.0	0.0	0.0	0.0
Jog	22.9	**60.4**	16.7	0.0	0.0	0.0
Run	6.3	38.9	**54.9**	0.0	0.0	0.0
Box	0.7	0.0	0.0	**97.9**	0.7	0.7
Hclp	1.4	0.0	0.0	35.4	**59.7**	3.5
Hwav	0.7	0.0	0.0	20.8	4.9	**73.6**

Classification using silhouettes of people

Different representations



Differences between methods

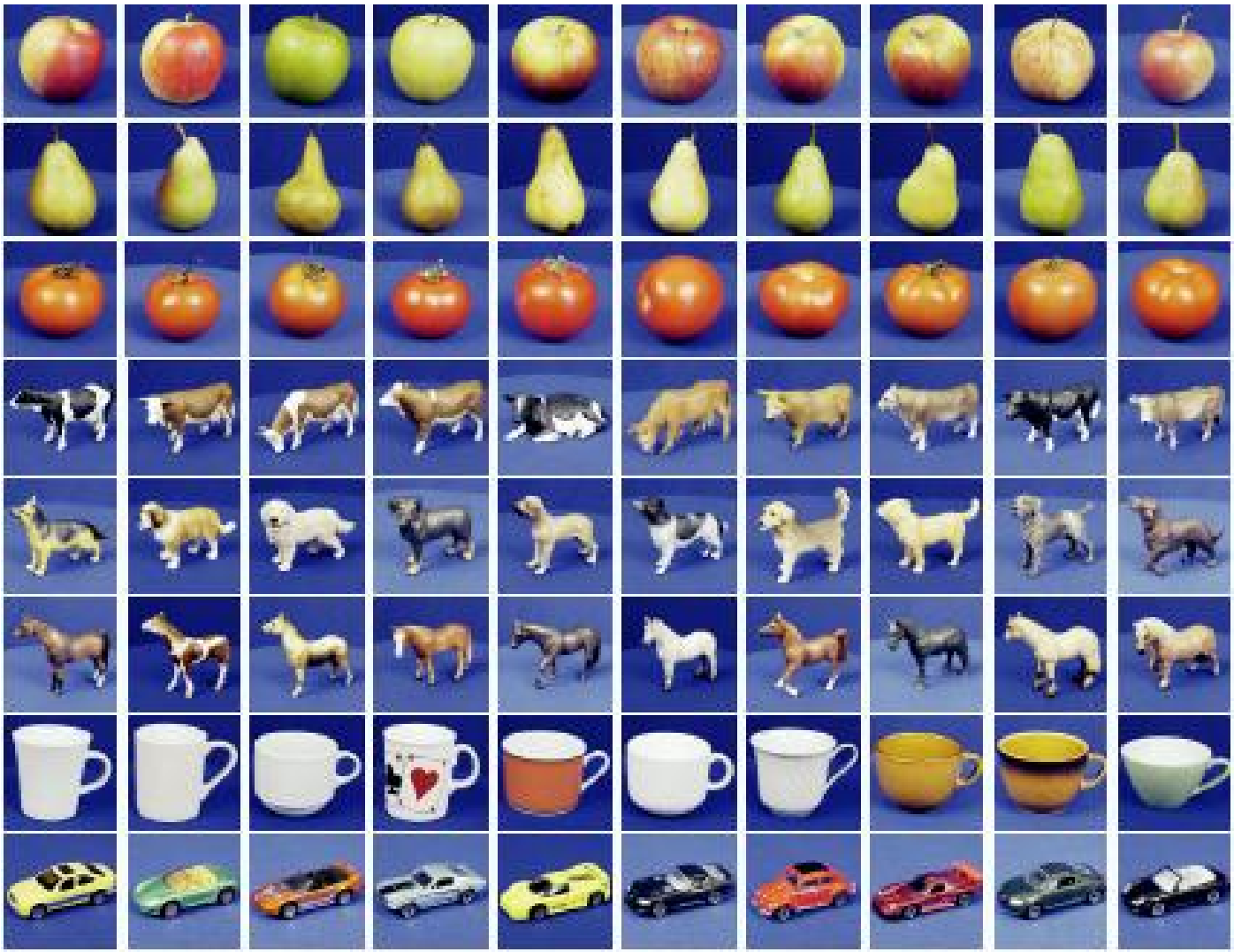
- Supervised vs unsupervised
  - Supervised: annotated training examples exist
  - Unsupervised (clustering): no annotated training examples
- Single-class vs multi-class
  - Single-class: Is it a cup or not?
  - Multi-class: Is it a cup, car, person, horse...?
- Generative vs discriminative
  - Generative: models how things look like
    ex. face = 2 eyes + 1 nose + 1 mouth
  - Discriminative: models the difference between things
    ex. face vs non-face

Computational steps

- Training
  - Preprocessing (segment, scale, ...)
  - Feature extraction
  - Classifier training
- Recognition
  - Preprocessing (segment, scale, ...)
  - Feature extraction
  - Recognition/Classification

Database for recognition (COIL 100 Database)

Database for categorization



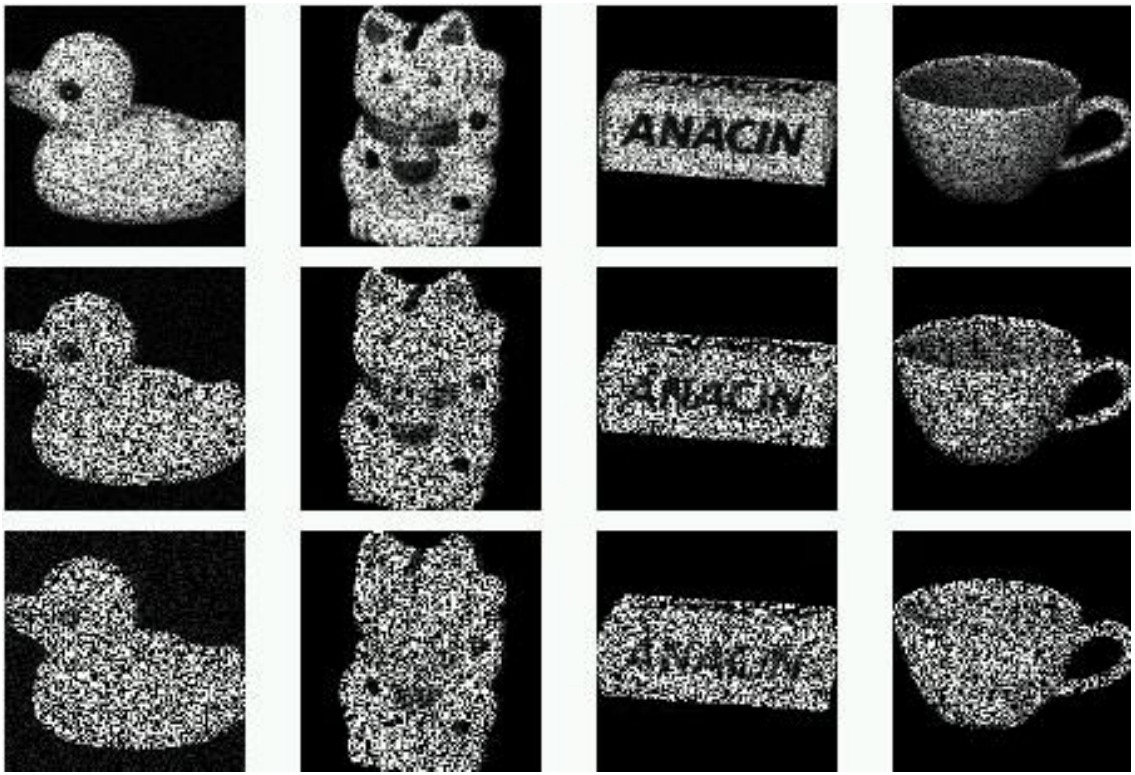
Invariance

- Appearance of object may vary due to different:
  - Lighting conditions (shadows, colours)
  - Poses (viewing directions)
  - Deformations (changes in shape)
  - Clutter (occlusions)
  - Projective sizes (distances)
  - Cameras (projections, distortions)
- Thus... matching has to be (more or less) invariant to these.

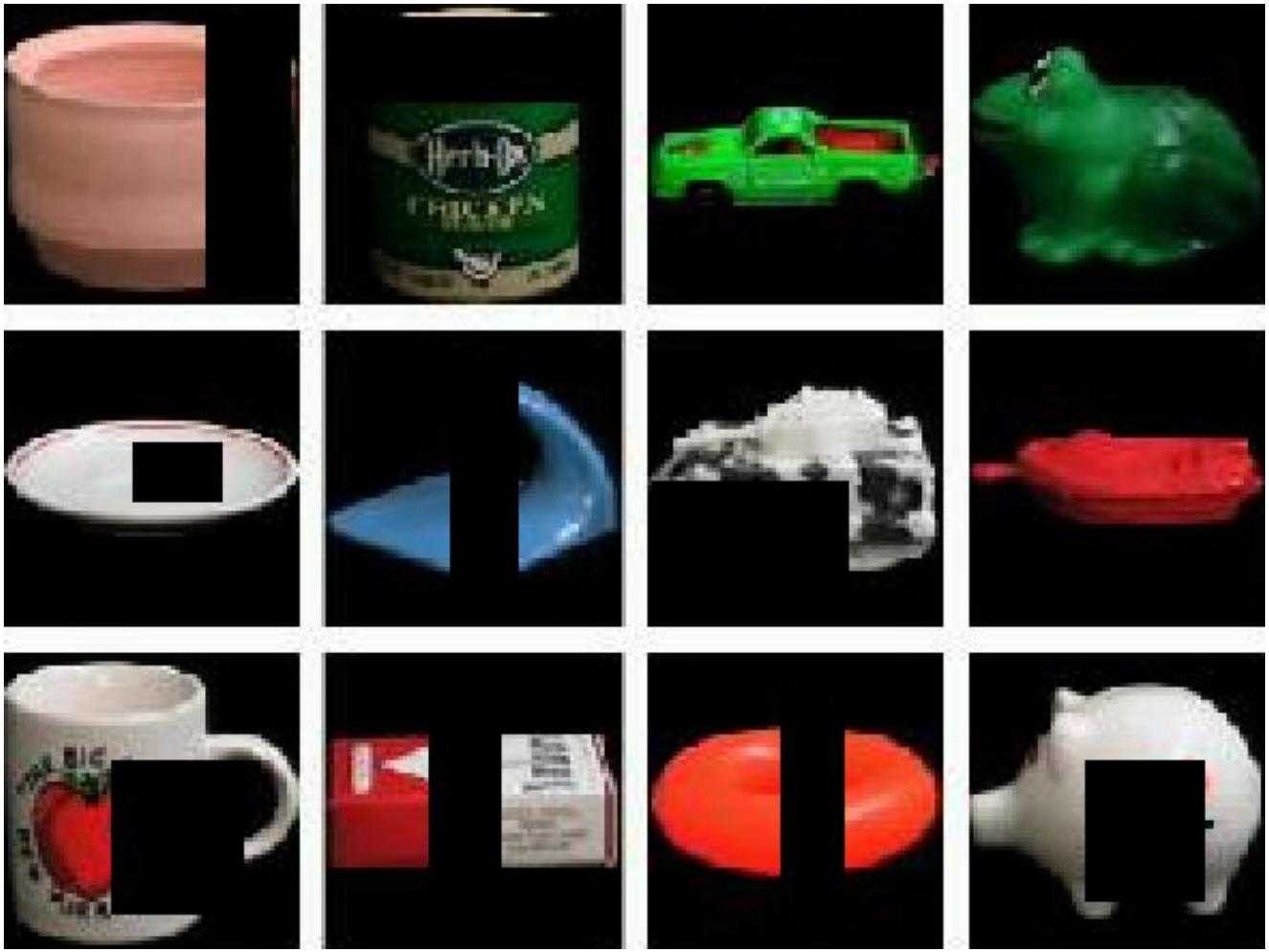
Illumination Invariance

Shadows

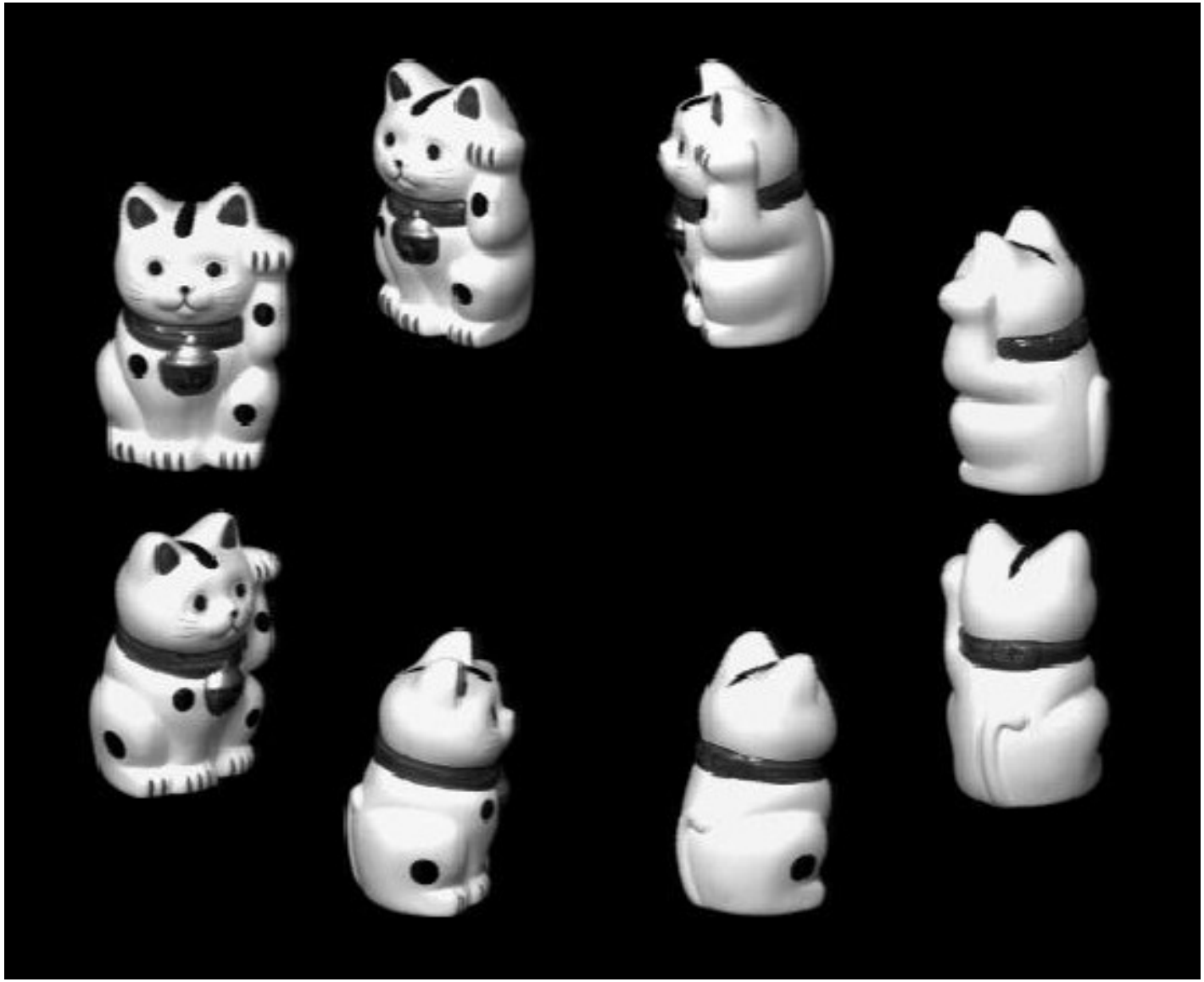
Noise



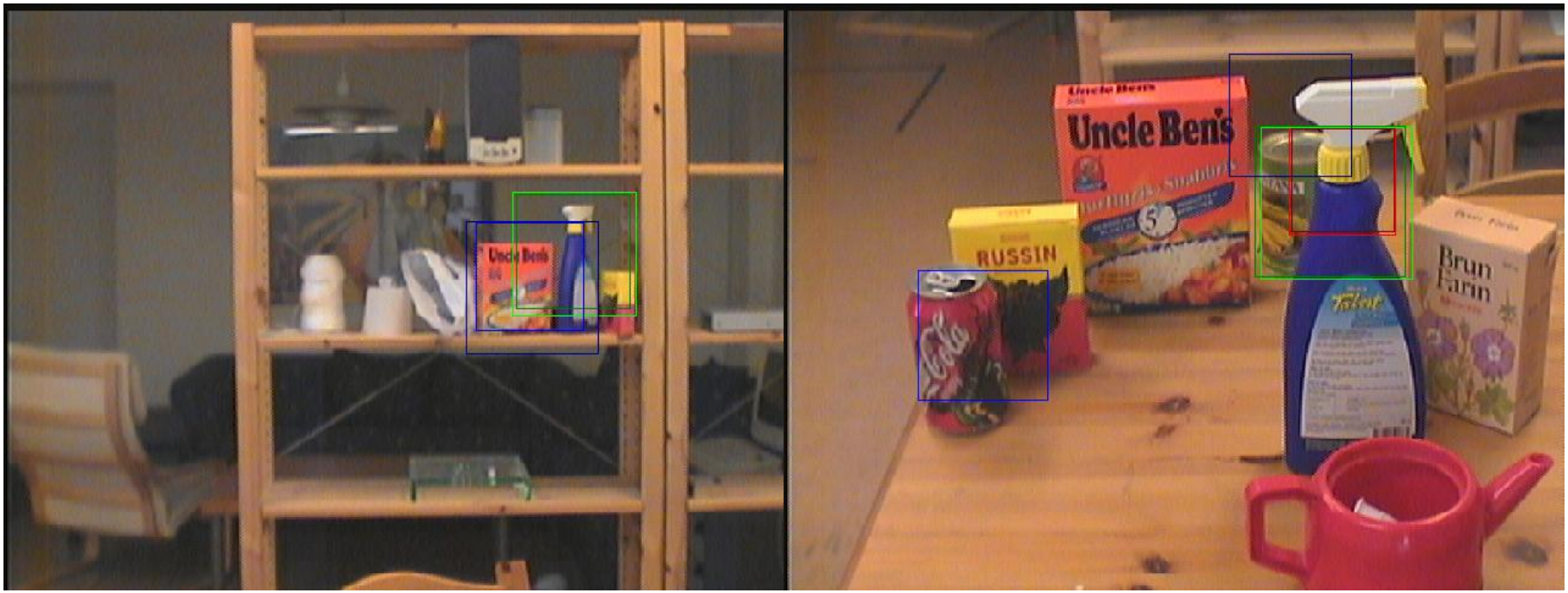
Occlusion



View-point Invariance



Occlusion? Noise?



Modeling occlusions and noise can be very difficult in practice.

Different methods

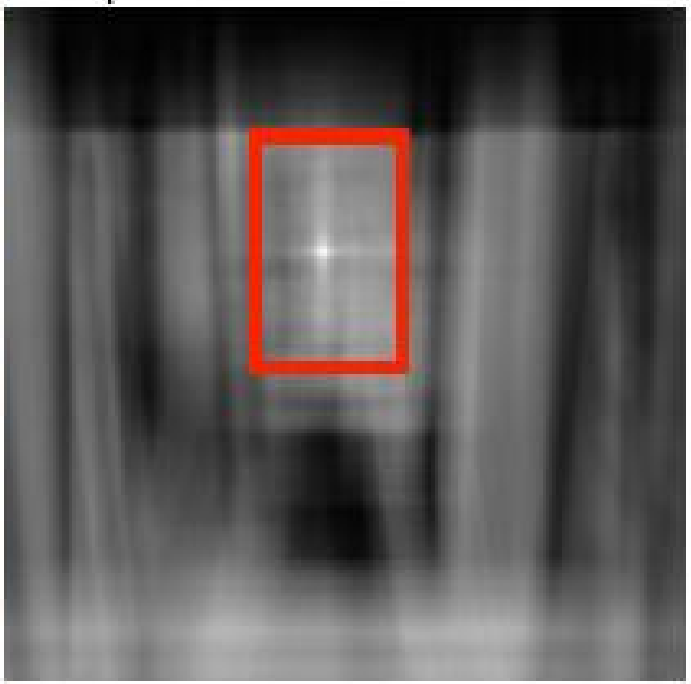
- Template based (dense)
  - Store some “image” of object (characters, faces).
  - Relative positions very important.
  - Normalized (lighting, scales) for invariance.
- Feature based (sparse)
  - Store sets of characteristic features (corners, contours).
  - Relative positions flexible.
  - Only keeps information that can be made invariant.
- Statistics based (usually dense)
  - Store histograms of image data (edges, colours).
  - Positions are disregarded (textures).
  - Each kind of data matched invariantly.

Template matching

This is a chair

Find the chair in this image

Output of normalized correlation



Template matching

Pretty much garbage. Template matching is not going to work.

A statistical method: Color Histograms

- Assume (r_R, r_G, r_B) is the object color and (R, G, B) is the image color. Then if we only have white illumination, there is only one unknown parameter.

$$(R, G, B) = k(r_R, r_G, r_B)$$

- Through a division k can be eliminated.

$$(r, g) = (R/B, G/B) = (r_R/r_B, r_G/r_B)$$

- Idea: Collect (r, g) for every pixel of a model image and create an object representation as a 2D histogram H_m.

Histogram Intersection

- How to compare model histogram H_m to a histogram H_t from a test image? Assuming normalized histograms, i.e. $\sum_k H^k = 1$.
- Typical correlation is not robust due to background clutter.

$$C = \sum_k H_m^k H_t^k$$

- Histogram Intersection is more robust in practice.

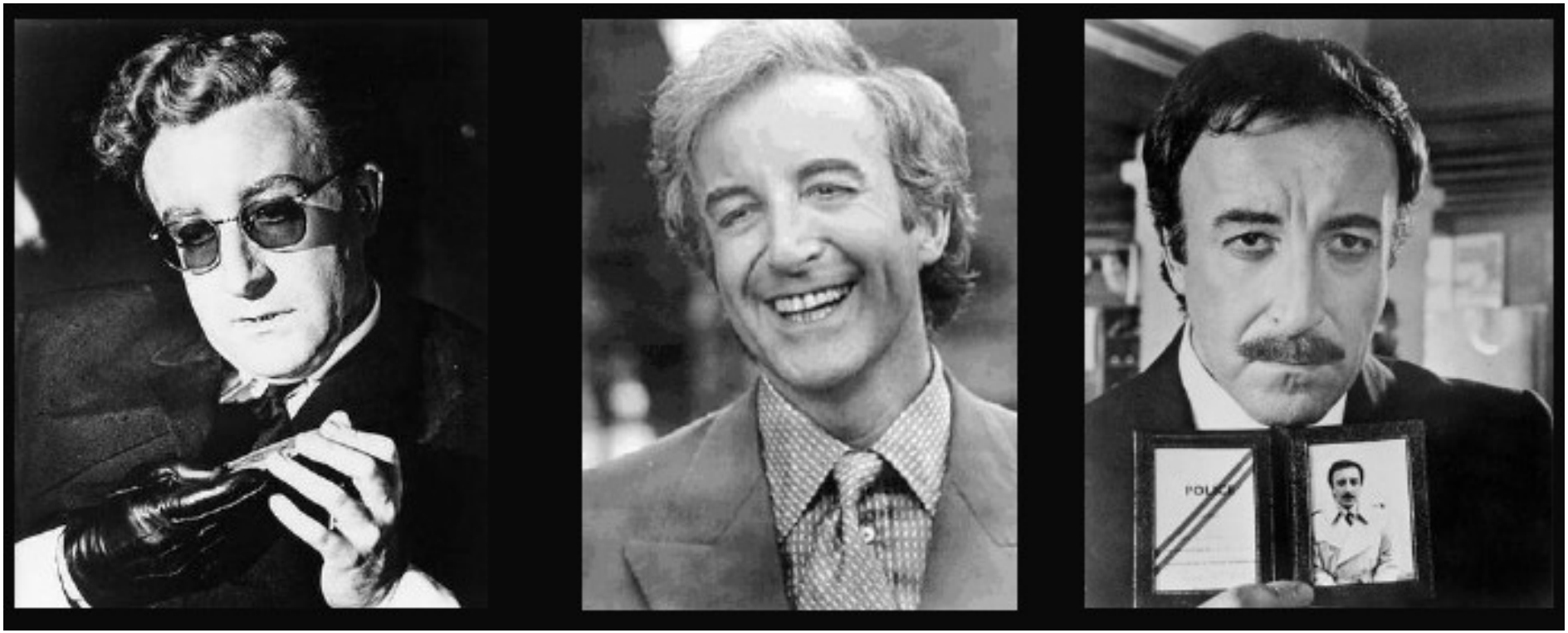
$$C = \sum_k \min(H_m^k, H_t^k)$$

Feature based methods

Most methods contain:

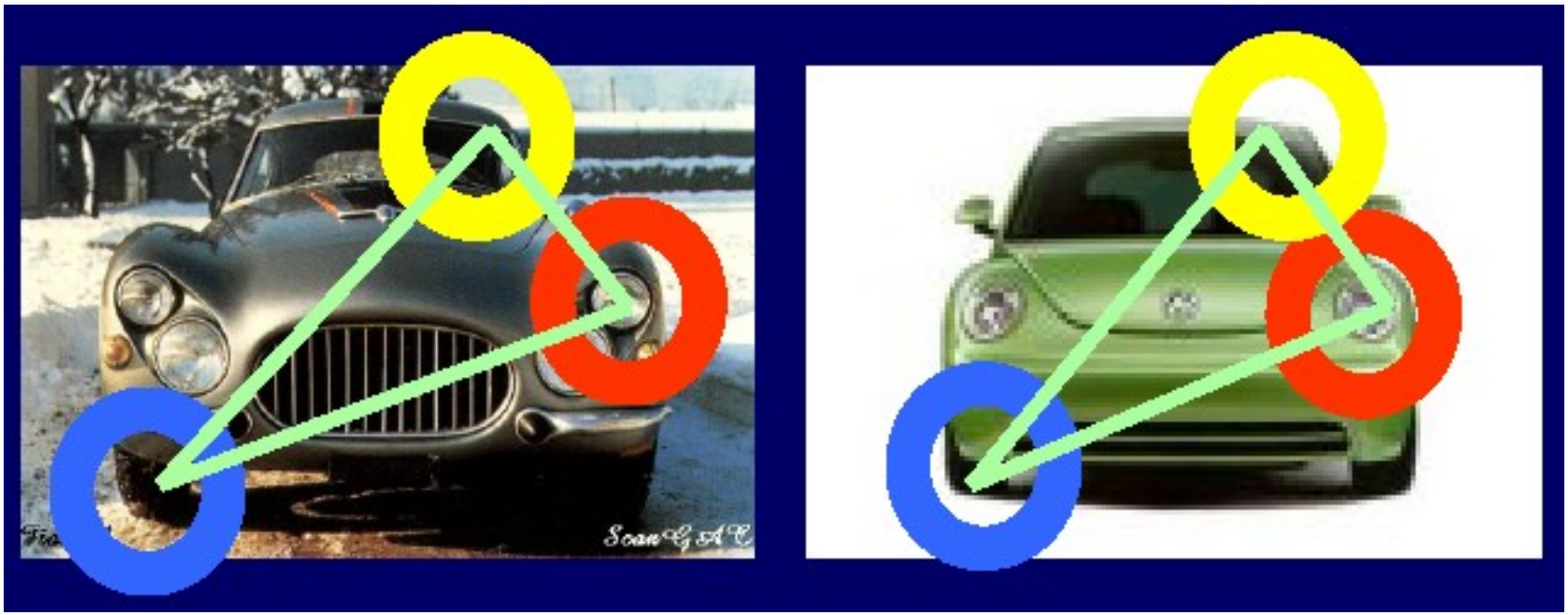
- Some feature detector
  - High curvature points (corners)
  - Scale-space blobs
  - Contours
- Some invariant descriptor(s)
  - PCA, moments
  - Templates
  - Statistics
- Some way of combining features
  - Voting or similar combination measure
  - Often using relative position statistics

Presence / Absence of features



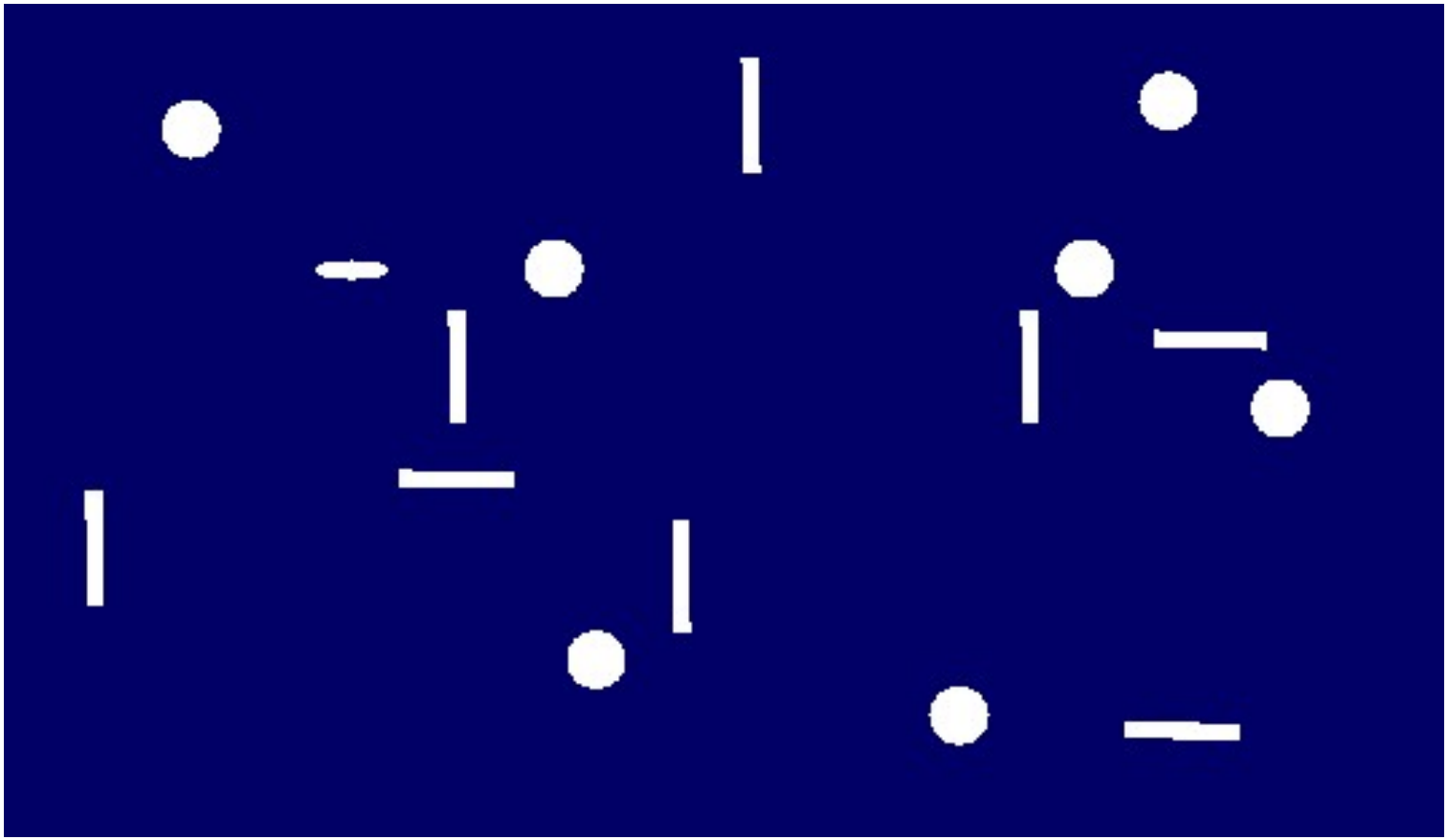
What kind of features exist in most training examples?

Part similarity



Each part might look like many other parts, but combined they might strengthen a hypothesis.

Importance of feature grouping



Can you see the face?

Corner detection revisited

- Corner features are frequently used for recognition.
- Second Moment matrix:

$$M(\mathbf{x}; s, t) = g(\mathbf{x}; s) * \begin{pmatrix} L_x^2(\mathbf{x}; t) & L_x L_y(\mathbf{x}; t) \\ L_y L_x(\mathbf{x}; t) & L_y^2(\mathbf{x}; t) \end{pmatrix}$$

- Local measure of shape
  - Type 2: Textured areas ($\lambda_{min} \gg 0, \lambda_{max} \gg 0$)
  - Type 1: Line segment ($\lambda_{min} \approx 0, \lambda_{max} \gg 0$)
  - Type 0: Texture-less ($\lambda_{min} \approx 0, \lambda_{max} \approx 0$)

Second Moment matrix

Only Type 2 features have accurate positions in 2D.

Type 0 Type 1 Type 2

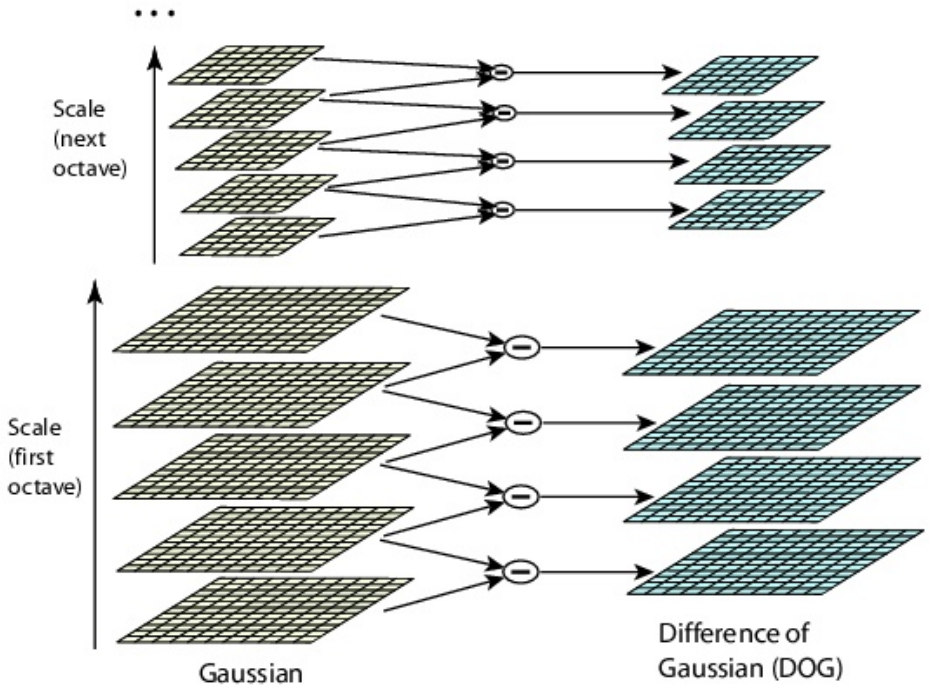
Corner features (Harris & Stephens '88)

- Quick approach of finding Type 2 features.
- Most popular feature detector for many years.
- Features detected a local maxima of

$$\begin{aligned} C &= \det(M) - K \operatorname{trace}^2(M) \\ &= \lambda_{max}\lambda_{min} - K\,(\lambda_{max} + \lambda_{min})^2 \end{aligned}$$

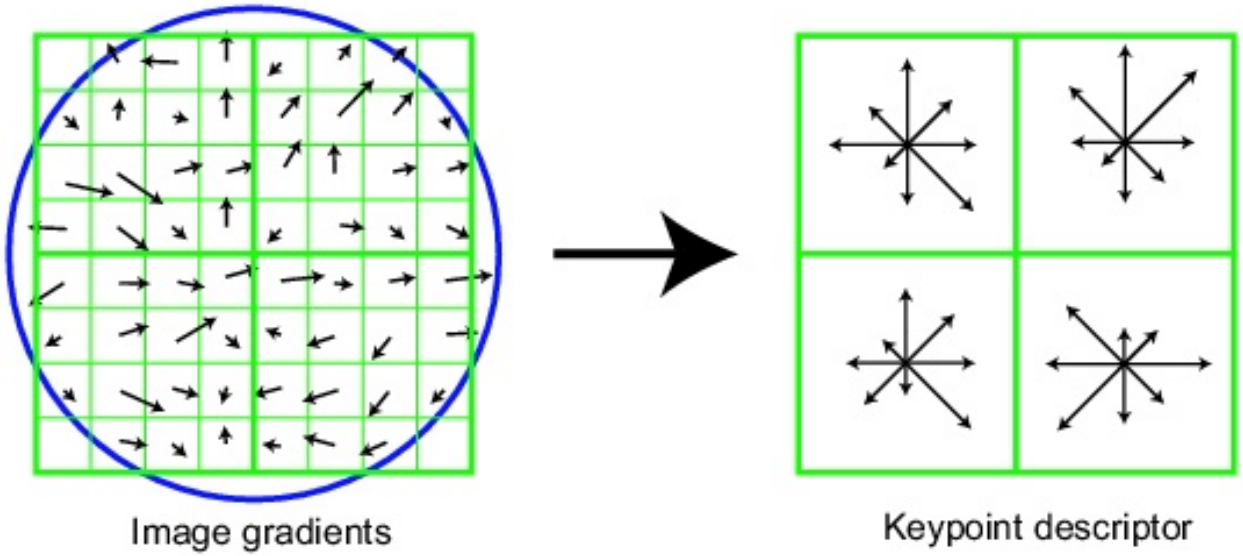
  if C is above a predefined threshold C_0.
- Typically $K \approx 0.05$ and C_0 depends on image quality.

SIFT features (detector) [Lowe '99, Lindeberg '94]



- Detected as peaks in Differences of Gaussians.
  - Approximation of Laplacian (blob detector).
- Multiple scales for scale invariance.

SIFT features (descriptor)

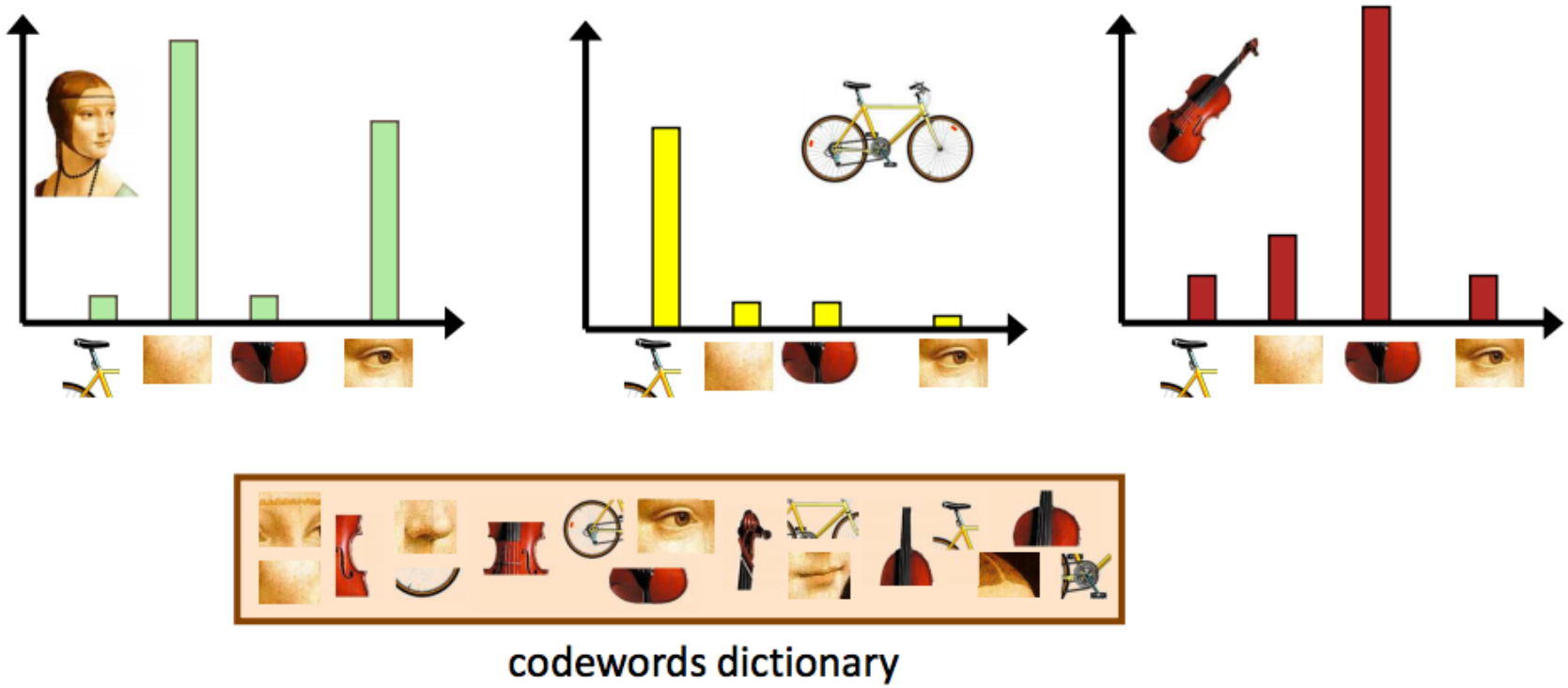


- Descriptors consists of local gradient direction histograms.
- Dominating direction as reference $\Rightarrow$ rotation invariance.
- Window size from scale-space detector $\Rightarrow$ scale invariance.
- Normalize feature vector v to $|v| = 1 \Rightarrow$ luminance invariance.

SIFT features (matching)

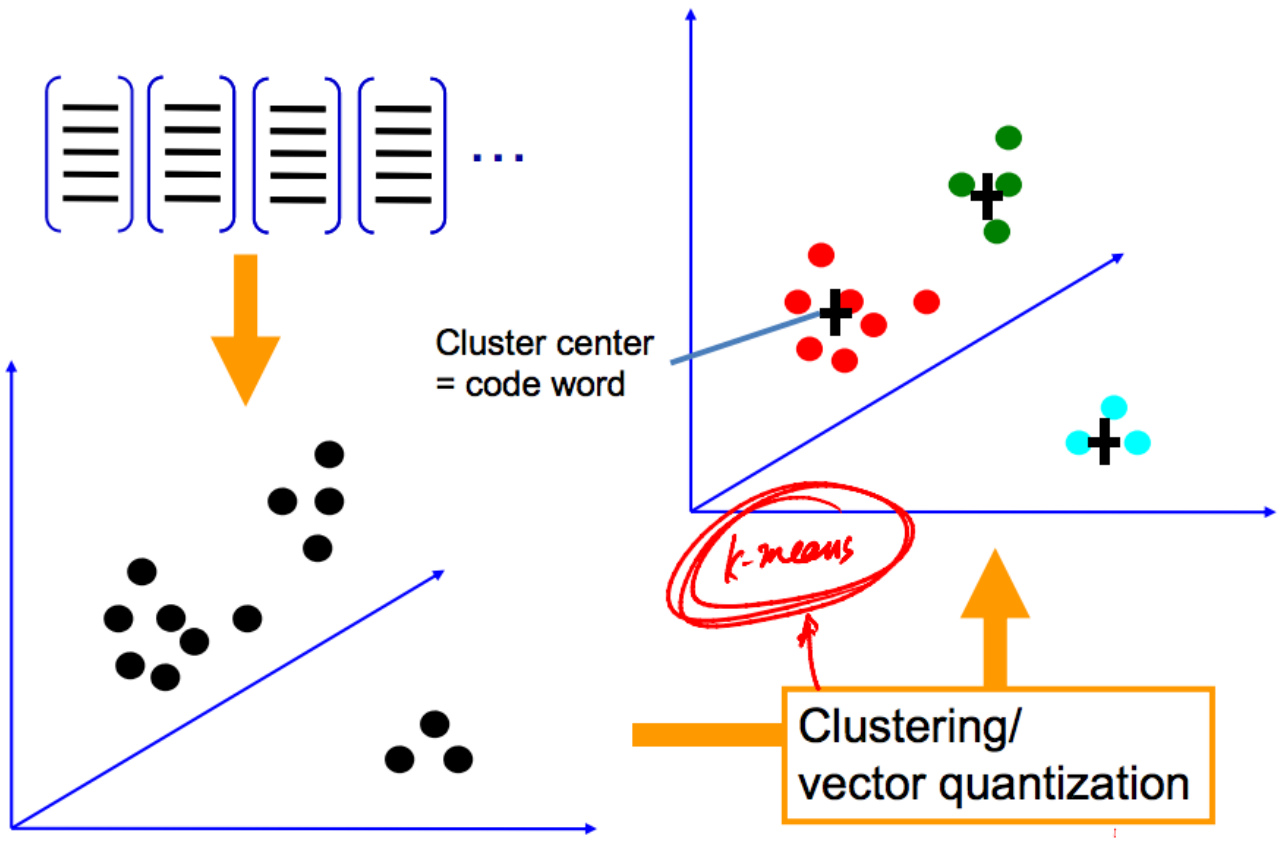
- Match features between image and model.
- Fit a model to remove mismatches (e.g. homography).
- Is there a frog or locomotive in the scene?
  - If enough features are matched the question is Yes.

Bag of Words (BoW): How to go to millions of images?



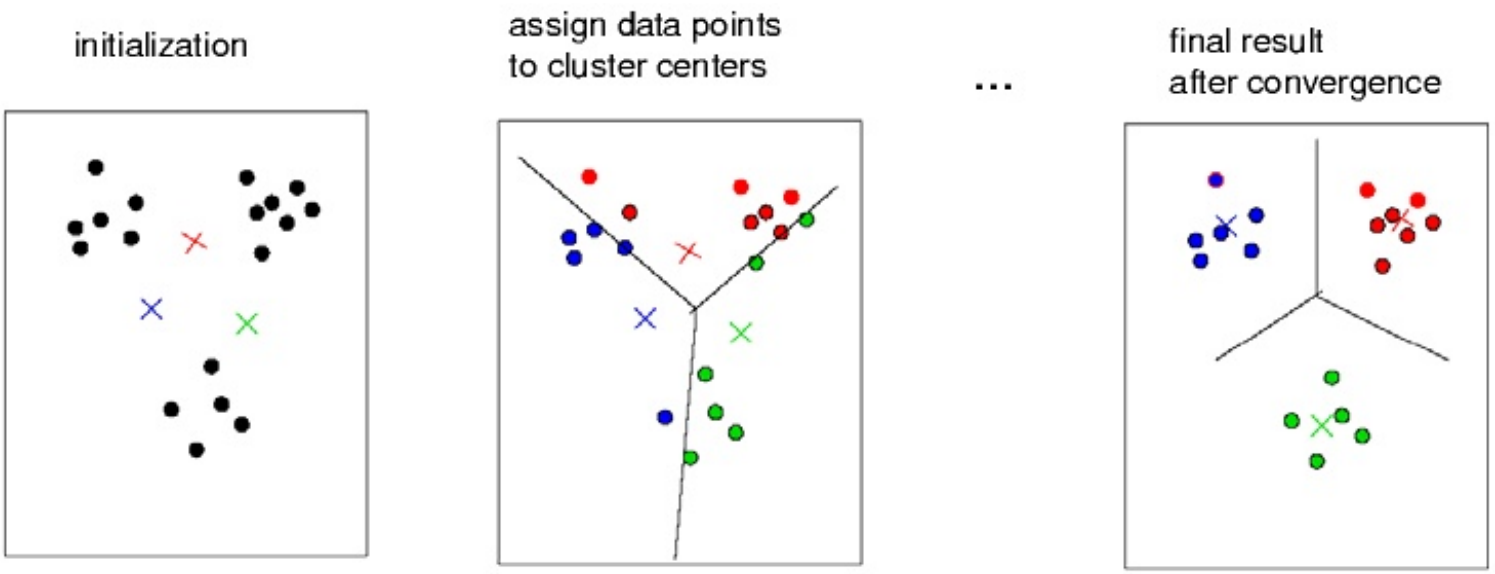
- Idea: represent images as a histograms of features (often SIFT).
- Google: a document is represented as a histogram of words.

Bag of Words (BoW): Codebook creation



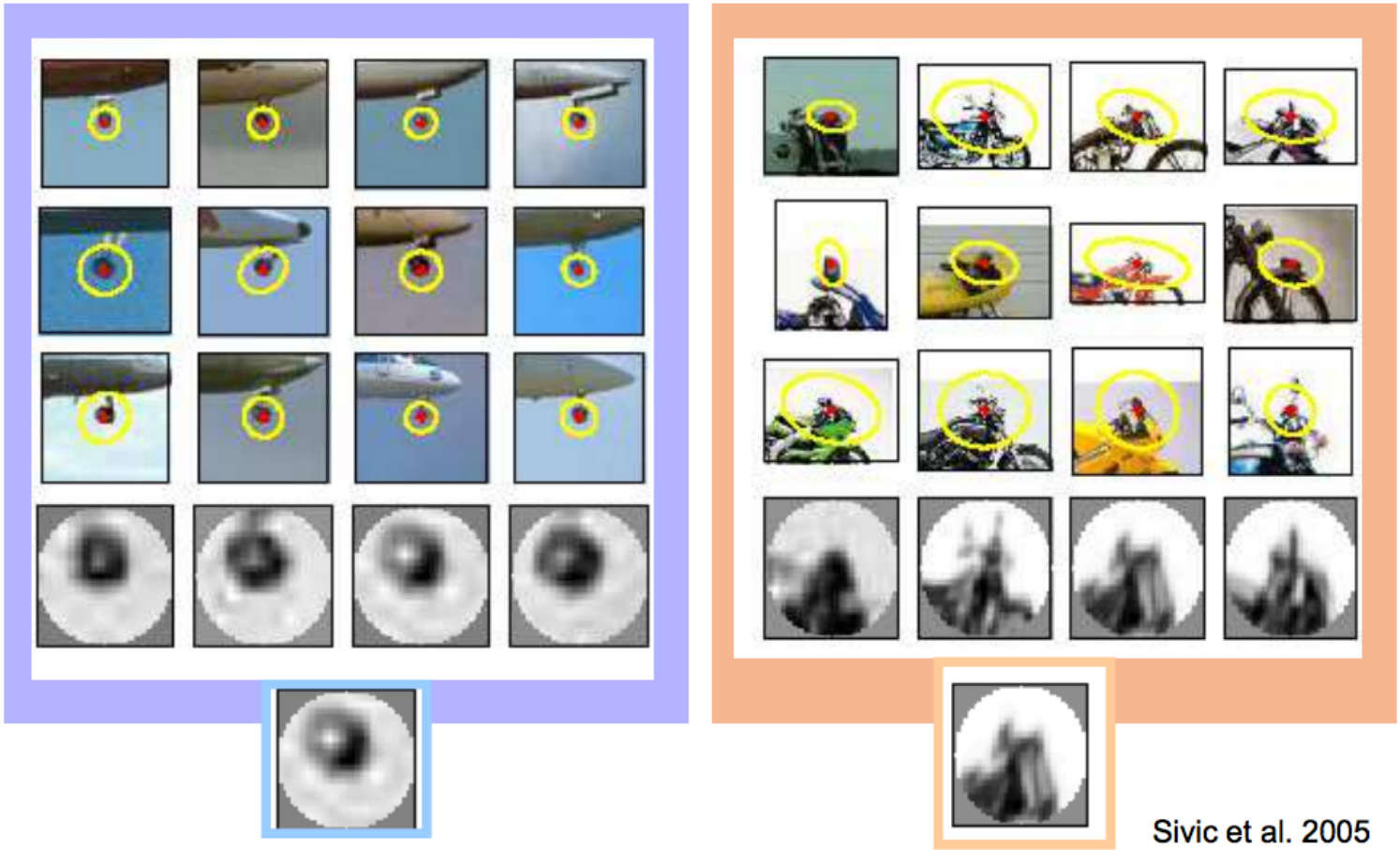
- Goal: Group (SIFT) features into clusters in feature space.
- K-means clustering algorithm is often used in practice:

K-means clustering (example)



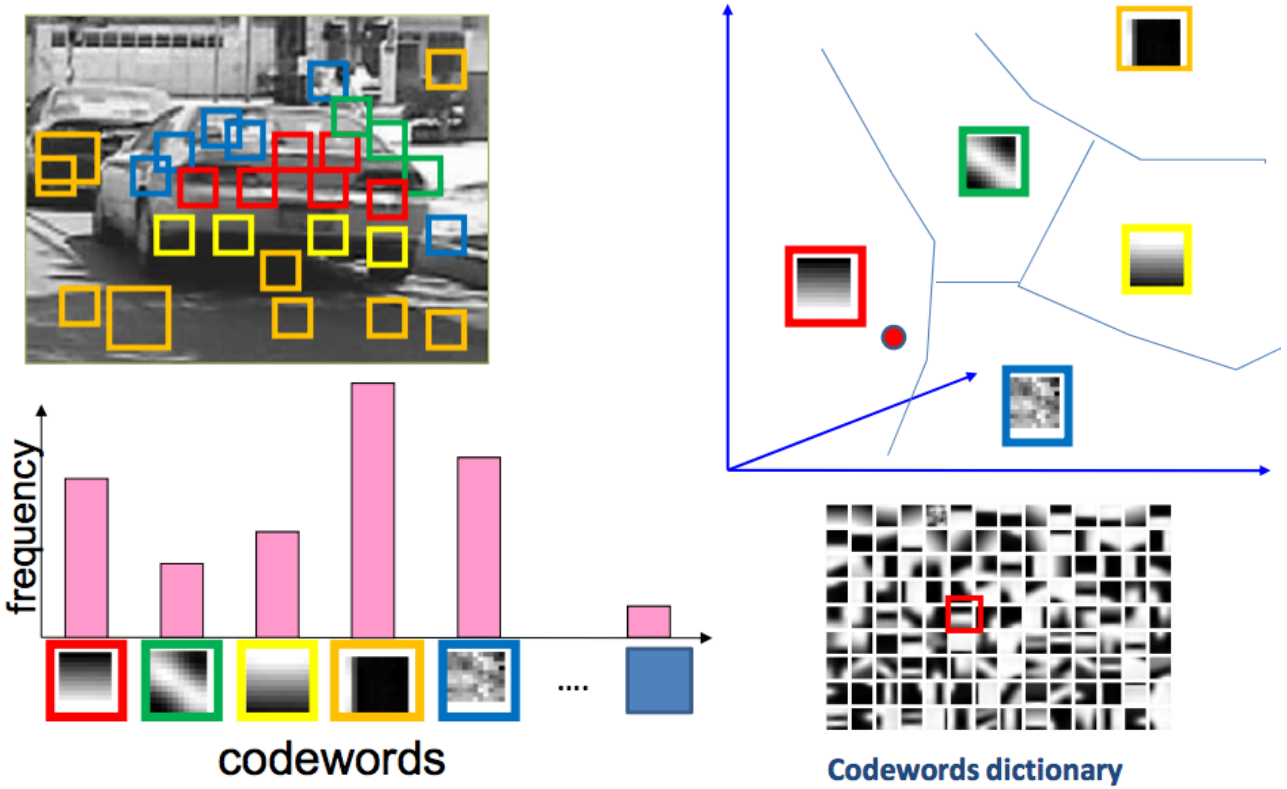
- Problem: Some clusters may have very few features.
- Common solution: Remove clusters that are too small and randomly add new ones.

Codeword examples



Similar features are matched to the same codeword.

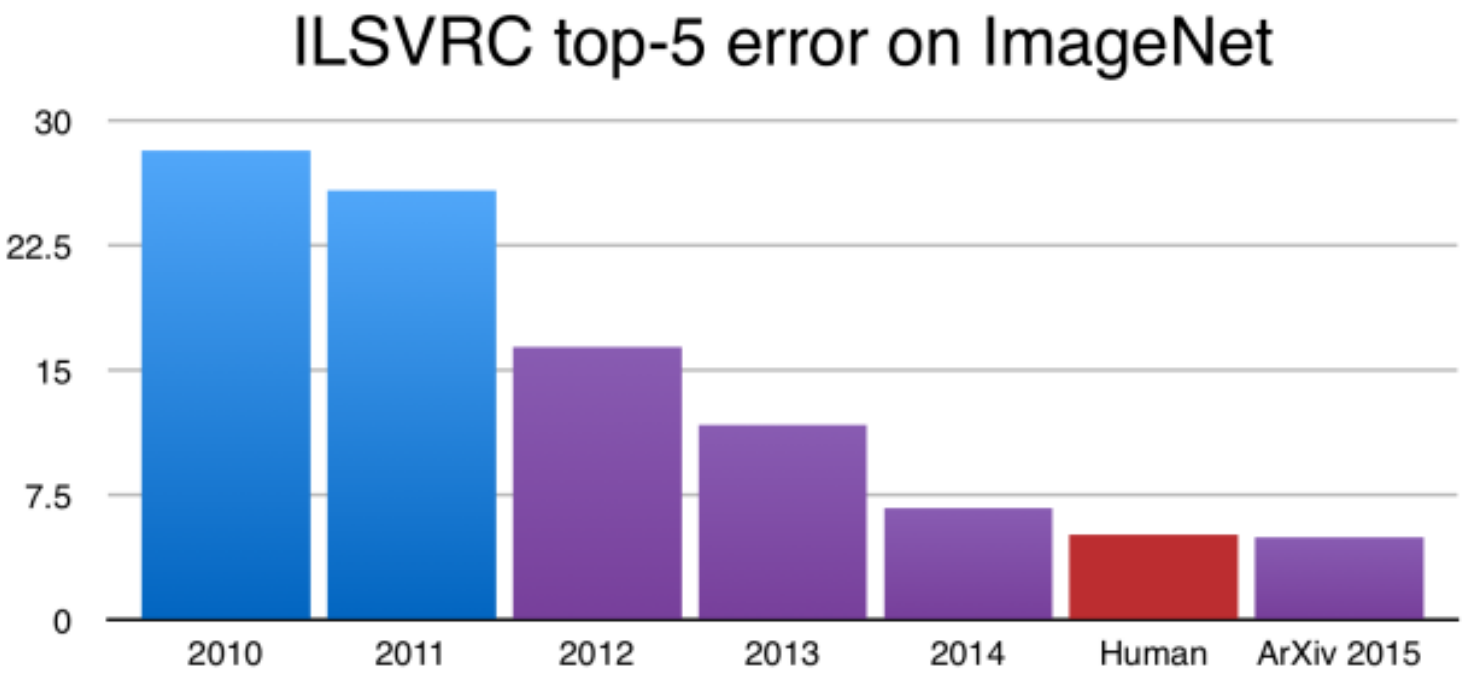
Bag of Words (BoW): Summary of steps



- Step 1: Feature extraction (e.g. SIFT)
- Step 2: Clustering in feature space (e.g. K-means)
- Step 3: Collect histograms of feature numbers
- Step 4: Classify histograms into different object classes

Deep Networks for object recognition

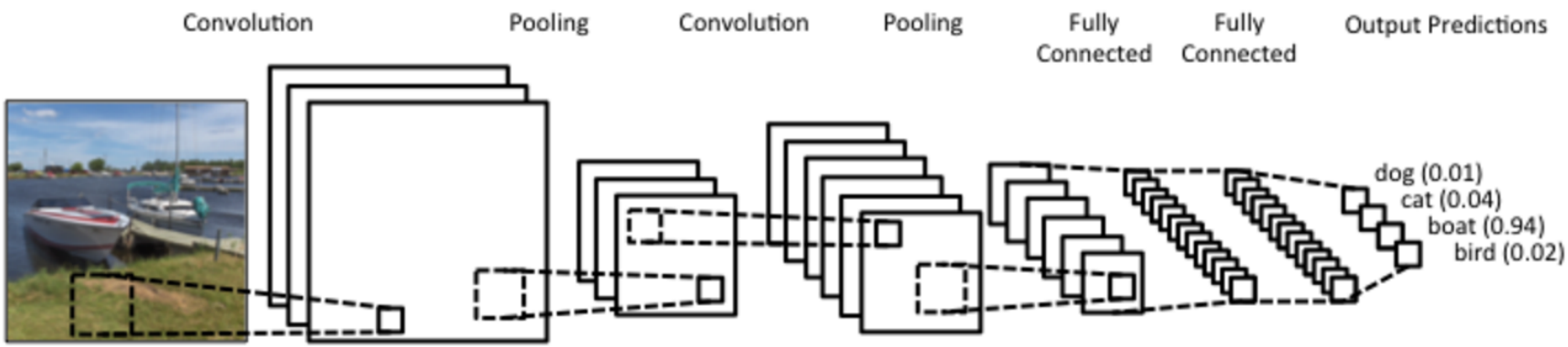
- Neural networks were long forgotten in computer vision.
- Recently, deep neural networks have become state-of-the-art.
- Superior on most challenging benchmarks (1K+ classes)



Traditional methods (blue) vs Deep Networks (violett)

Convolutional Neural Networks

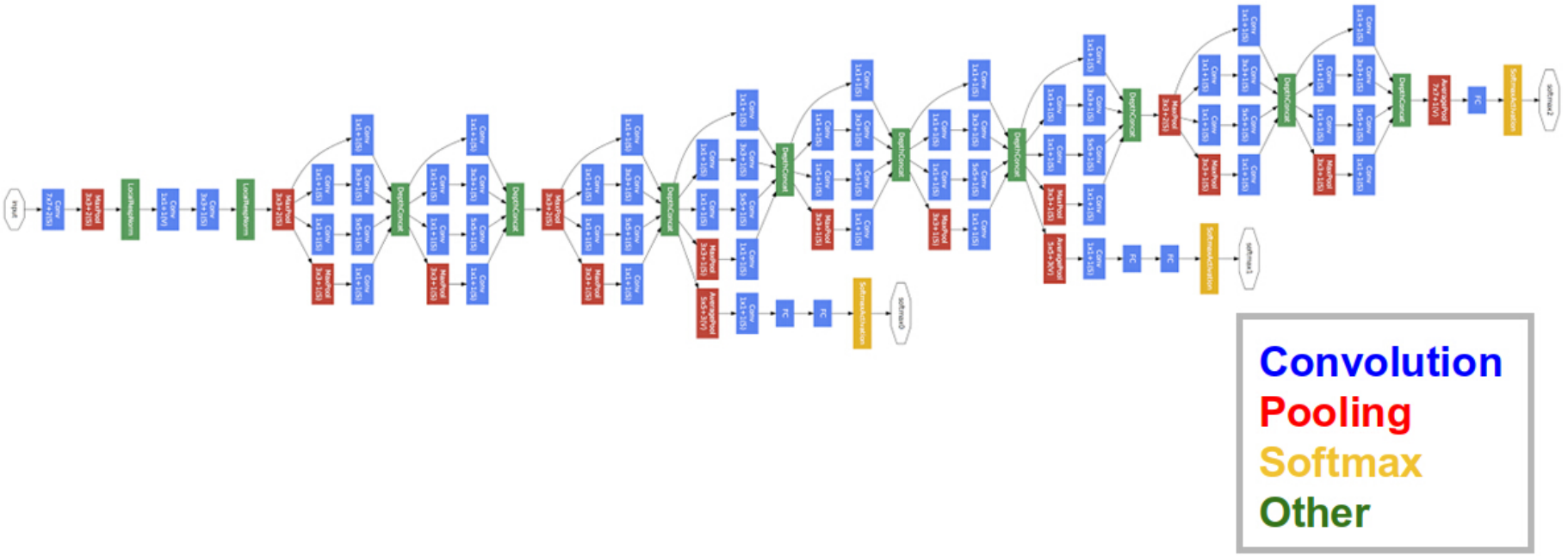
- Idea: have a very simple structure, but learn everything.
- Each layer includes four steps:
  1. Convolutions (normal filtering operations)
  2. Non-linear operator (e.g. set negative values to zero)
  3. Pooling (e.g. find local maximum and subsample)
  4. Normalization of feature maps
- Last layers are fully connected.



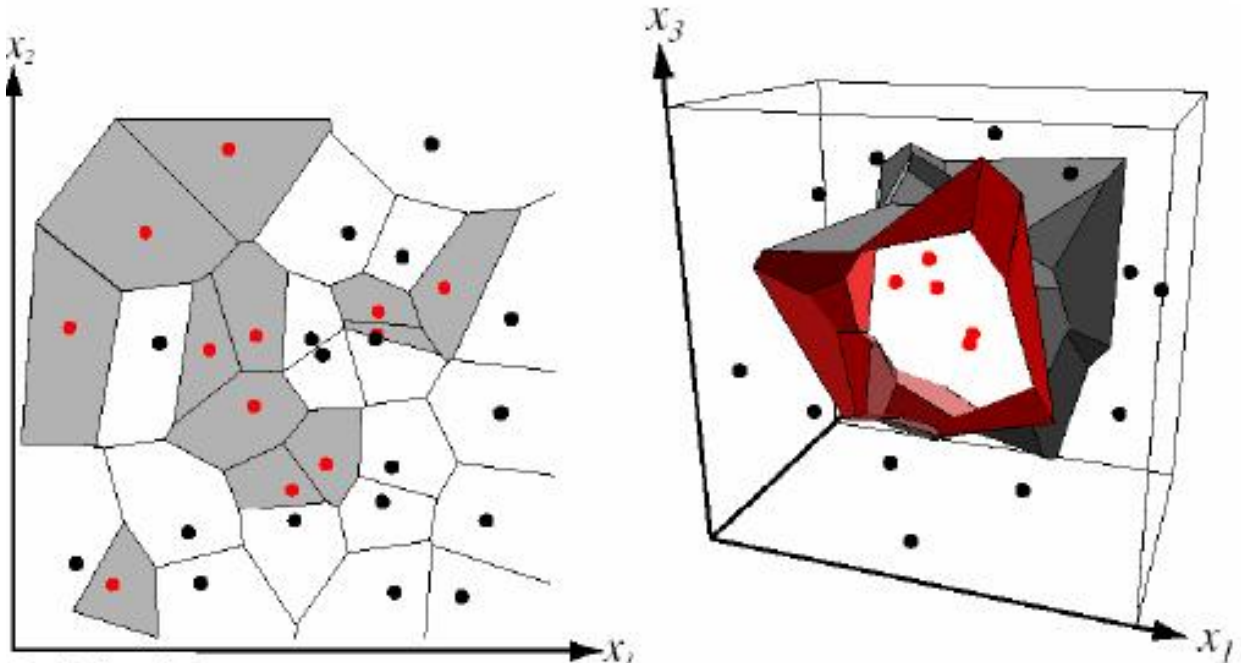
Convolutional Neural Networks

- Pros: simple structure, every good performance
- Cons: hard to analyze, training takes lots of CPU power
- State-of-the-art networks are very large.

Example: GoogleNet



Nearest neighbour classifier



- Straight-forward (model-less) method: given a test image, find the nearest neighbor among all training examples.
- Problem: Number of training examples can be very large and search for nearest neighbour too costly.

Nearest neighbour classification

- Measure feature vectors for a representative selection of images from known classes.
- Given a feature vector, let the classification be the class index of the nearest representative in the scatter diagram.

\+ Works for well separated classes.

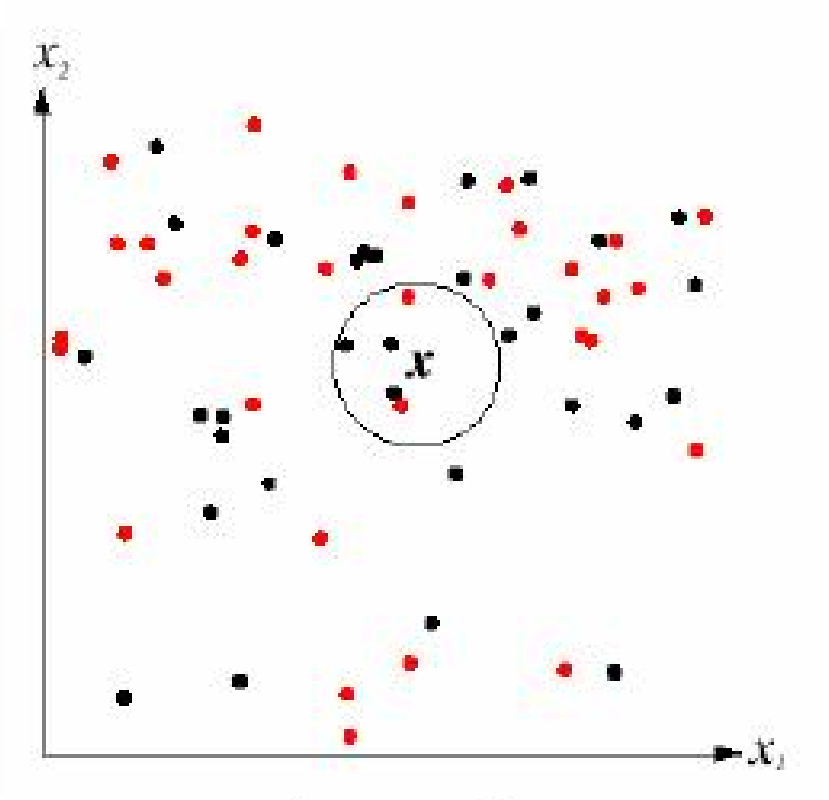
\+ Can represent clusters with complex shape.

\- May require complex computations in higher dimensions.

\- Depends on choice of metrics.

\- Highly sensitive to outliers (no suppression).

K-Nearest neighbours



$$k = 5$$

Several points 'vote' for the final answer.

Statistical classification methods

- Idea: consider the class assignment and feature vectors as stochastic variables. Determine a classification function that minimizes a given measure of the classification error.
- Let z = measurement vector, k = class
- If we know the prior probability for the class, $p(k)$,

  and measurement distribution of class, $p(z \mid k)$,

  we know class probability given measurements, $p(k \mid z)$.

Bayesian classification

$$p(z,k) = p(z \mid k)\, p(k) = p(k \mid z)\, p(z) \Rightarrow$$

Bayes' formula:

$$p(k \mid z) = \frac{p(z \mid k)\, p(k)}{p(z)} = \frac{p(z \mid k)\, p(k)}{\sum_i p(z \mid k_i)\, p(k_i)}$$

Bayesian classification (continue)

Choose the class k that maximizes $p(k \mid z)$.

- Minimizes the probability of wrong classification, IF statistical model is correct.
- Given most probable class, if prior probabilities $p(k)$ are known.
- Common assumption: All $p(k)$ equal $\Rightarrow$ Maximum likelihood

Example: Plain thresholding

$$\text{Let} \begin{cases} k = A & \text{if} \quad z \leq t \\ k = B & \text{if} \quad z > t \end{cases}$$

Probability of wrong classification:

$$\begin{aligned} p_{err}(t) &= p(k=A) \cdot p(z > t | k = A) + p(k=B) \cdot p(z \leq t | k = B) \\ &= \{\text{with} \quad P(t) = p(z \leq t)\} \\ &= p(k=A)(1 - P(t|k=A)) + p(k=B)P(t|k=B) \end{aligned}$$

Example: Two Gaussian distributions

Assume that the probability density functions are given by

- Gaussian distributions with same variance and different means

$$f(z|k=A) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(z-\mu)^2/(2\sigma^2)}$$

$$f(z|k=B) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(z-\nu)^2/(2\sigma^2)}$$

  and that the prior probabilities are

$$p(k=A) \quad \text{and} \quad p(k=B)$$

- Probability of wrong classification

$$p_{err}(t) = (1 - F(t|k=A))p(k=A) + F(t|k=B)p(k=B)$$

Example: Two Gaussian distributions (continue)

- Minimize p_{err} with respect to t:

$$\begin{aligned}\frac{dp_{err}}{dt} &= -f(t|k=A)p(k=A) + f(t|k=B)p(k=B) \\ &= -p_A \frac{1}{\sqrt{2\pi}\sigma} e^{-(t-\mu)^2/(2\sigma^2)} + p_B \frac{1}{\sqrt{2\pi}\sigma} e^{-(t-\nu)^2/(2\sigma^2)}\end{aligned}$$

- Logarithms of both sides

$$\log p_A - \frac{(t-\mu)^2}{2\sigma^2} = \log p_B - \frac{(t-\nu)^2}{2\sigma^2}$$

- Solve for $t \Rightarrow$

$$t = \frac{\mu+\nu}{2} + \frac{\sigma^2}{(\nu-\mu)} \log \frac{p(k=A)}{p(k=B)}$$

Two Gaussian distributions (continue)

- Optimal threshold (if variance is the same)

$$t = \frac{\mu + \nu}{2} + \frac{\sigma^2}{\nu - \mu} \log \frac{p(k = A)}{p(k = B)}$$

- If both classes are equally common:

$$p(k = A) = p(k = B) \Rightarrow t = \frac{\mu + \nu}{2}$$

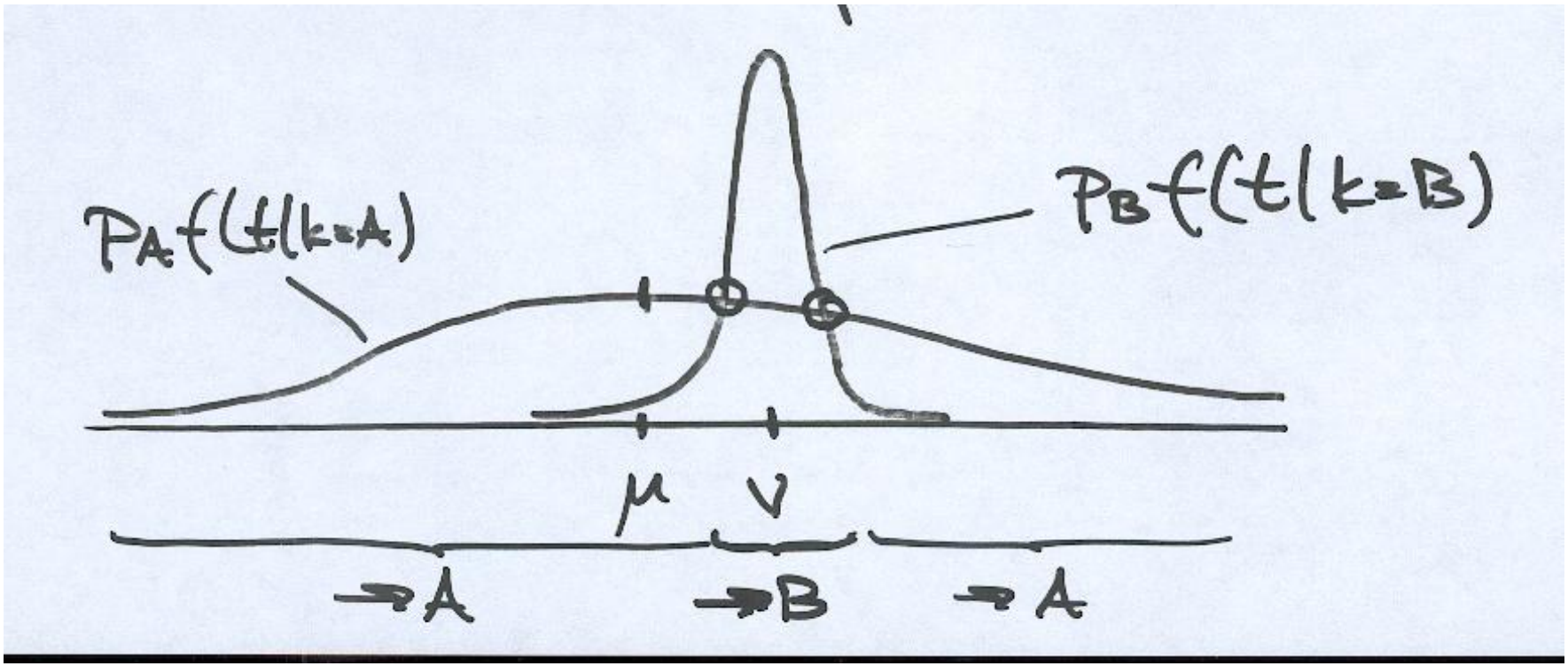
- If they are not equally common:

$$p(k = A) > p(k = B) \Rightarrow \frac{\sigma^2}{\nu - \mu} \log \frac{p(k = A)}{p(k = B)} > 0$$

This is, the optimal threshold is moved away from the more common class.

Two Gaussian distributions (continue)

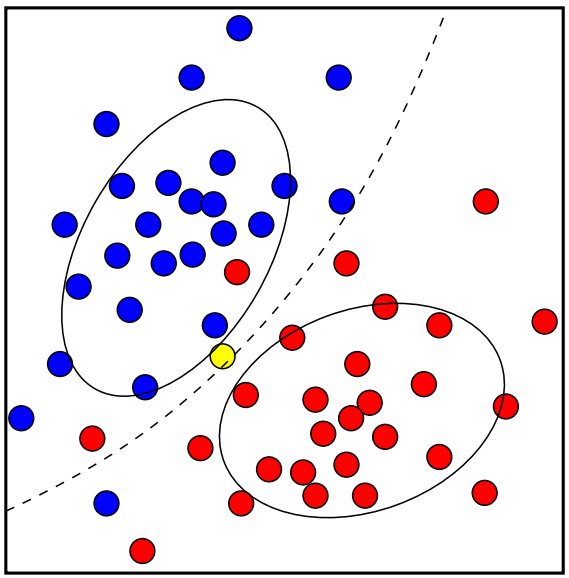
Different variances $\sigma_A^2 \neq \sigma_B^2 \Rightarrow$ two thresholds possible.



Assuming 2D Gaussian distributions

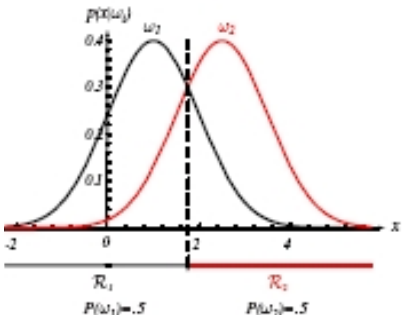
$$p(\mathbf{z} \mid k_i) = \frac{1}{(2\pi)^{D/2} \mid \Sigma_i \mid^{1/2}} \exp\left\{ -\frac{1}{2}(\mathbf{z} - \mu_\mathbf{i})^\top \Sigma_i^{-1} (\mathbf{z} - \mu_\mathbf{i}) \right\}$$

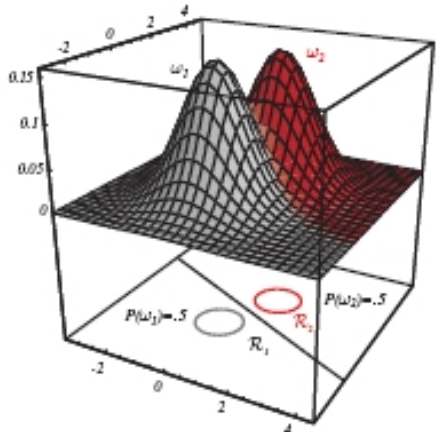
$$p(k_1 \mid \mathbf{z}) = \frac{p(\mathbf{z} \mid k_1)\, p(k_1)}{p(\mathbf{z} \mid k_1)\, p(k_1) + p(\mathbf{z} \mid k_2)\, p(k_2)}$$

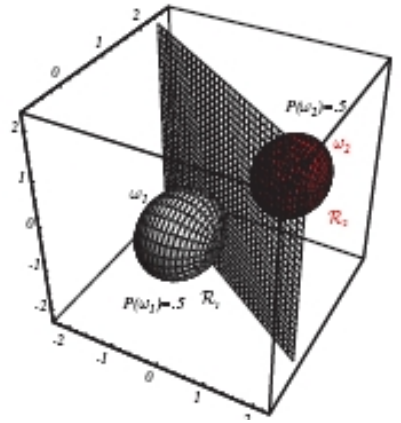


Decision boundary: $p(\mathbf{z} \mid k_1)\, p(k_1) = p(\mathbf{z} \mid k_2)\, p(k_2)$

Two class case

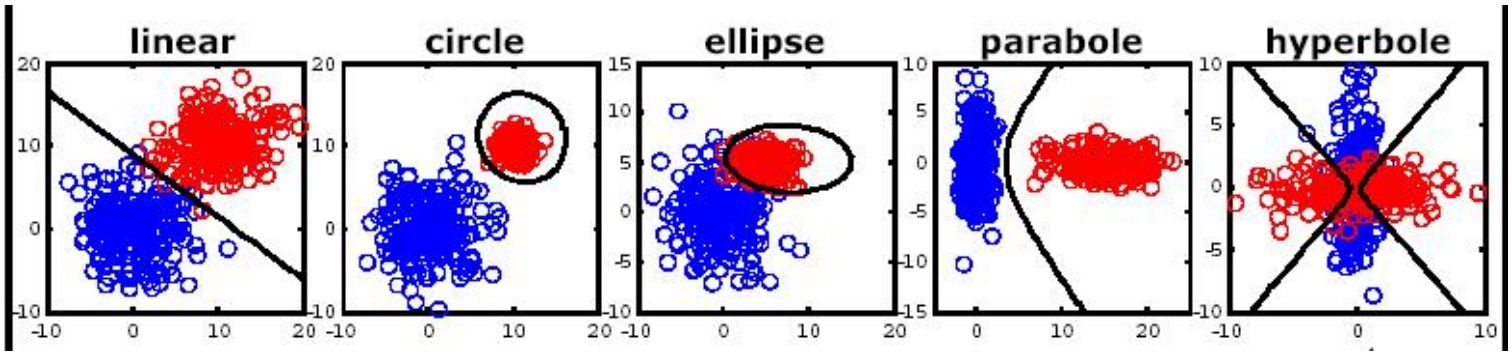


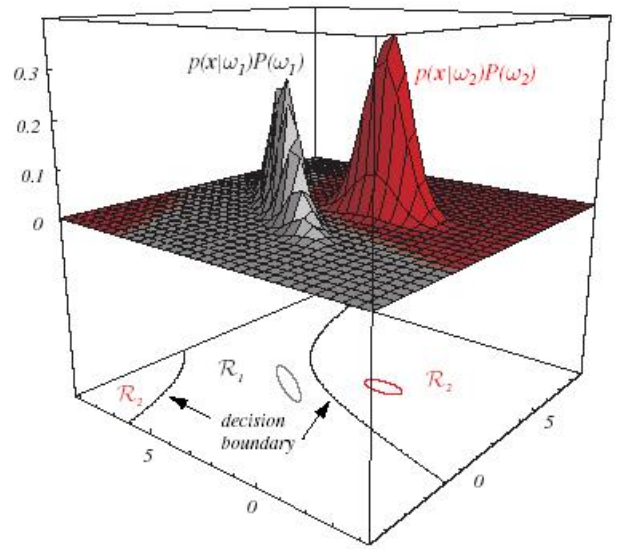




If the covariances matrices are the same, the decision boundary is a hyperplane separating the classes.

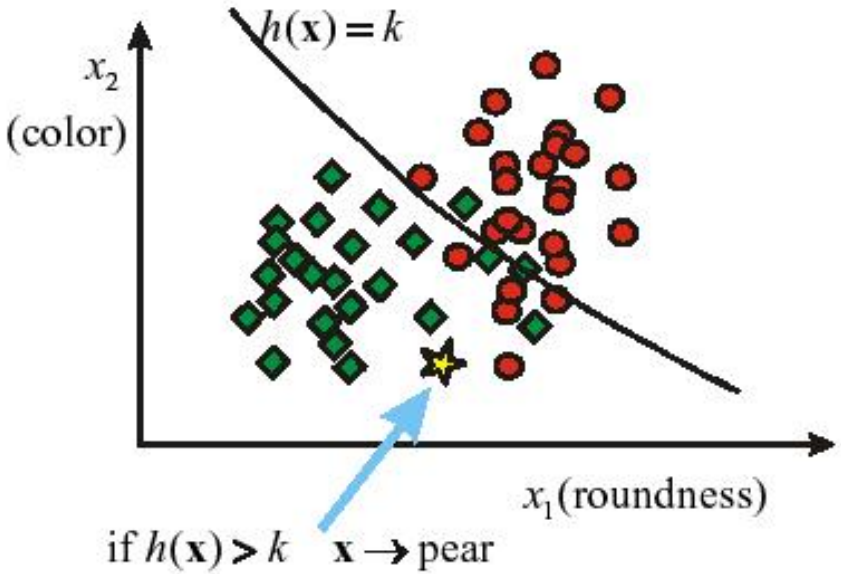
Two class case





If not, the decision boundaries can have different shapes.

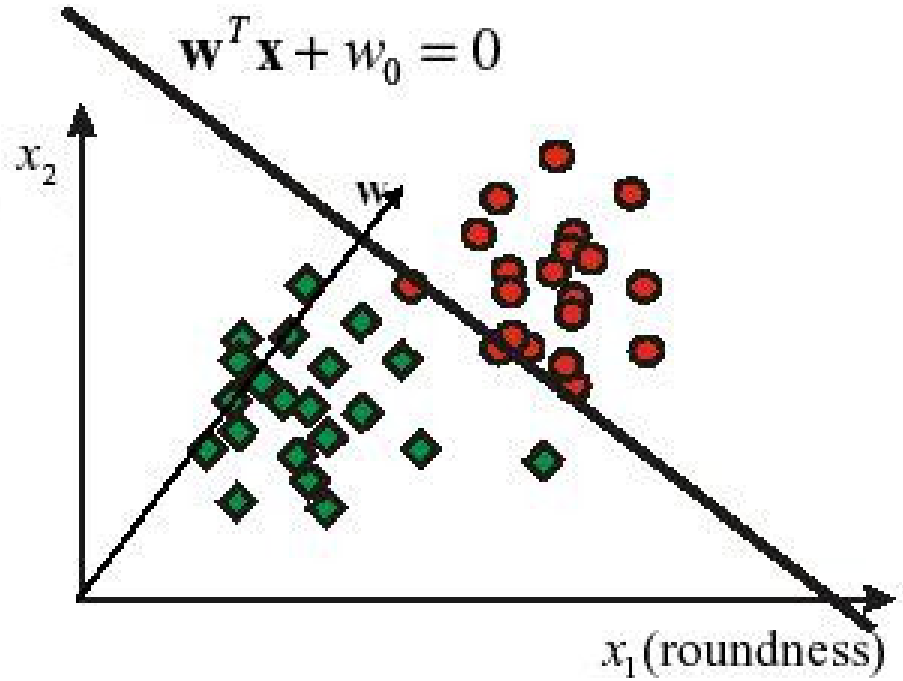
Discriminant functions



Instead of modelling the distributions and then find a decision function, find a decision function and its boundary directly.

1. Choose class of decision function.
2. Estimate parameters of this function from training data.
3. Classify a new point based on the decision function.

Linear Discriminant Functions



- Problem: Finding the best $\mathbf{v} = (\mathbf{w}, w_o)$ given training examples.
- Most modern machine learning methods (Boosting, SVM, etc) use combinations of many linear discriminant functions.

Summary of good questions

- What is the difference between recognition and classification?
- What makes a good feature space for recognition?
- What kind of invariances do you often want in recognition?
- What classes of recognition methods exist and what are their differences?
- What does a typical feature based method consist of?
- What steps does a Bag of Words approach include?
- What characteristics does a nearest neighbour classifier have?
- How do you find a decision boundary for Bayesian classification?

Readings

- Gonzalez & Woods: Chapter 12.1 – 12.2.2
- Szeliski: Chapter 14