Bag-of-Words models

Lectures 16 and 17

Slides from: S. Lazebnik, A. Torralba, L. Fei-Fei, D. Lowe, C. Szurka

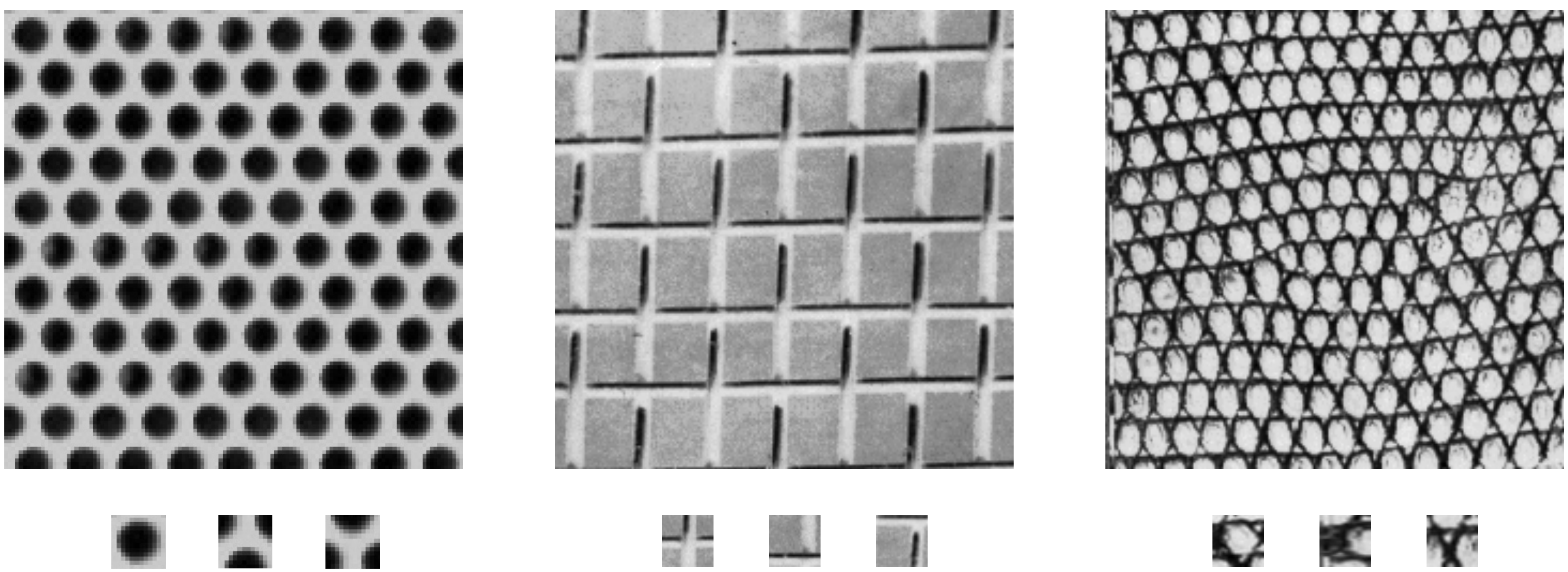
Bag-of-features models

Overview: Bag-of-features models

- Origins and motivation
- Image representation
- Discriminative methods
  - Nearest-neighbor classification
  - Support vector machines
- Generative methods
  - Naïve Bayes
  - Probabilistic Latent Semantic Analysis
- Extensions: incorporating spatial information

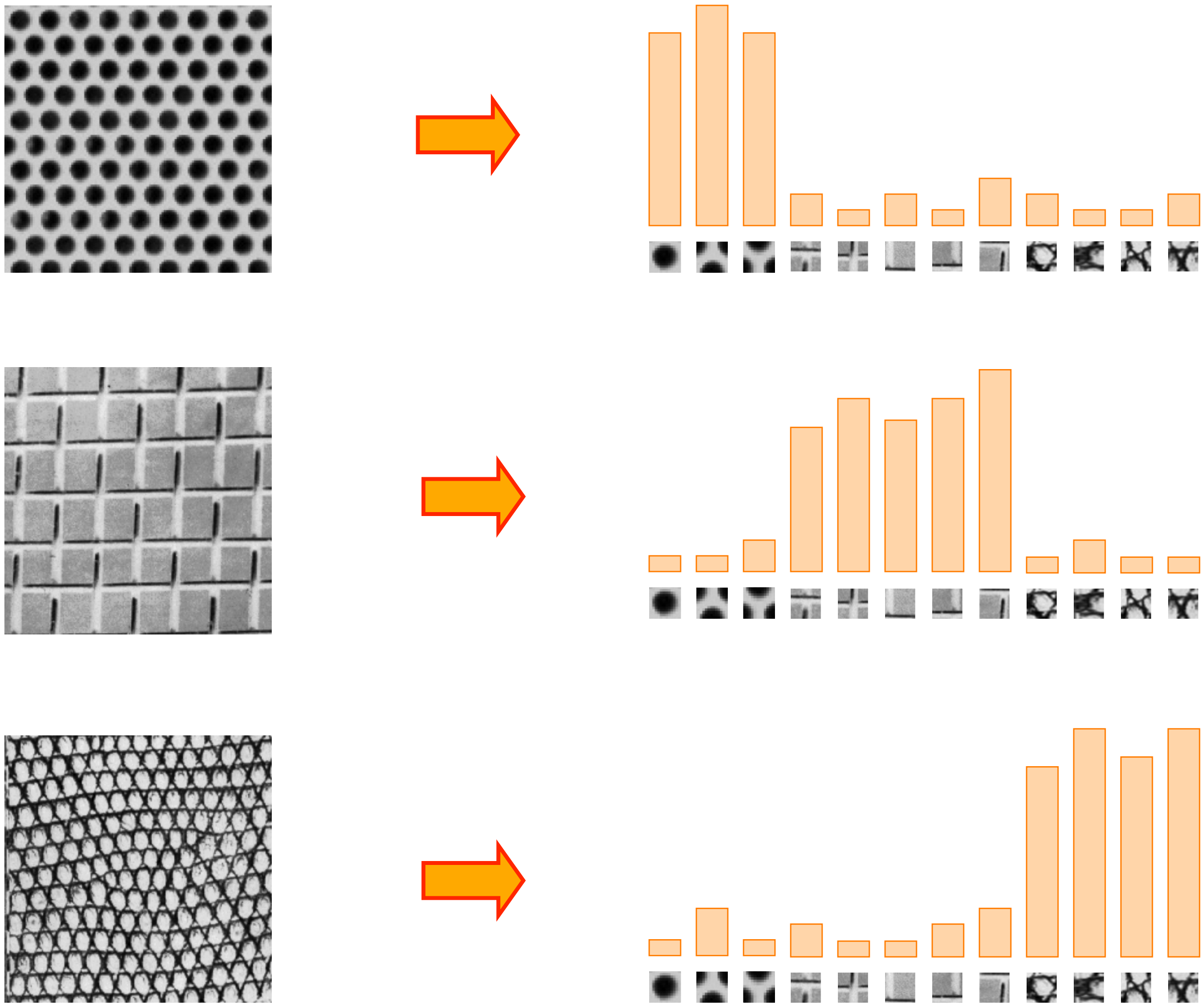
Origin 1: Texture recognition

- Texture is characterized by the repetition of basic elements or *textons*
- For stochastic textures, it is the identity of the textons, not their spatial arrangement, that matters



Julesz, 1981; Cula & Dana, 2001; Leung & Malik 2001; Mori, Belongie & Malik, 2001; Schmid 2001; Varma & Zisserman, 2002, 2003; Lazebnik, Schmid & Ponce, 2003

Origin 1: Texture recognition



Julesz, 1981; Cula & Dana, 2001; Leung & Malik 2001; Mori, Belongie & Malik, 2001; Schmid 2001; Varma & Zisserman, 2002, 2003; Lazebnik, Schmid & Ponce, 2003

Origin 2: Bag-of-words models

- Orderless document representation: frequencies of words from a dictionary Salton & McGill (1983)

Origin 2: Bag-of-words models

- Orderless document representation: frequencies of words from a dictionary Salton & McGill (1983)

Origin 2: Bag-of-words models

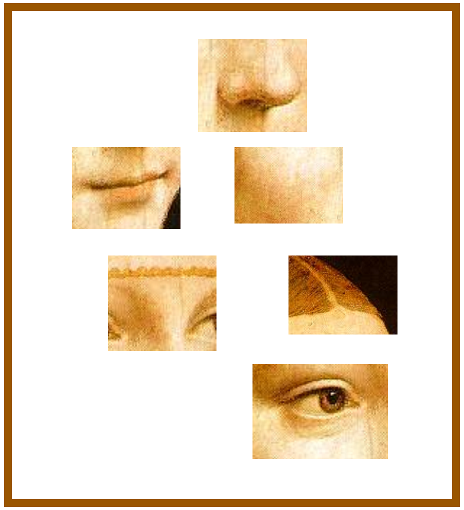
- Orderless document representation: frequencies of words from a dictionary Salton & McGill (1983)

Origin 2: Bag-of-words models

- Orderless document representation: frequencies of words from a dictionary Salton & McGill (1983)

Bags of features for image classification

1. Extract features



Bags of features for image classification

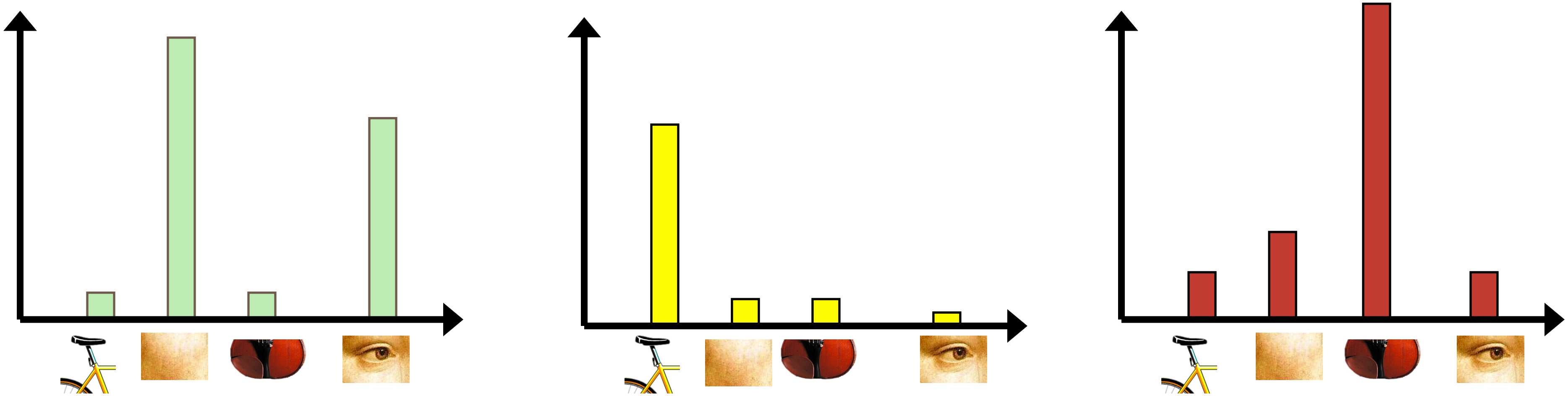
1. Extract features
2. Learn “visual vocabulary”

Bags of features for image classification

1. Extract features
2. Learn “visual vocabulary”
3. Quantize features using visual vocabulary

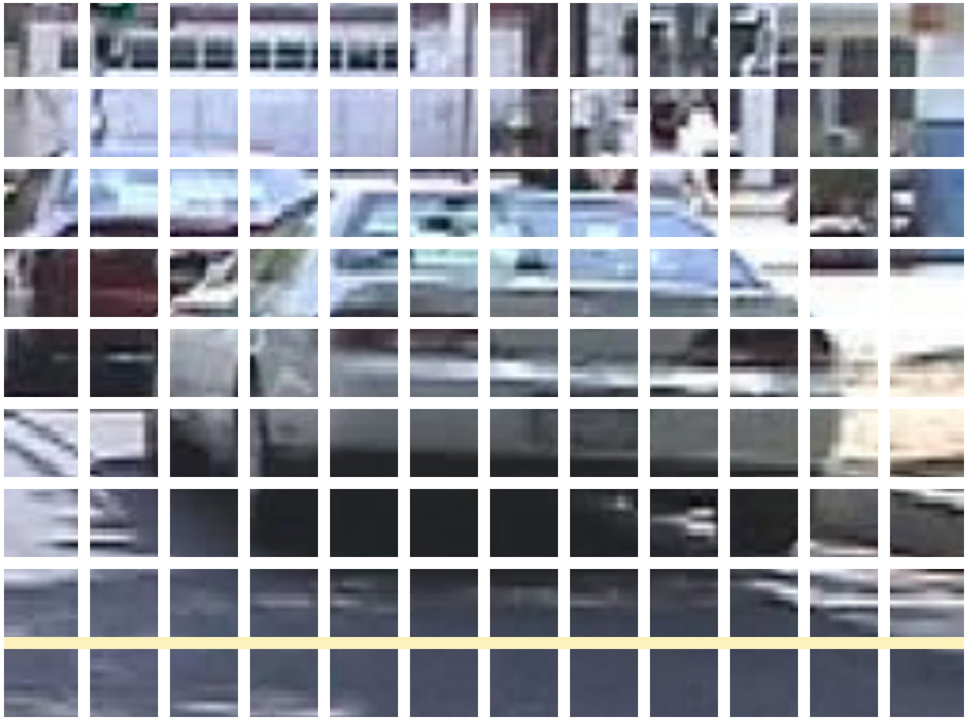
Bags of features for image classification

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



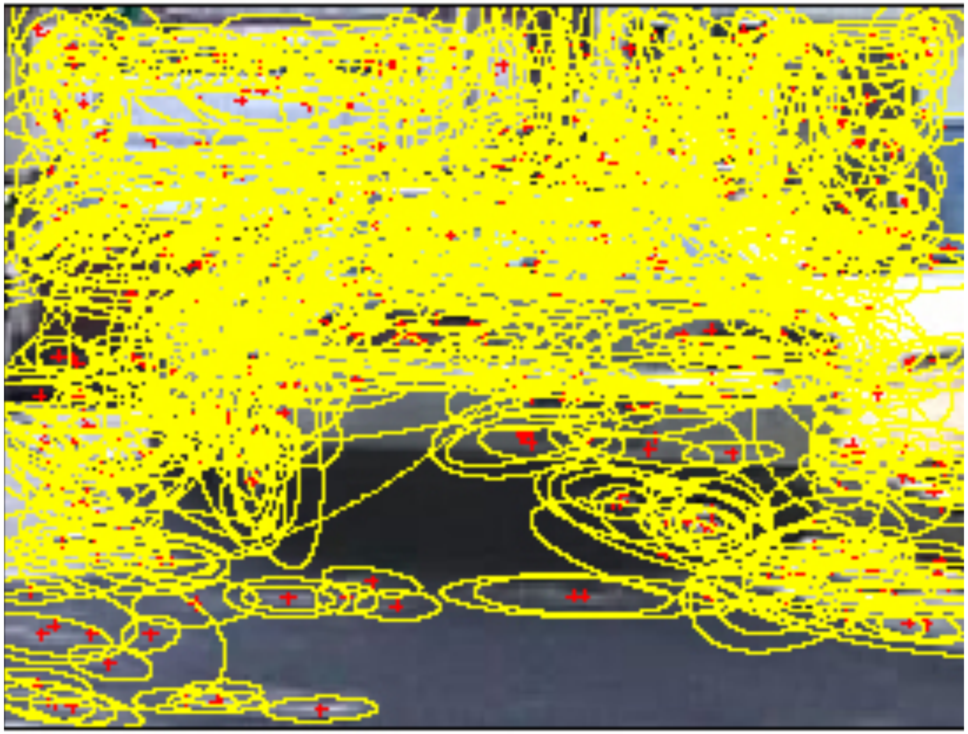
1. Feature extraction

- Regular grid
  - Vogel & Schiele, 2003
  - Fei-Fei & Perona, 2005



1. Feature extraction

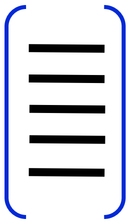
- Regular grid
  - Vogel & Schiele, 2003
  - Fei-Fei & Perona, 2005
- Interest point detector
  - Csurka et al. 2004
  - Fei-Fei & Perona, 2005
  - Sivic et al. 2005



1. Feature extraction

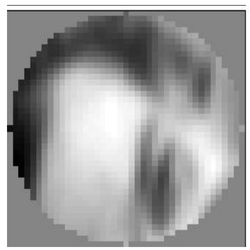
- Regular grid
  - Vogel & Schiele, 2003
  - Fei-Fei & Perona, 2005
- Interest point detector
  - Csurka et al. 2004
  - Fei-Fei & Perona, 2005
  - Sivic et al. 2005
- Other methods
  - Random sampling (Vidal-Naquet & Ullman, 2002)
  - Segmentation-based patches (Barnard et al. 2003)

1. Feature extraction

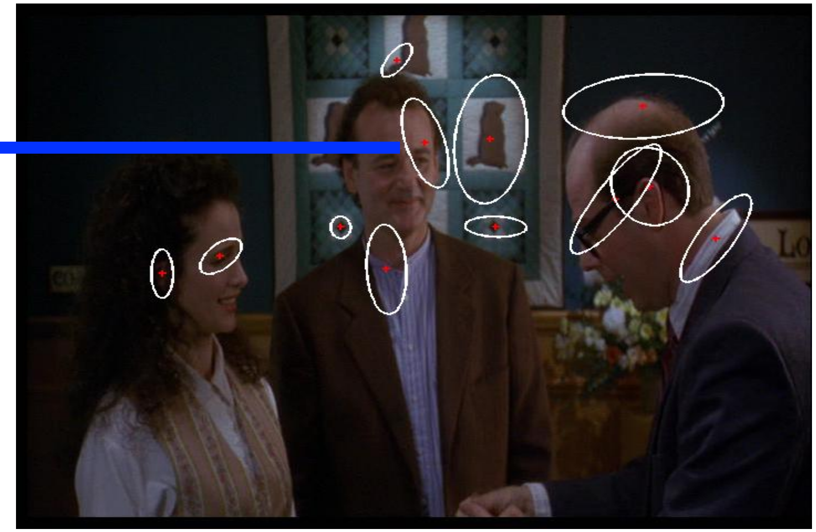


Compute SIFT descriptor

[Lowe'99]



Normalize patch



Detect patches

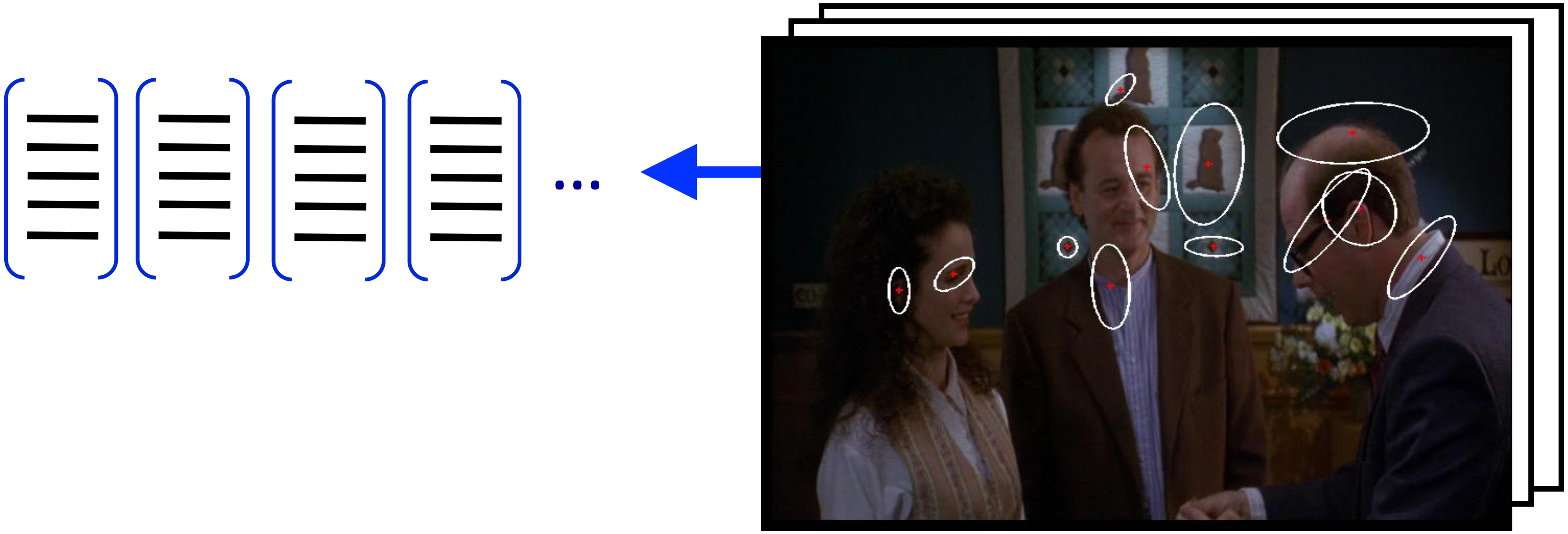
[Mikojaczyk and Schmid '02]

[Mata, Chum, Urban & Pajdla, '02]

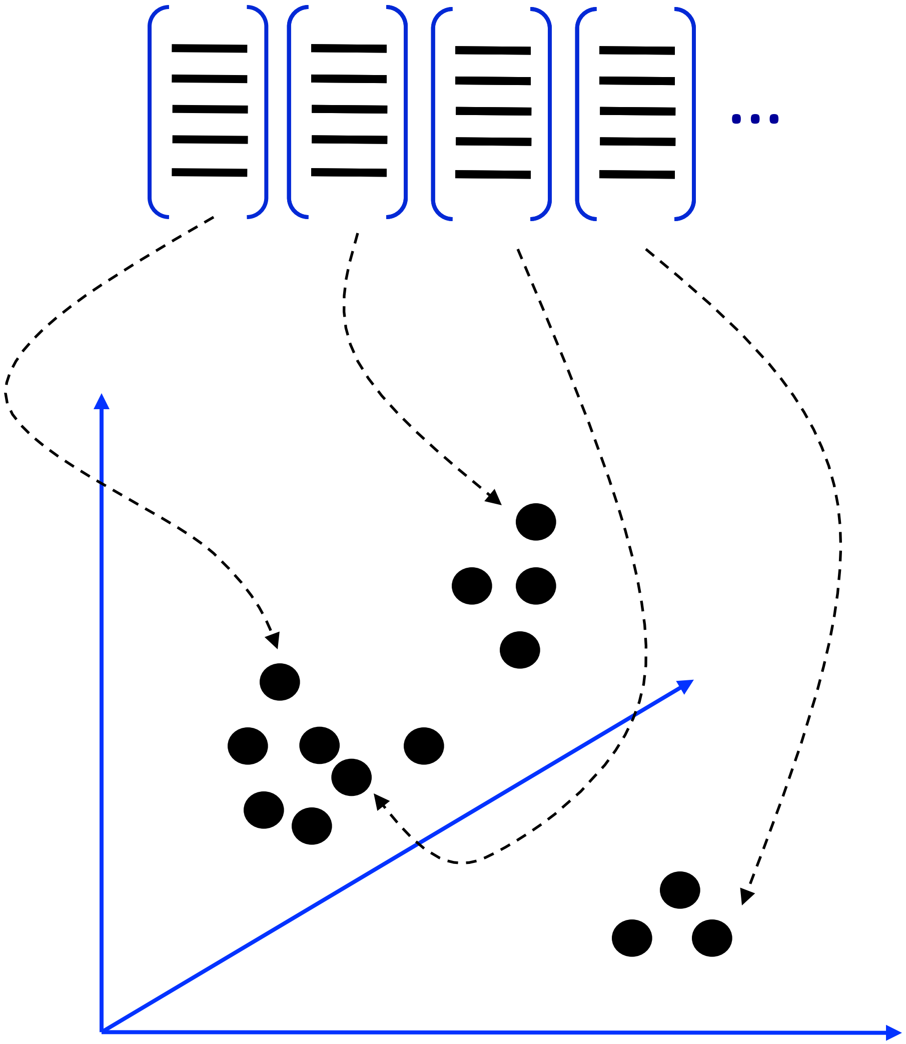
[Sivic & Zisserman, '03]

Slide credit: Josef Sivic

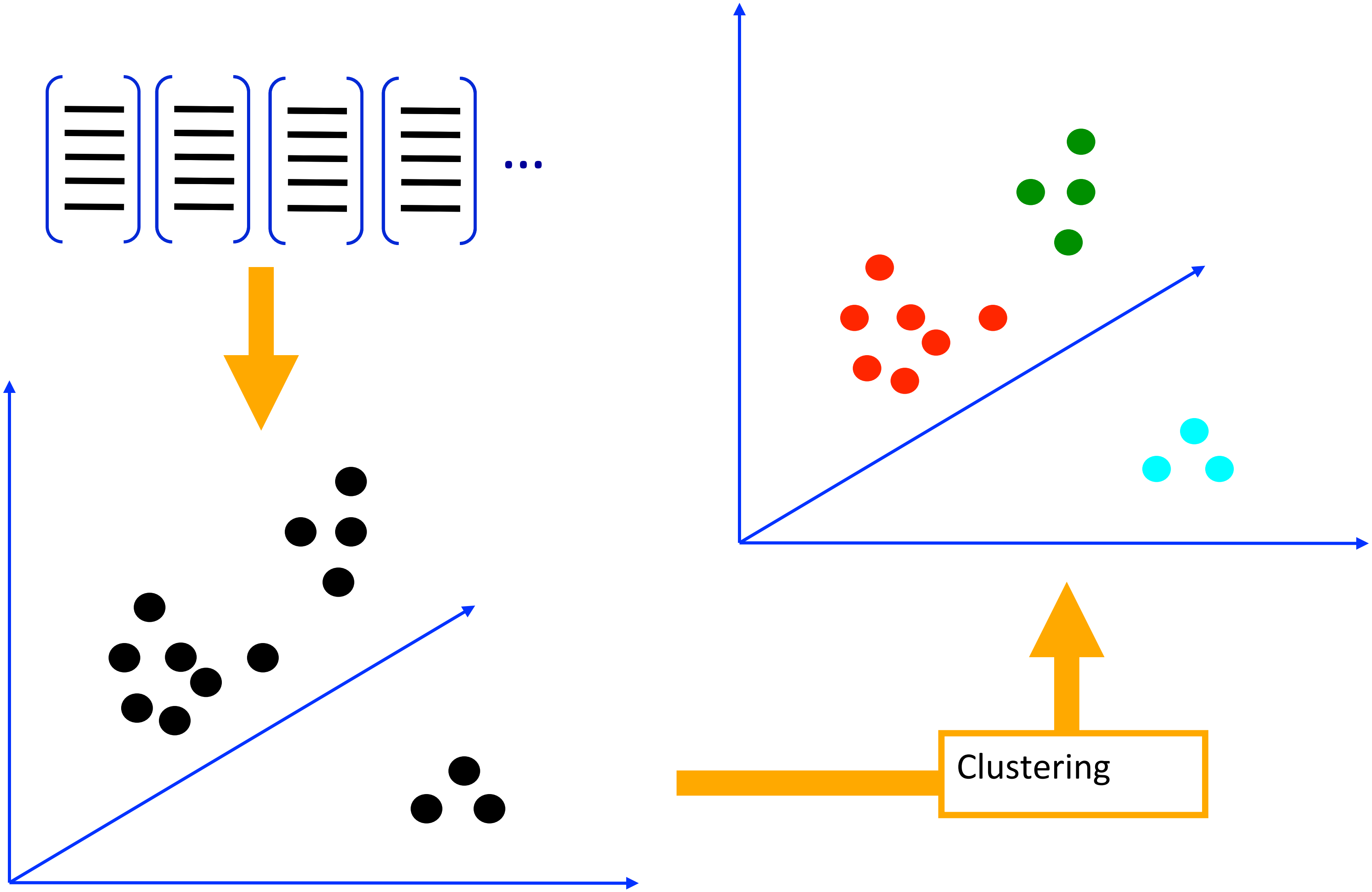
1. Feature extraction



2. Learning the visual vocabulary

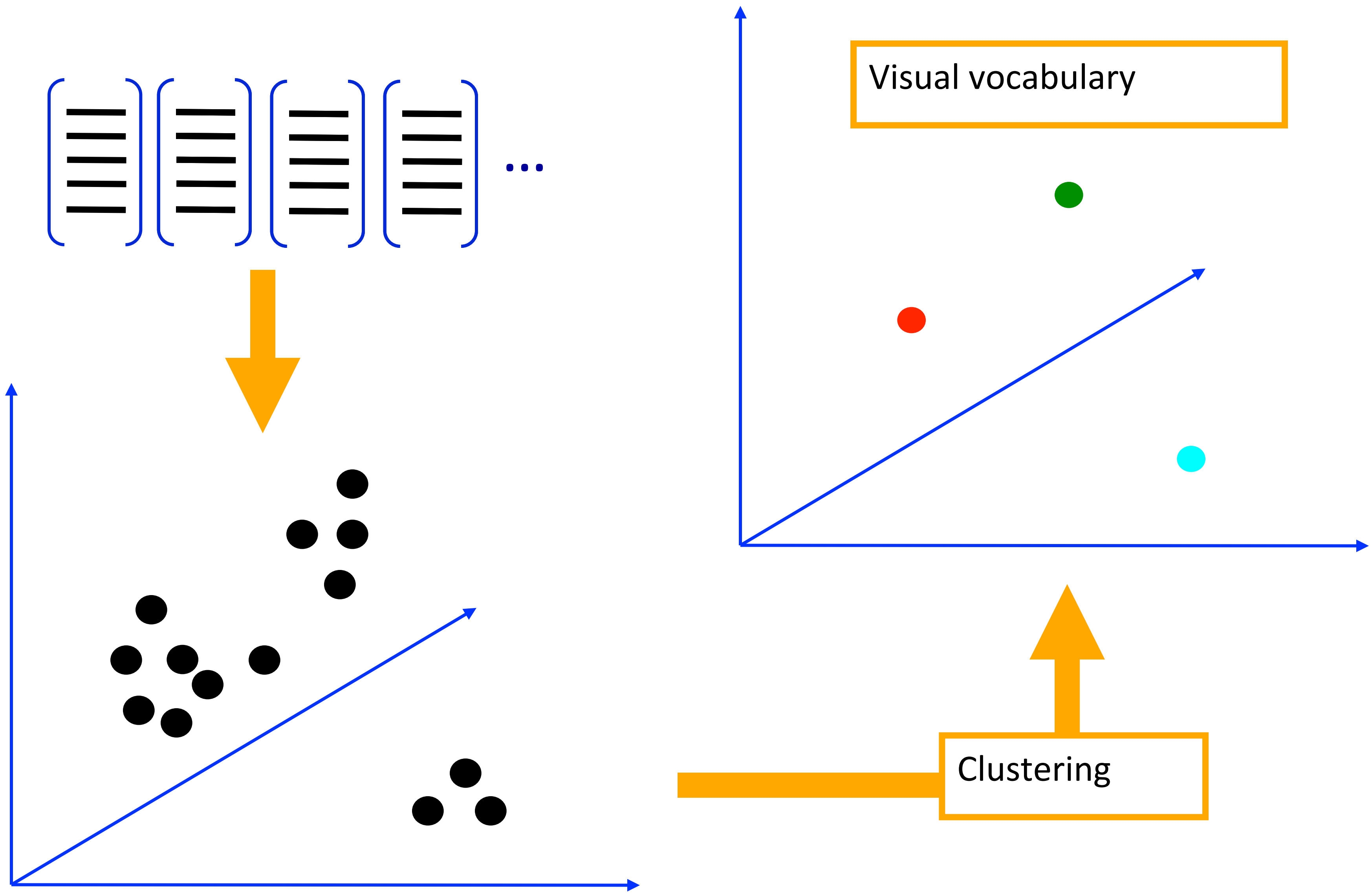


2. Learning the visual vocabulary



Slide credit: Josef Sivic

2. Learning the visual vocabulary



Slide credit: Josef Sivic

K-means clustering

- Want to minimize sum of squared Euclidean distances between points x_i and their nearest cluster centers m_k

$$D(X,M) = \sum_{\text{cluster } k} \sum_{\substack{\text{point } i \text{ in} \\ \text{cluster } k}} (x_i - m_k)^2$$

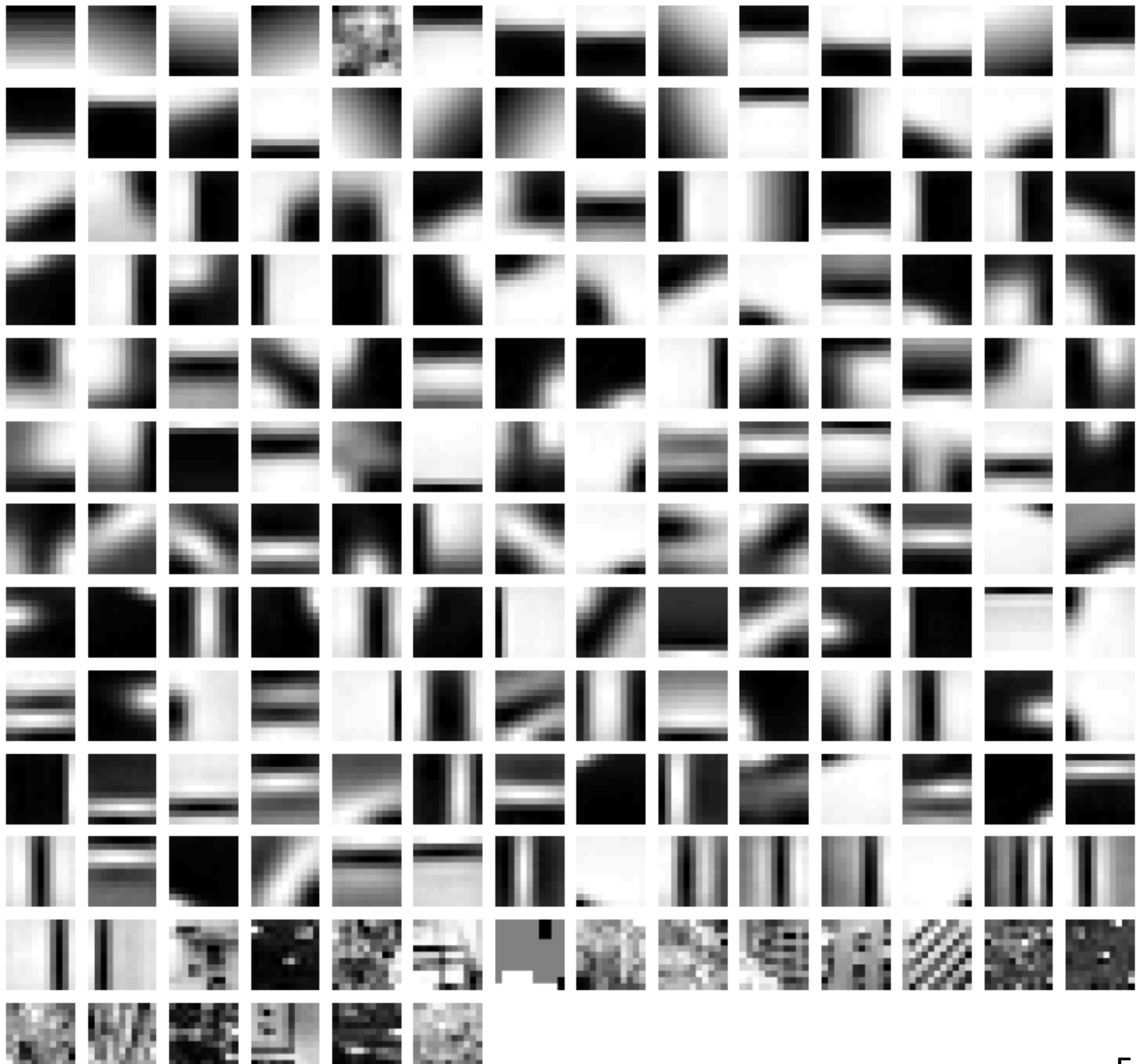
- Algorithm:
- Randomly initialize K cluster centers
- Iterate until convergence:
  - Assign each data point to the nearest center
  - Recompute each cluster center as the mean of all points assigned to it

From clustering to vector quantization

- Clustering is a common method for learning a visual vocabulary or codebook
  - Unsupervised learning process
  - Each cluster center produced by k-means becomes a codevector
  - Codebook can be learned on separate training set
  - Provided the training set is sufficiently representative, the codebook will be "universal"

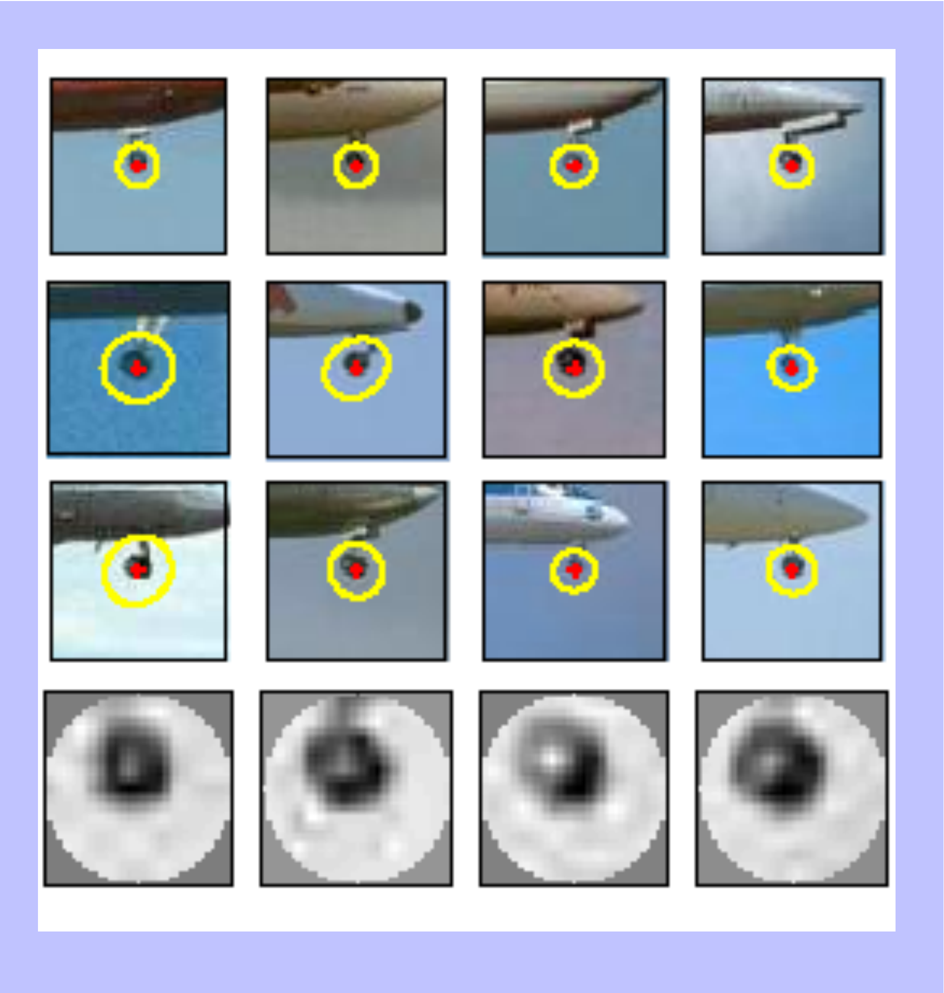
- The codebook is used for quantizing features
  - A *vector quantizer* takes a feature vector and maps it to the index of the nearest codevector in a codebook
  - Codebook = visual vocabulary
  - Codevector = visual word

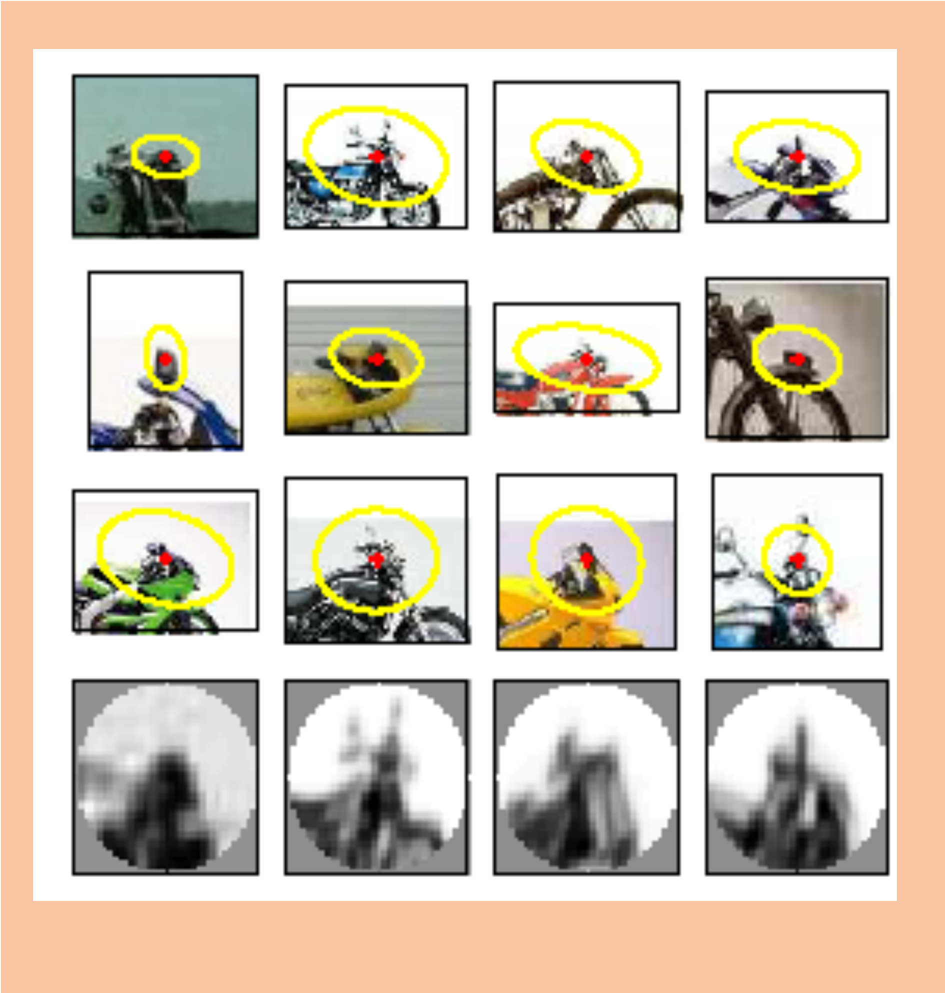
Example visual vocabulary



Fei-Fei et al. 2005

Image patch examples of visual words

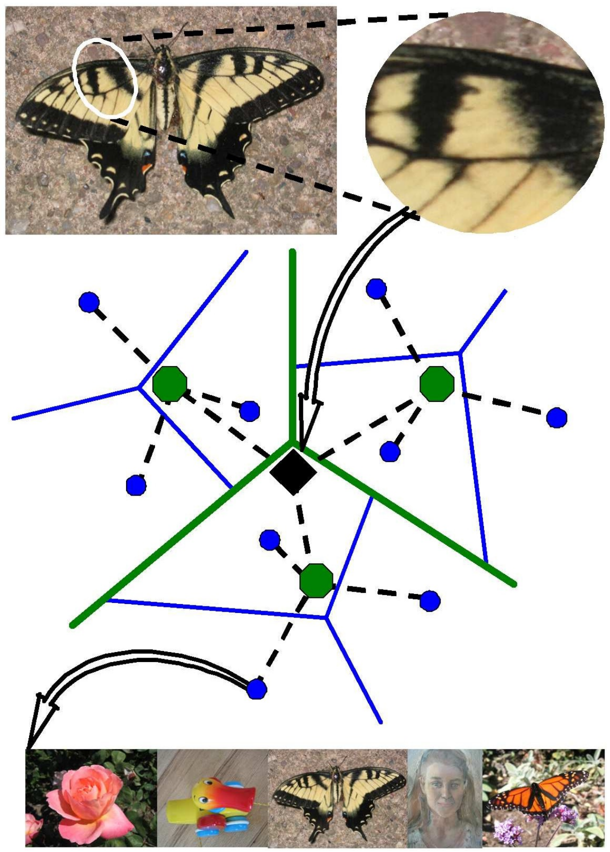




Sivic et al. 2005

Visual vocabularies: Issues

- How to choose vocabulary size?
  - Too small: visual words not representative of all patches
  - Too large: quantization arti
- Generative or discriminati
- Computational efficiency
  - Vocabulary trees
    (Nister & Stewenius, 2006)



3. Image representation

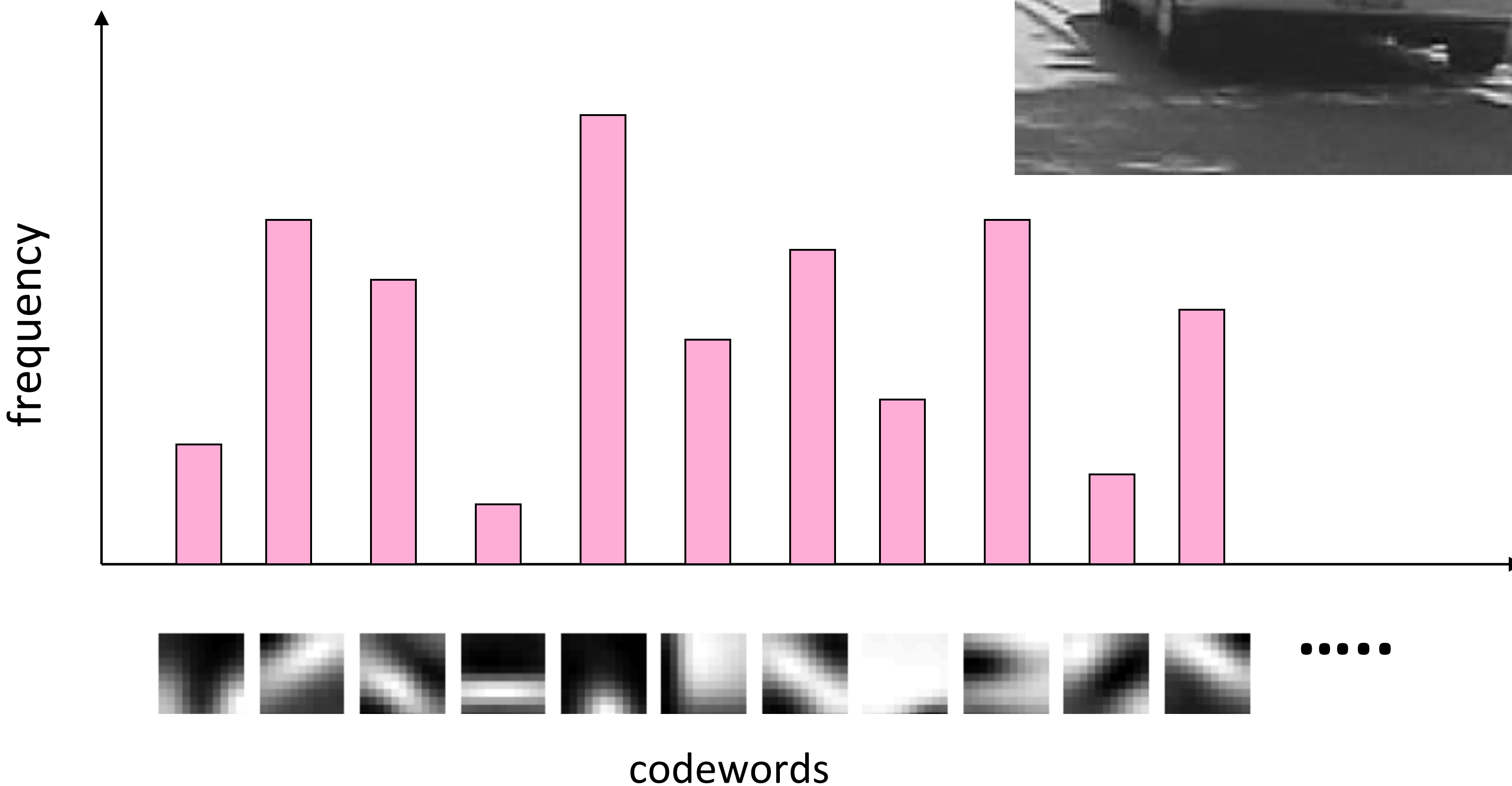
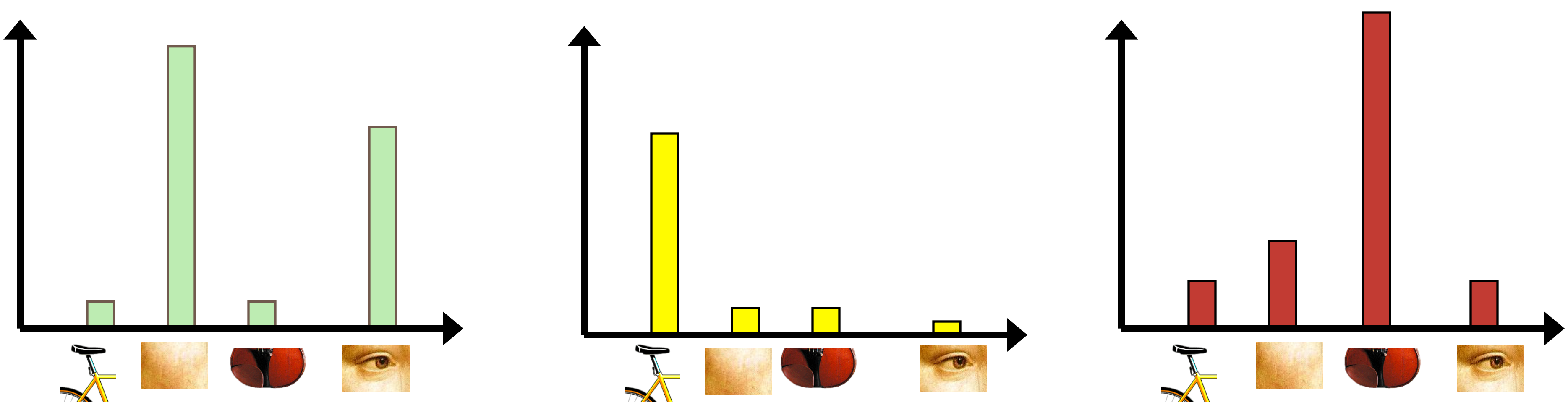
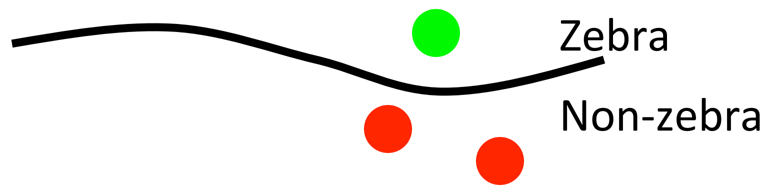


Image classification

- Given the bag-of-features representations of images from different classes, how do we learn a model for distinguishing them?



Discriminative and generative methods for bags of features



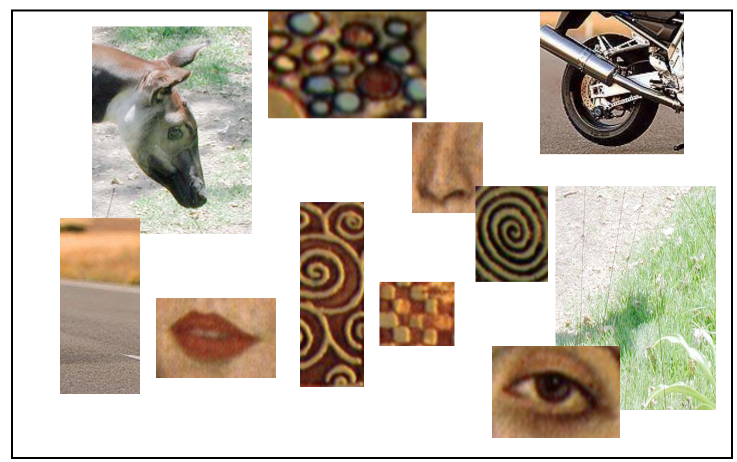
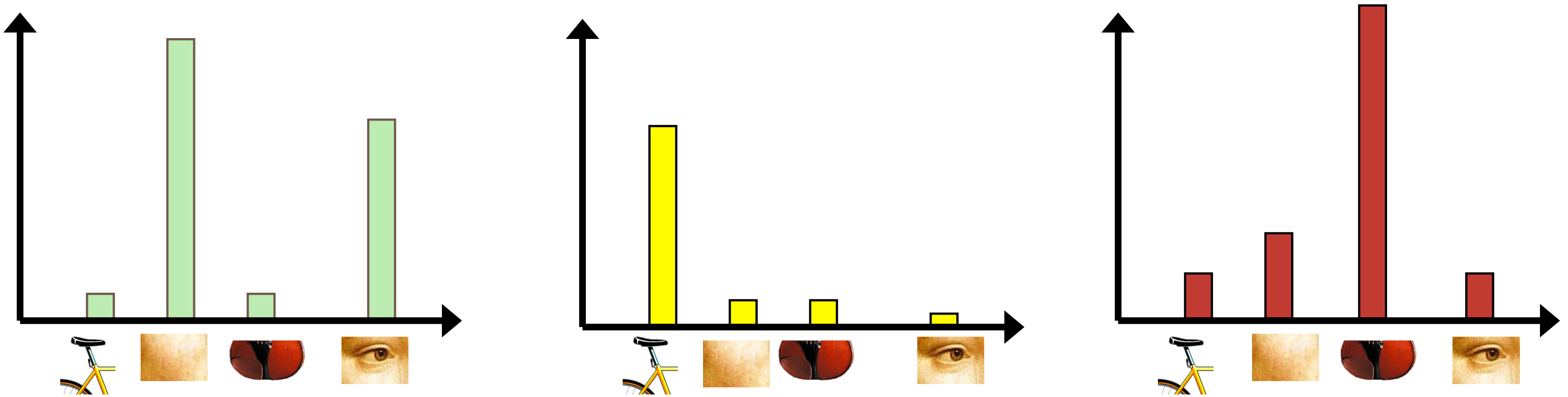


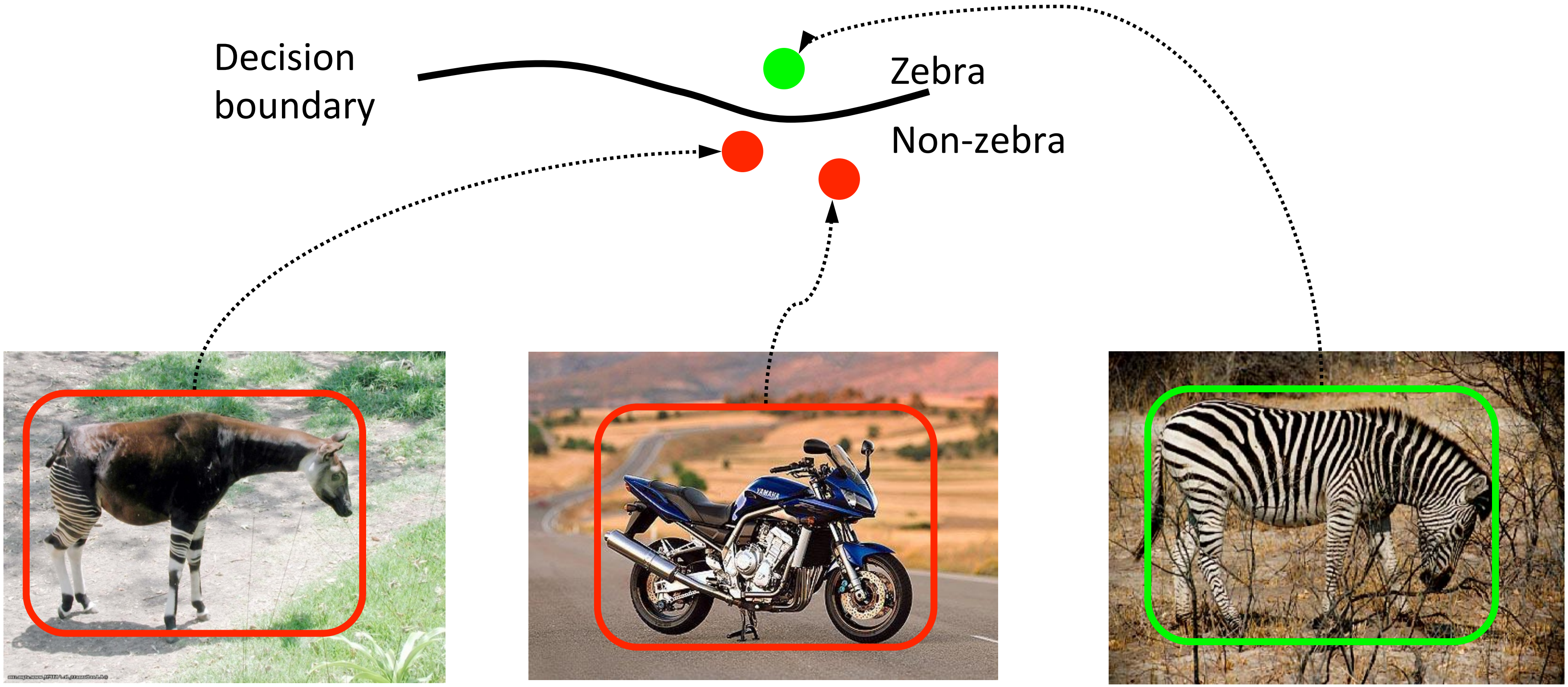
Image classification

- Given the bag-of-features representations of images from different classes, how do we learn a model for distinguishing them?



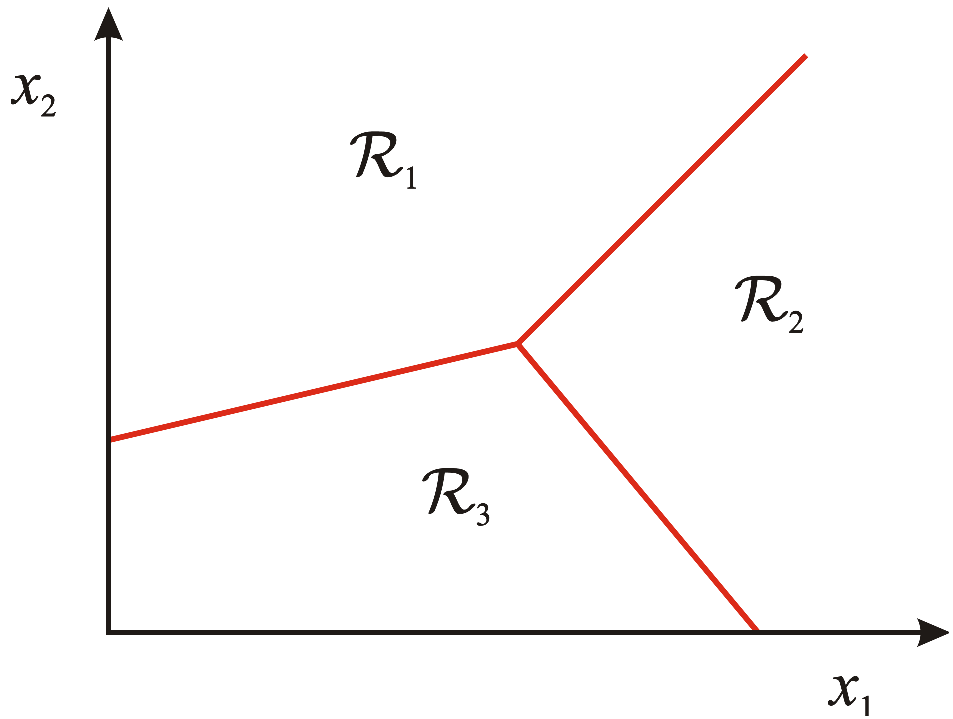
Discriminative methods

- Learn a decision rule (classifier) assigning bag-of-features representations of images to different classes



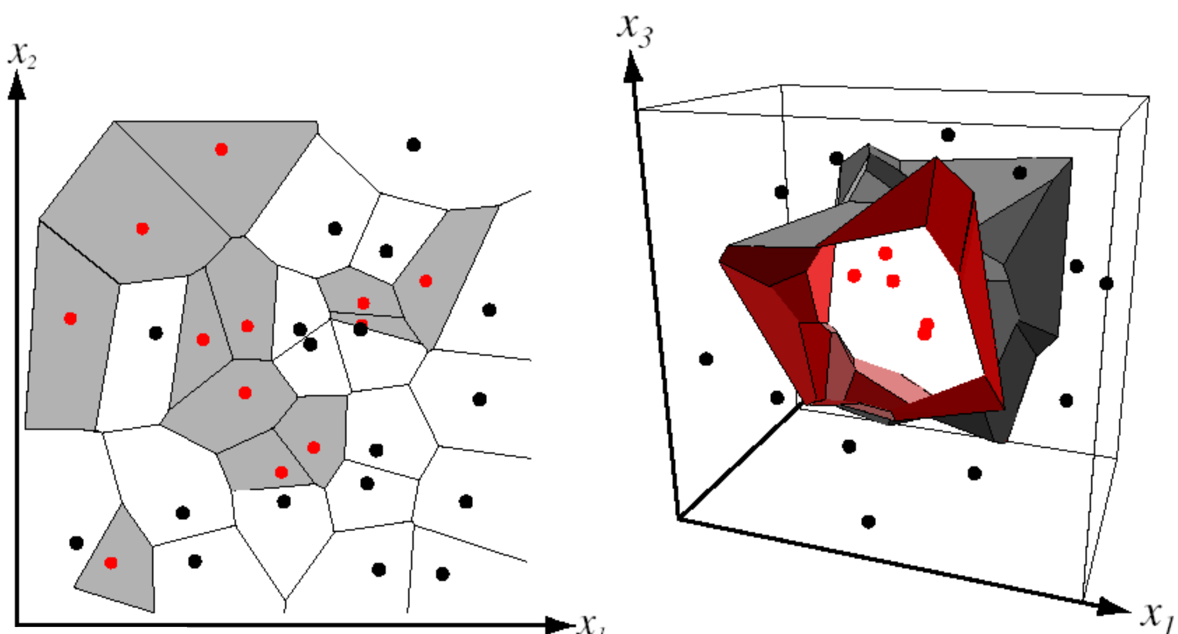
Classification

- Assign input vector to one of two or more classes
- Any decision rule divides input space into *decision regions* separated by *decision boundaries*



Nearest Neighbor Classifier

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



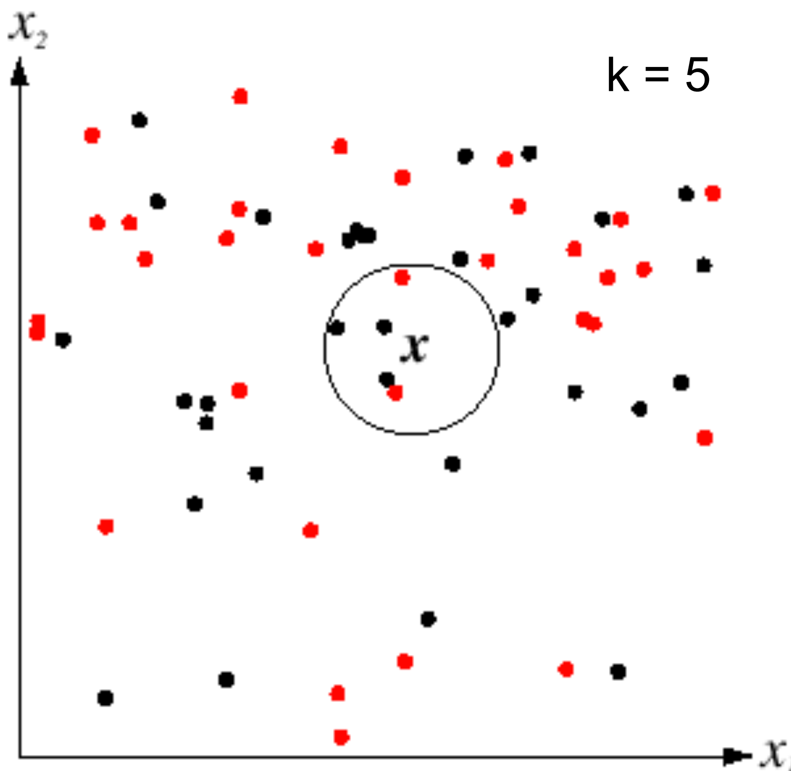
from Duda *et al.*

Voronoi partitioning of feature space
for two-category 2D and 3D data

Source: D. Lowe

K-Nearest Neighbors

- For a new point, find the k closest points from training data
- Labels of the k points “vote” to classify
- Works well provided there is lots of data and the distance function is good



Source: D. Lowe

Functions for comparing histograms

- L1 distance

$$D(h_1, h_2) = \sum_{i=1}^{N} | h_1(i) - h_2(i) |$$

- χ^2 distance

$$D(h_1, h_2) = \sum_{i=1}^{N} \frac{\left(h_1(i) - h_2(i)\right)^2}{h_1(i) + h_2(i)}$$

- Quadratic distance (*cross-bin*)

$$D(h_1, h_2) = \sum_{i,j} A_{ij} (h_1(i) - h_2(j))^2$$

Jan Puzicha, Yossi Rubner, Carlo Tomasi, Joachim M. Buhmann:
[Empirical Evaluation of Dissimilarity Measures for Color and Texture](). ICCV 1999

Earth Mover's Distance

- Each image is represented by a *signature S* consisting of a set of *centers* $\{m_i\}$ and *weights* $\{w_i\}$
- Centers can be codewords from universal vocabulary, clusters of features in the image, or individual features (in which case quantization is not required)
- Earth Mover's Distance has the form

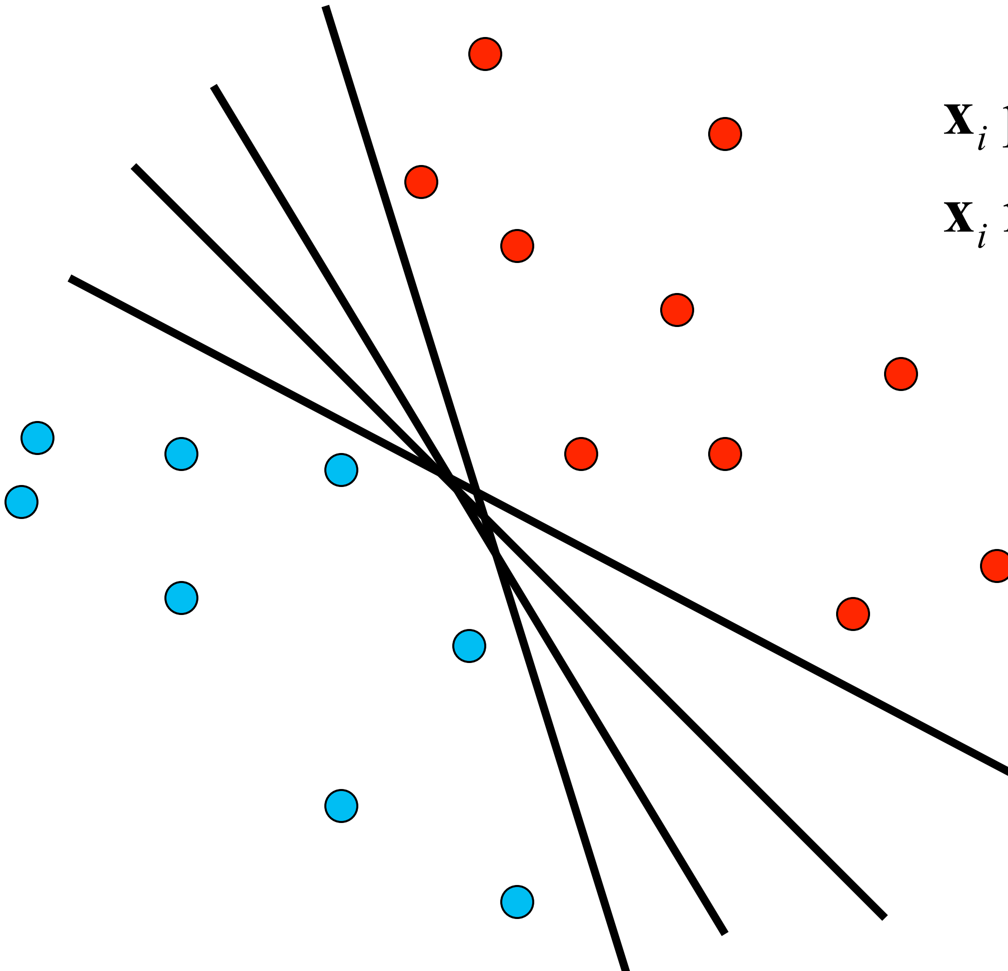
$$EMD(S_1, S_2) = \sum_{i,j} \frac{f_{ij}\, d(m_{1i}, m_{2j})}{f_{ij}}$$

where the *flows* f_{ij} are given by the solution of a *transportation problem*

Y. Rubner, C. Tomasi, and L. Guibas: A Metric for Distributions with Applications to Image Databases. ICCV 1998

Linear classifiers

- Find linear function (*hyperplane*) to separate positive and negative examples



$$\mathbf{x}_i \text{ positive}: \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 0$$

$$\mathbf{x}_i \text{ negative}: \quad \mathbf{x}_i \cdot \mathbf{w} + b < 0$$

Which hyperplane is best?

Slide: S. Lazebnik

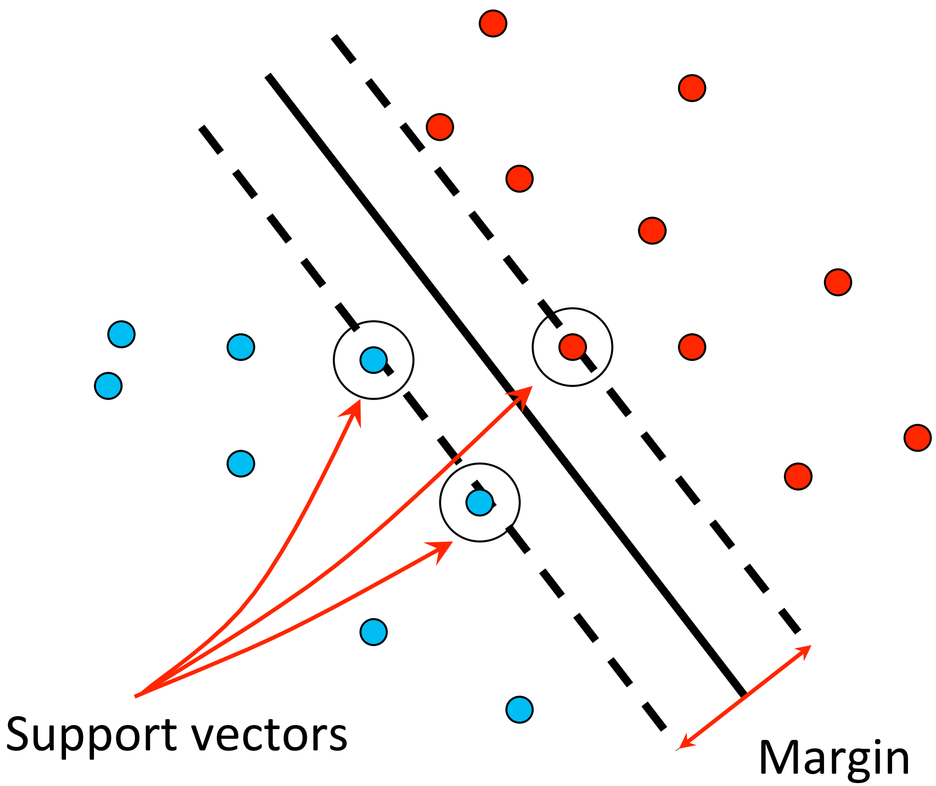
Support vector machines

- Find hyperplane that maximizes the *margin* between the positive and negative examples

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Support vector machines

- Find hyperplane that maximizes the *margin* between the positive and negative examples



$\mathbf{x}_i$ positive $(y_i = 1)$: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$

$\mathbf{x}_i$ negative $(y_i = -1)$: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$

For support vectors, $\quad \mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$

Distance between point and hyperplane: $\quad \dfrac{|\mathbf{x}_i \cdot \mathbf{w} + b|}{\|\mathbf{w}\|}$

Therefore, the margin is $2 / \|\mathbf{w}\|$

C. Burges, [A Tutorial on Support Vector Machines for Pattern Recognition](), Data Mining and Knowledge Discovery, 1998

Finding the maximum margin hyperplane

1. Maximize margin $2/||\mathbf{w}||$
2. Correctly classify all training data:

$$\mathbf{x}_i \text{ positive } (y_i = 1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$

$$\mathbf{x}_i \text{ negative } (y_i = -1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$$

- *Quadratic optimization problem*:
- Minimize $\frac{1}{2}\mathbf{w}^T\mathbf{w}$

  Subject to $y_i(\mathbf{w}\cdot\boldsymbol{x}_i+b) \geq 1$

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Finding the maximum margin hyperplane

- Solution: $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$

learned weight

Support vector

C. Burges, [A Tutorial on Support Vector Machines for Pattern Recognition](), Data Mining and Knowledge Discovery, 1998

Finding the maximum margin hyperplane

- Solution: $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$

  $b = y_i - \mathbf{w} \cdot \mathbf{x}_i$ for any support vector

- Classification function (decision boundary):

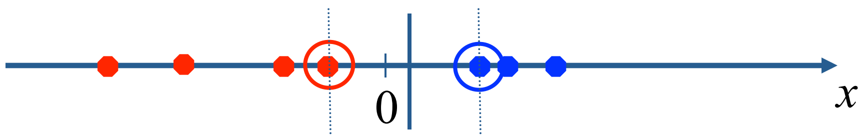
$$\mathbf{w} \cdot \mathbf{x} + b = \sum_i \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b$$

- Notice that it relies on an *inner product* between the test point $\boldsymbol{x}$ and the support vectors $\boldsymbol{x}_i$
- Solving the optimization problem also involves computing the inner products $\boldsymbol{x}_i \cdot \boldsymbol{x}_j$ between all pairs of training points

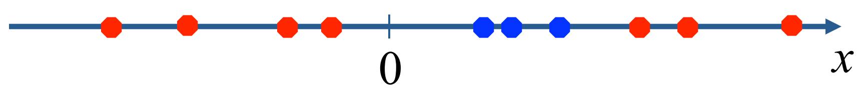
C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Nonlinear SVMs

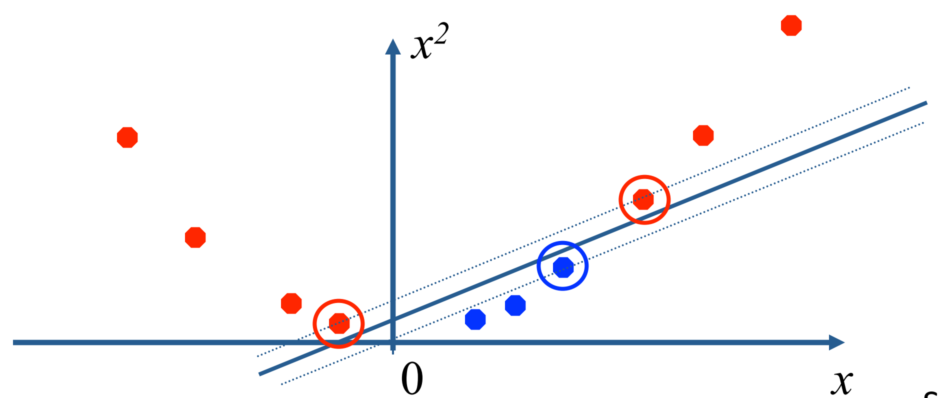
- Datasets that are linearly separable work out great:



- But what if the dataset is just too hard?



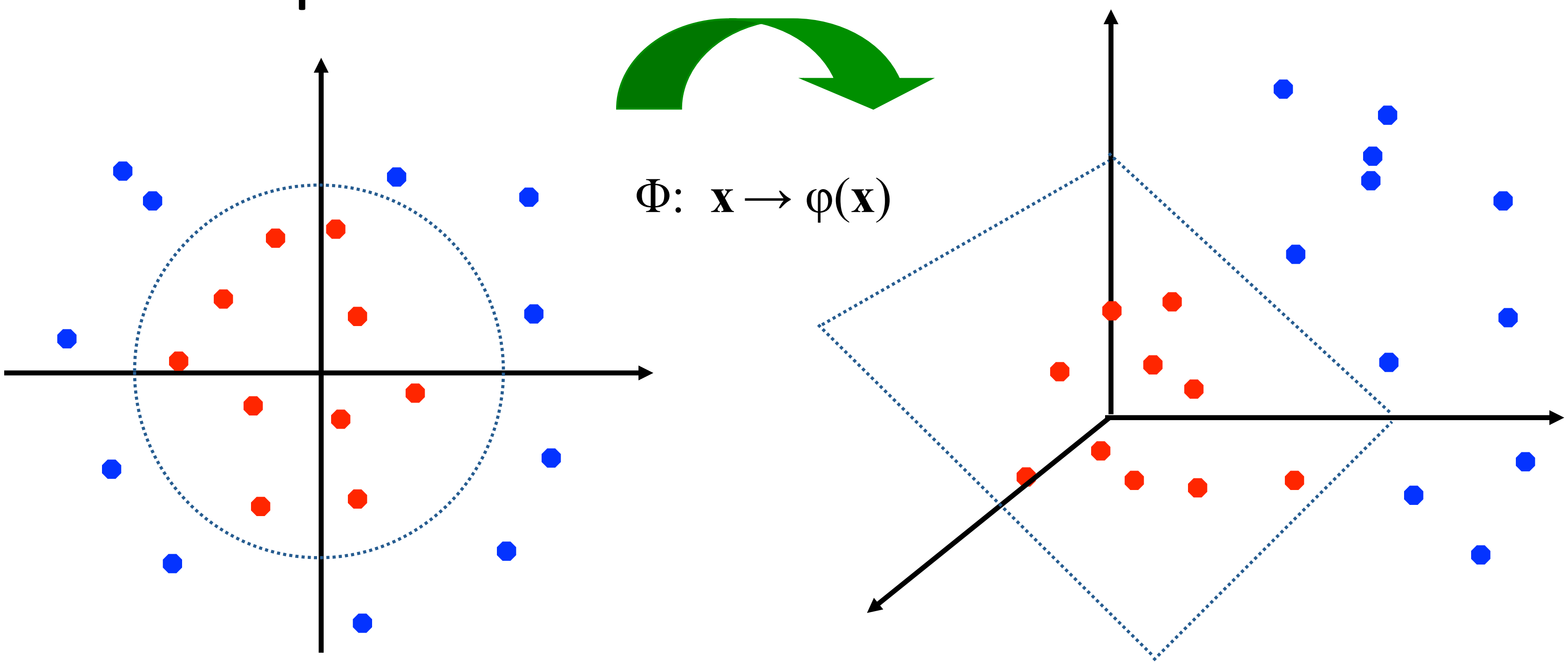
- We can map it to a higher-dimensional space:



Slide credit: Andrew Moore

Nonlinear SVMs

- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:



Slide credit: Andrew Moore

Nonlinear SVMs

- *The kernel trick*: instead of explicitly computing the lifting transformation $\boldsymbol{\varphi}(\mathbf{x})$, define a kernel function K such that

$$K(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\varphi}(\mathbf{x}_i) \cdot \boldsymbol{\varphi}(\mathbf{x}_j)$$

- (to be valid, the kernel function must satisfy *Mercer's condition*)
- This gives a nonlinear decision boundary in the original feature space:

$$\sum_i \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$$

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Kernels for bags of features

- Histogram intersection kernel:

$$I(h_1, h_2) = \sum_{i=1}^{N} \min(h_1(i), h_2(i))$$

- Generalized Gaussian kernel:

$$K(h_1, h_2) = \exp\left(-\frac{1}{A} D(h_1, h_2)^2 \right)$$

- *D* can be Euclidean distance, χ^2 distance, Earth Mover's Distance, etc.

J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid, Local Features and Kernels for Classifcation of Texture and Object Categories: A Comprehensive Study, IJCV 2007

Summary: SVMs for image classification

1. Pick an image representation (in our case, bag of features)
2. Pick a kernel function for that representation
3. Compute the matrix of kernel values between every pair of training examples
4. Feed the kernel matrix into your favorite SVM solver to obtain support vectors and weights
5. At test time: compute kernel values for your test example and each support vector, and combine them with the learned weights to get the value of the decision function

What about multi-class SVMs?

- Unfortunately, there is no “definitive” multi-class SVM formulation
- In practice, we have to obtain a multi-class SVM by combining multiple two-class SVMs
- One vs. others
  - Traning: learn an SVM for each class vs. the others
  - Testing: apply each SVM to test example and assign to it the class of the SVM that returns the highest decision value
- One vs. one
  - Training: learn an SVM for each pair of classes
  - Testing: each learned SVM “votes” for a class to assign to the test example

Slide: S. Lazebnik

SVMs: Pros and cons

- Pros
  - Many publicly available SVM packages: http://www.kernel-machines.org/software
  - Kernel-based framework is very powerful, flexible
  - SVMs work very well in practice, even with very small training sample sizes

- Cons
  - No "direct" multi-class SVM, must combine two-class SVMs
  - Computation, memory
    - During training time, must compute matrix of kernel values for every pair of examples
    - Learning can take a very long time for large-scale problems

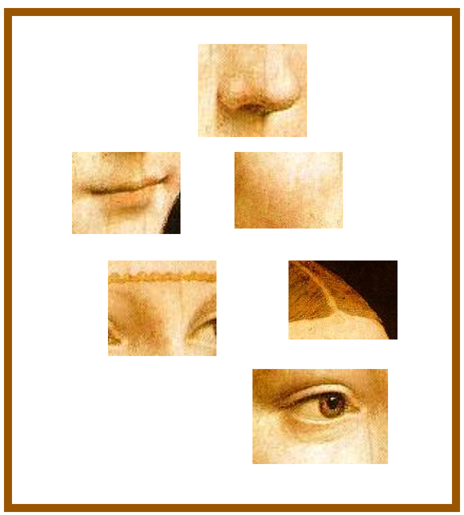
Slide: S. Lazebnik

Summary: Discriminative methods

- Nearest-neighbor and k-nearest-neighbor classifiers
  - L1 distance, χ^2 distance, quadratic distance, Earth Mover's Distance
- Support vector machines
  - Linear classifiers
  - Margin maximization
  - The kernel trick
  - Kernel functions: histogram intersection, generalized Gaussian, pyramid match
  - Multi-class
- Of course, there are many other classifiers out there
  - Neural networks, boosting, decision trees, …

Generative learning methods for bags of features

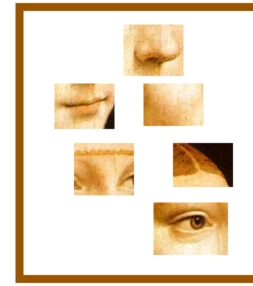
- Model the probability of a bag of features given a class



Generative methods

- We will cover two models, both inspired by text document analysis:
  - Naïve Bayes
  - Probabilistic Latent Semantic Analysis

The Naïve Bayes model



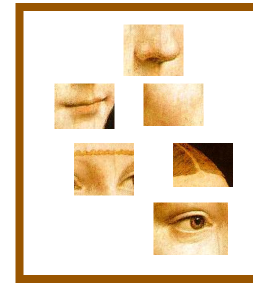
- Assume that each feature is conditionally independent *given the class*

$$p(f_1, \ldots, f_N \mid c) = \prod_{i=1}^{N} p(f_i \mid c)$$

f_i: ith feature in the image
N: number of features in the image

Csurka et al. 2004

The Naïve Bayes model



- Assume that each feature is conditionally independent *given the class*

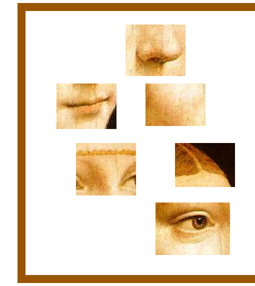
$$p(f_1, \ldots, f_N \mid c) = \prod_{i=1}^{N} p(f_i \mid c) = \prod_{j=1}^{M} p(w_j \mid c)^{n(w_j)}$$

f_i: ith feature in the image
N: number of features in the image

w_j: jth visual word in the vocabulary
M: size of visual vocabulary
$n(w_j)$: number of features of type w_j in the image

Csurka et al. 2004

The Naïve Bayes model



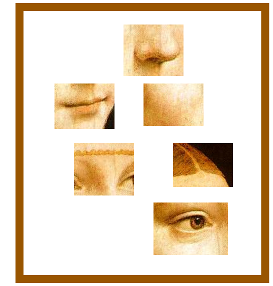
- Assume that each feature is conditionally independent *given the class*

$$p(f_1, \ldots, f_N \mid c) = \prod_{i=1}^{N} p(f_i \mid c) = \prod_{j=1}^{M} p(w_j \mid c)^{n(w_j)}$$

$$p(w_j \mid c) = \frac{\text{No. of features of type } w_j \text{ in training images of class } c}{\text{Total no. of features in training images of class } c}$$

Csurka et al. 2004

The Naïve Bayes model



- Assume that each feature is conditionally independent *given the class*

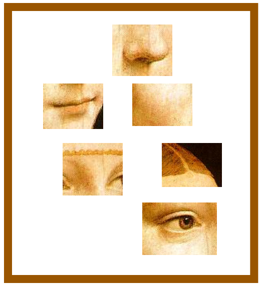
$$p(f_1, \ldots, f_N \mid c) = \prod_{i=1}^{N} p(f_i \mid c) = \prod_{j=1}^{M} p(w_j \mid c)^{n(w_j)}$$

$$p(w_j \mid c) = \frac{\text{No. of features of type } w_j \text{ in training images of class } c + 1}{\text{Total no. of features in training images of class } c + M}$$

(*Laplace smoothing* to avoid zero counts)

Csurka et al. 2004

The Naïve Bayes model



- Maximum A Posteriori decision:

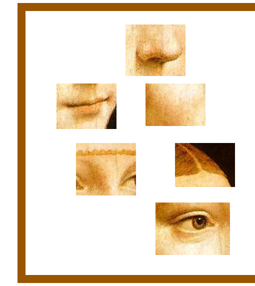
$$c^* = \arg\max_c \; p(c) \prod_{j=1}^{M} p(w_j \mid c)^{n(w_j)}$$

$$= \arg\max_c \; \log p(c) + \sum_{j=1}^{M} n(w_j) \log p(w_j \mid c)$$

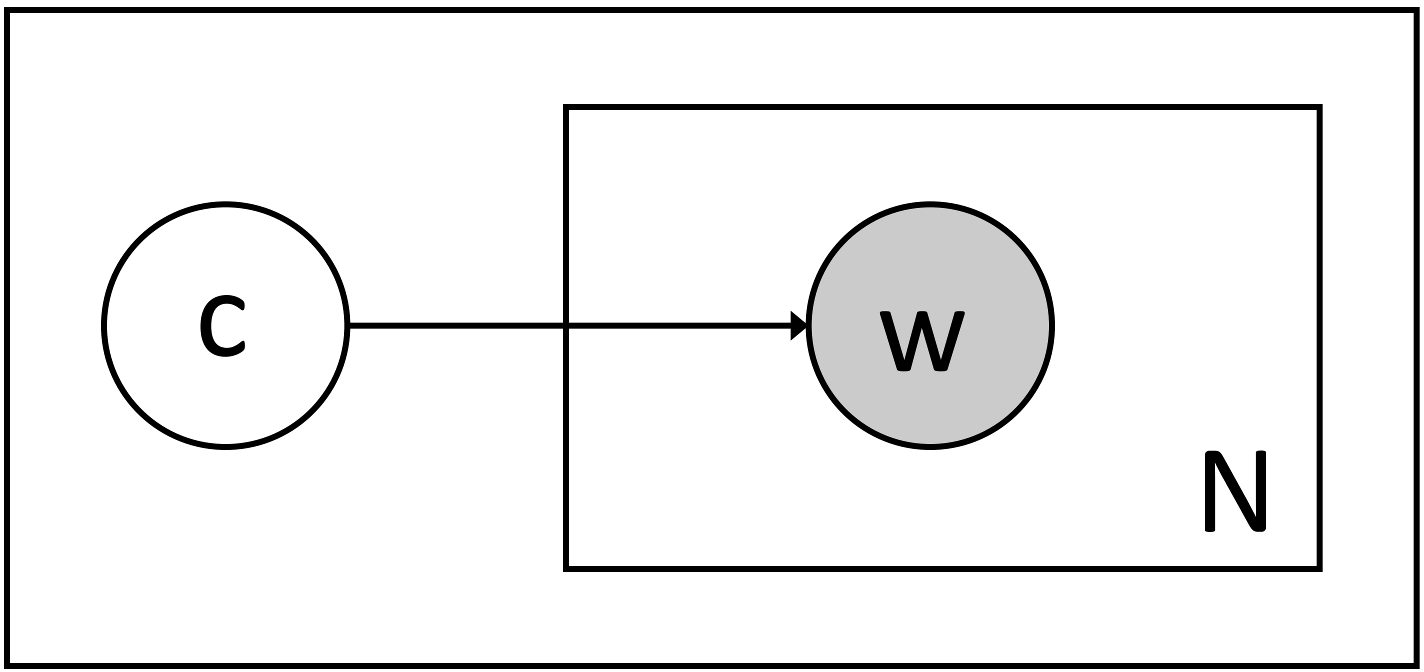
(you should compute the log of the likelihood instead of the likelihood itself in order to avoid underflow)

Csurka et al. 2004

The Naïve Bayes model

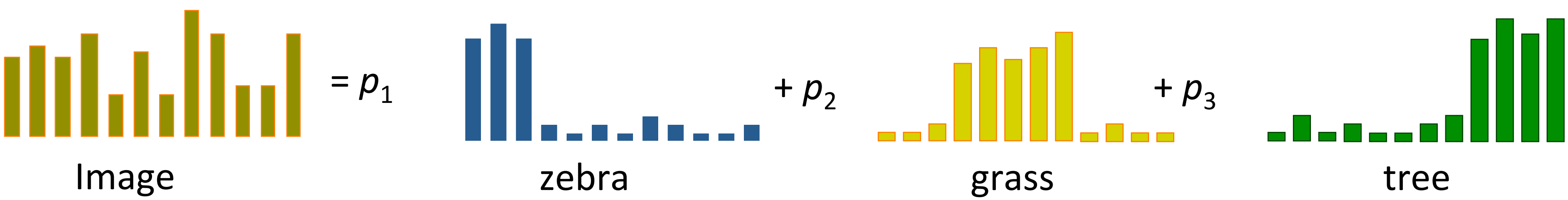


- “Graphical model”:



Csurka et al. 2004

Probabilistic Latent Semantic Analysis

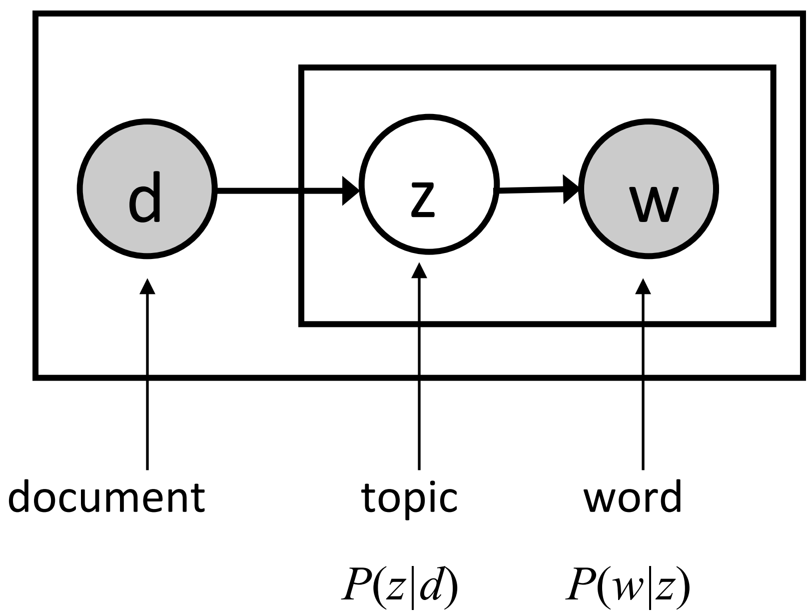


“visual topics”

T. Hofmann, [Probabilistic Latent Semantic Analysis](), UAI 1999

Probabilistic Latent Semantic Analysis

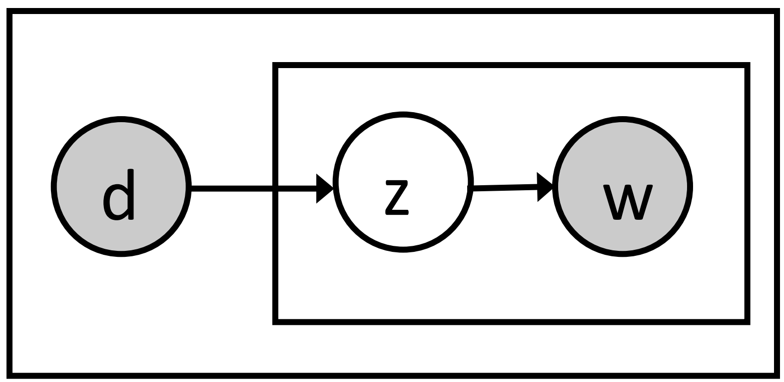
- Unsupervised technique
- Two-level generative model: a document is a mixture of topics, and each topic has its own characteristic word distribution



T. Hofmann, [Probabilistic Latent Semantic Analysis](), UAI 1999

Probabilistic Latent Semantic Analysis

- Unsupervised technique
- Two-level generative model: a document is a mixture of topics, and each topic has its own characteristic word distribution



$$p(w_i \mid d_j) = \sum_{k=1}^{K} p(w_i \mid z_k) p(z_k \mid d_j)$$

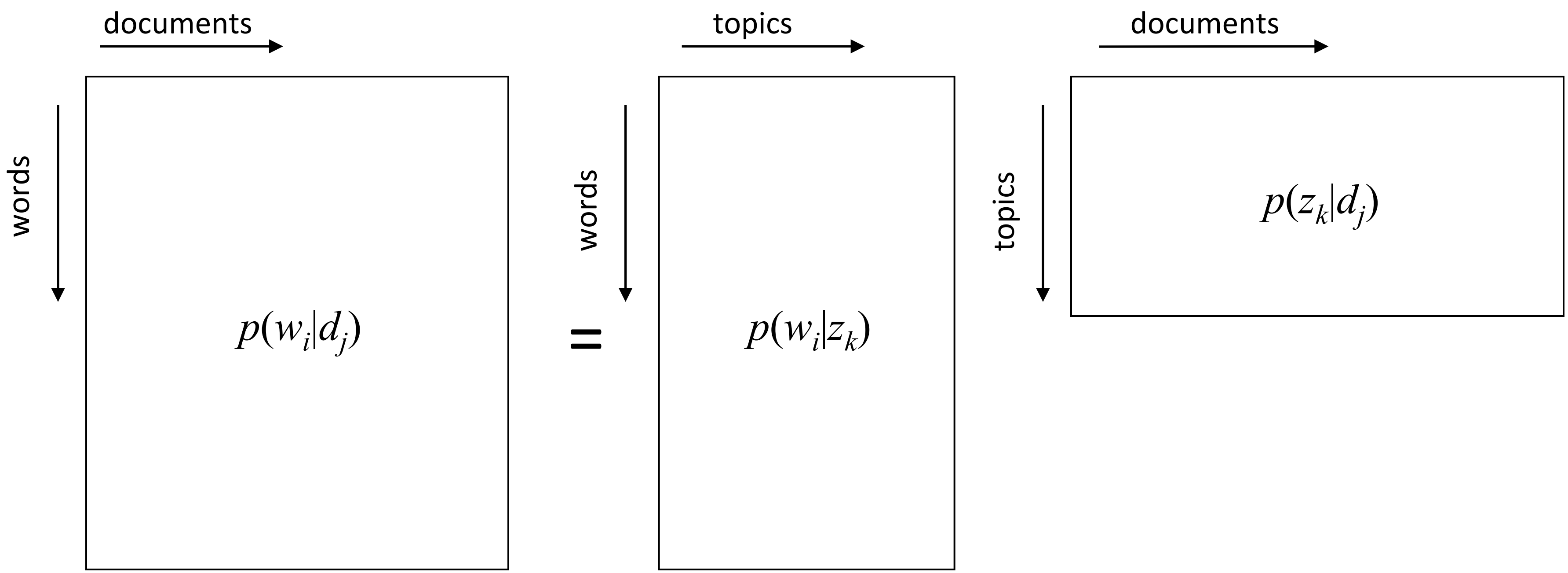
T. Hofmann, Probabilistic Latent Semantic Analysis, UAI 1999

The pLSA model

$$\underbrace{p(w_i \mid d_j)}_{\substack{\text{Probability of word i}\\ \text{in document j}\\ \text{(known)}}} = \sum_{k=1}^{K} \underbrace{p(w_i \mid z_k)}_{\substack{\text{Probability of}\\ \text{word i given}\\ \text{topic k}\\ \text{(unknown)}}} \underbrace{p(z_k \mid d_j)}_{\substack{\text{Probability of}\\ \text{topic k given}\\ \text{document j}\\ \text{(unknown)}}}$$

The pLSA model

$$p(w_i \mid d_j) = \sum_{k=1}^{K} p(w_i \mid z_k)\, p(z_k \mid d_j)$$



Observed codeword distributions ($M \times N$)

Codeword distributions per topic (class) ($M \times K$)

Class distributions per image ($K \times N$)

Learning pLSA parameters

Maximize likelihood of data:

Observed counts of word *i* in document *j*

$$L = \prod_{i=1}^{M} \prod_{j=1}^{N} P(w_i|d_j)^{n(w_i,d_j)}$$

M … number of codewords

N … number of images

$$\sum_{k=1}^{K} P(z_k|d_j)P(w_i|z_k)$$

Slide credit: Josef Sivic

Inference

- Finding the most likely topic (class) for an image:

$$z^* = \arg\max_z p(z \mid d)$$

Inference

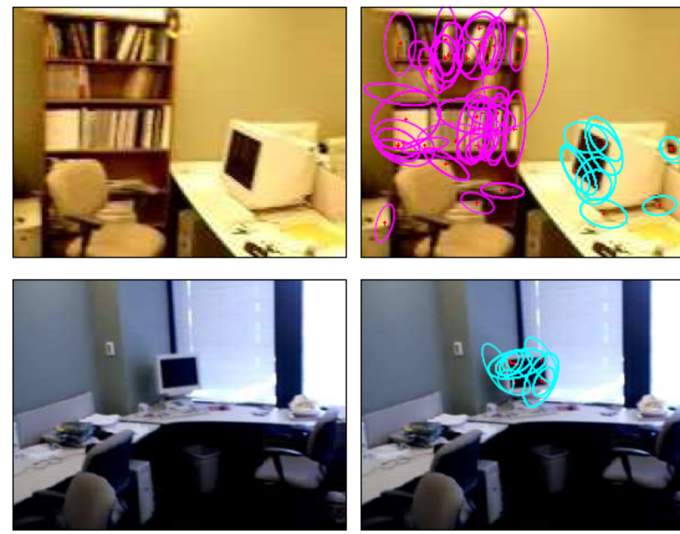
- Finding the most likely topic (class) for an image:

$$z^* = \arg\max_z p(z \mid d)$$

- Finding the most likely topic (class) for a visual word in a given image:

$$z^* = \arg\max_z p(z \mid w, d) = \arg\max_z \frac{p(w \mid z)\, p(z \mid d)}{\sum_{z'} p(w \mid z')\, p(z' \mid d)}$$

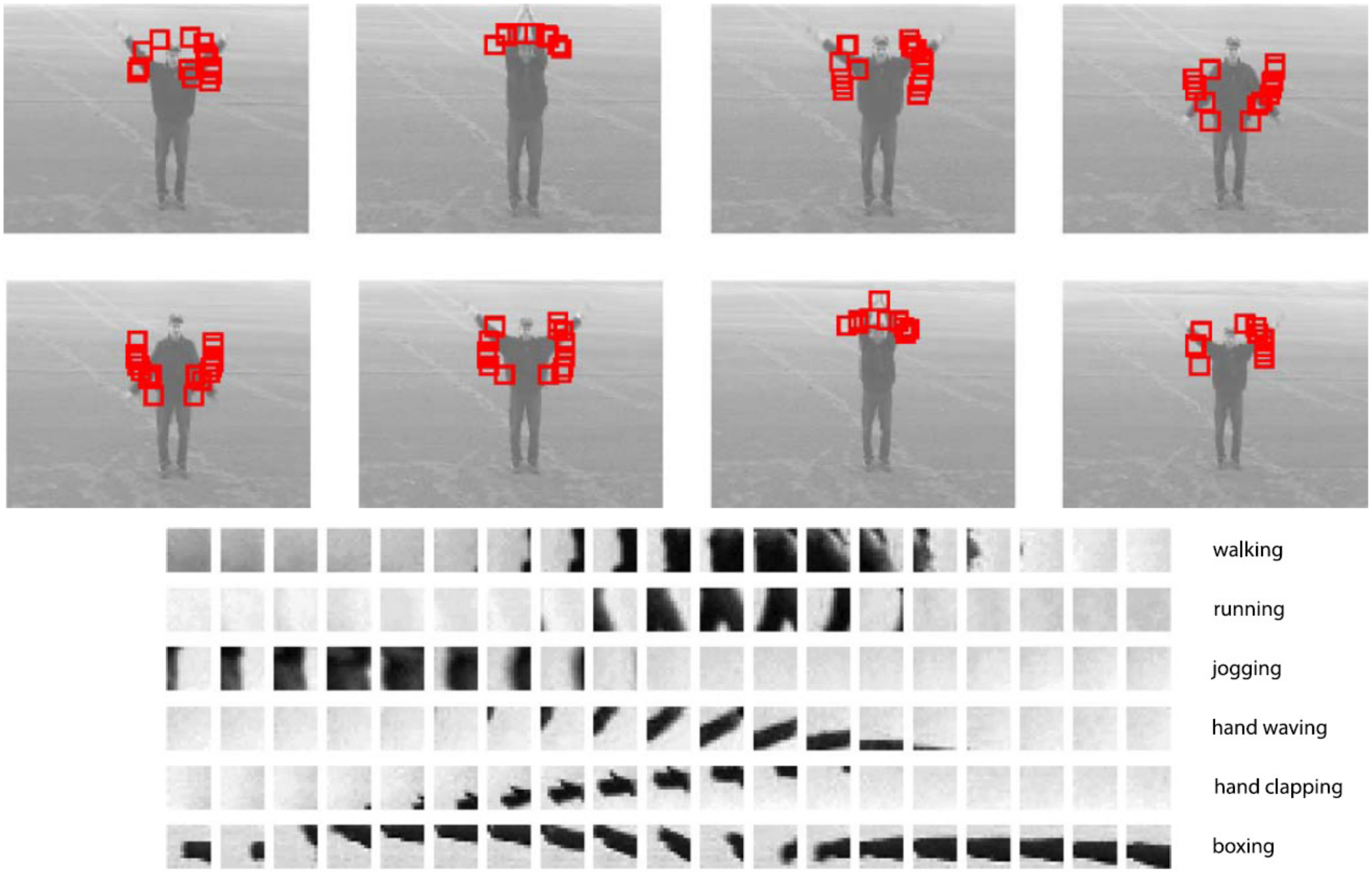
Topic discovery in images



J. Sivic, B. Russell, A. Efros, A. Zisserman, B. Freeman,
[Discovering Objects and their Location in Images](), *ICCV 2005*

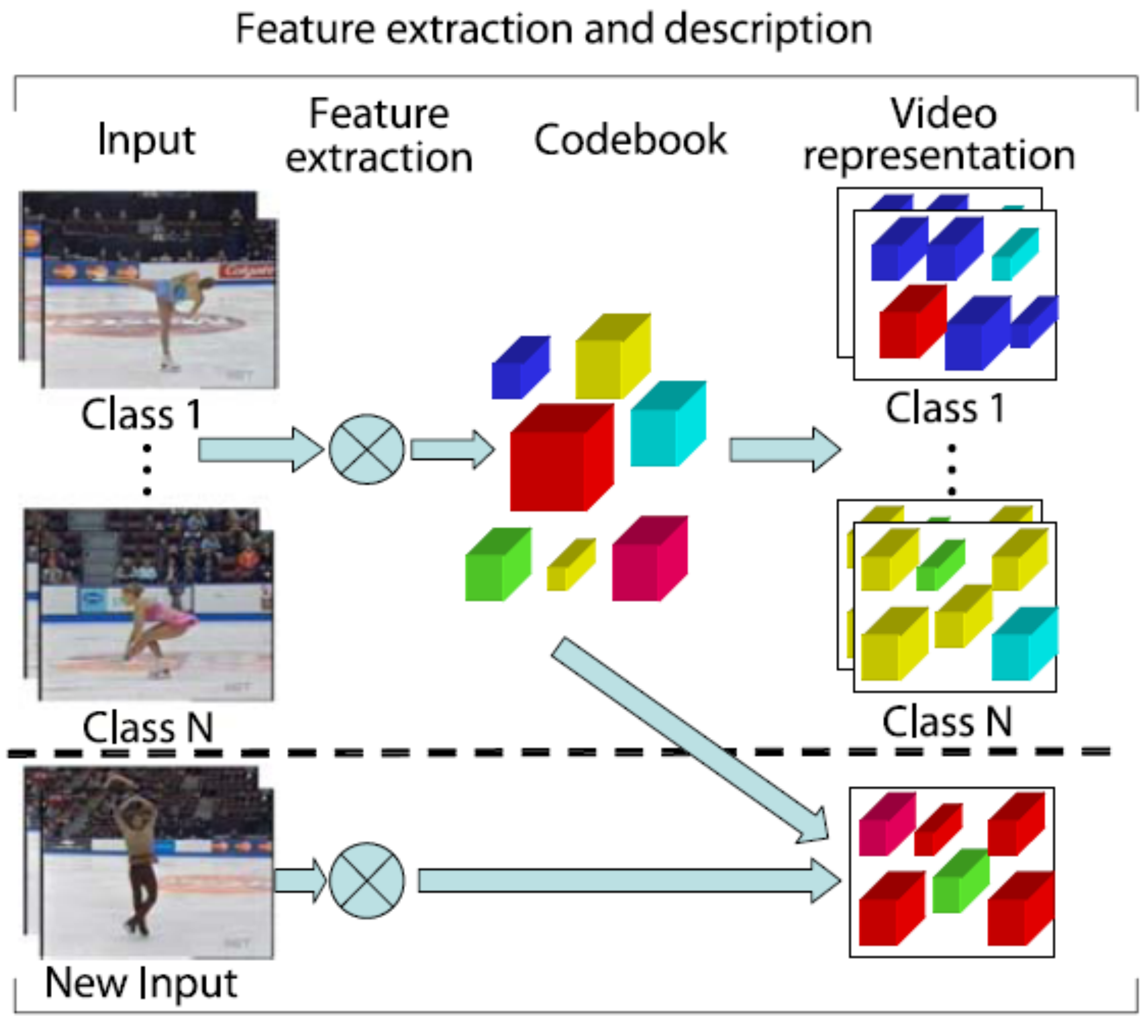
Application of pLSA: Action recognition

Space-time interest points



Juan Carlos Niebles, Hongcheng Wang and Li Fei-Fei,
[Unsupervised Learning of Human Action Categories Using Spatial-Temporal Words](), IJCV 2008.

Application of pLSA: Action recognition



Juan Carlos Niebles, Hongcheng Wang and Li Fei-Fei,
Unsupervised Learning of Human Action Categories Using Spatial-Temporal Words, IJCV 2008.

pLSA model

$$\underbrace{p(w_i \mid d_j)}_{\substack{\text{Probability of word i} \\ \text{in video j} \\ \text{(known)}}} = \sum_{k=1}^{K} \underbrace{p(w_i \mid z_k)}_{\substack{\text{Probability of} \\ \text{word i given} \\ \text{topic k} \\ \text{(unknown)}}} \underbrace{p(z_k \mid d_j)}_{\substack{\text{Probability of} \\ \text{topic k given} \\ \text{video j} \\ \text{(unknown)}}}$$

- w_i = spatial-temporal word
- d_j = video
- $n(w_i, d_j)$ = co-occurrence table
  (# of occurrences of word w_i in video d_j)
- z = topic, corresponding to an action

Action recognition example

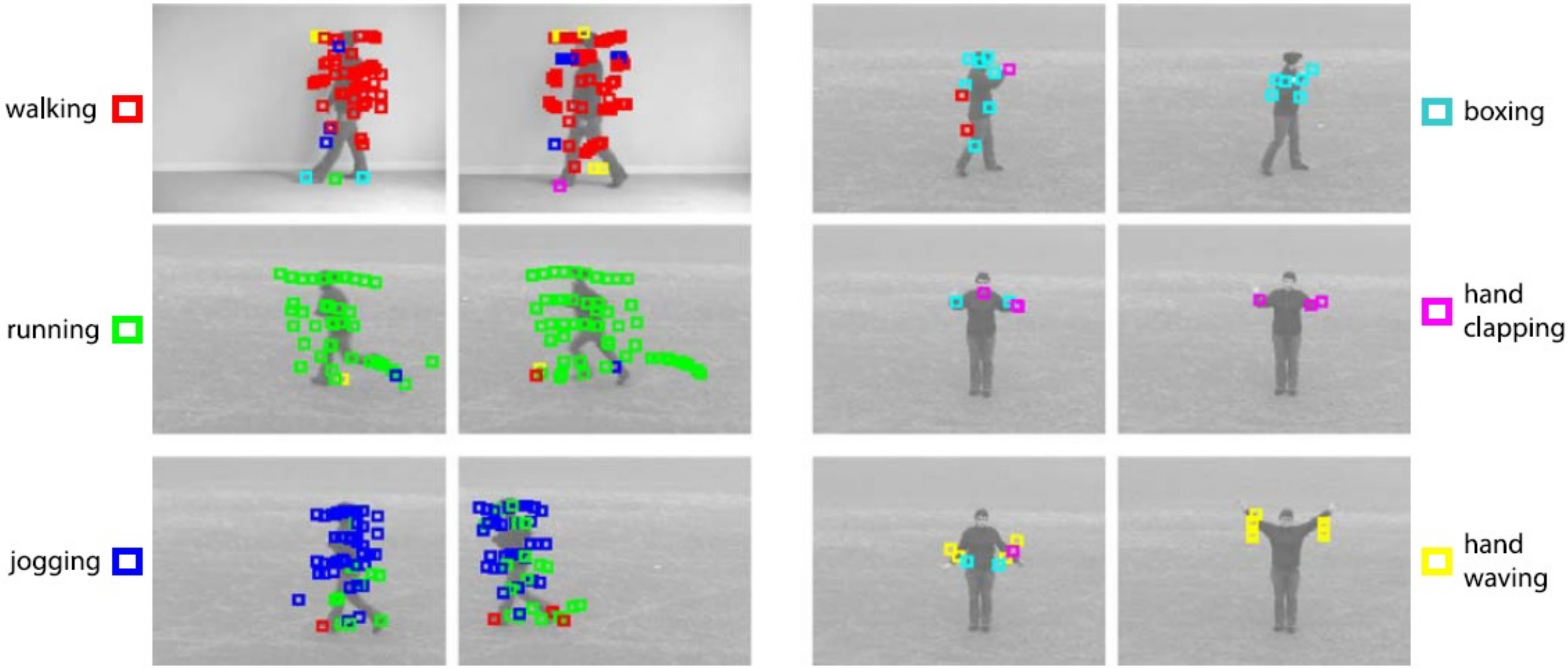
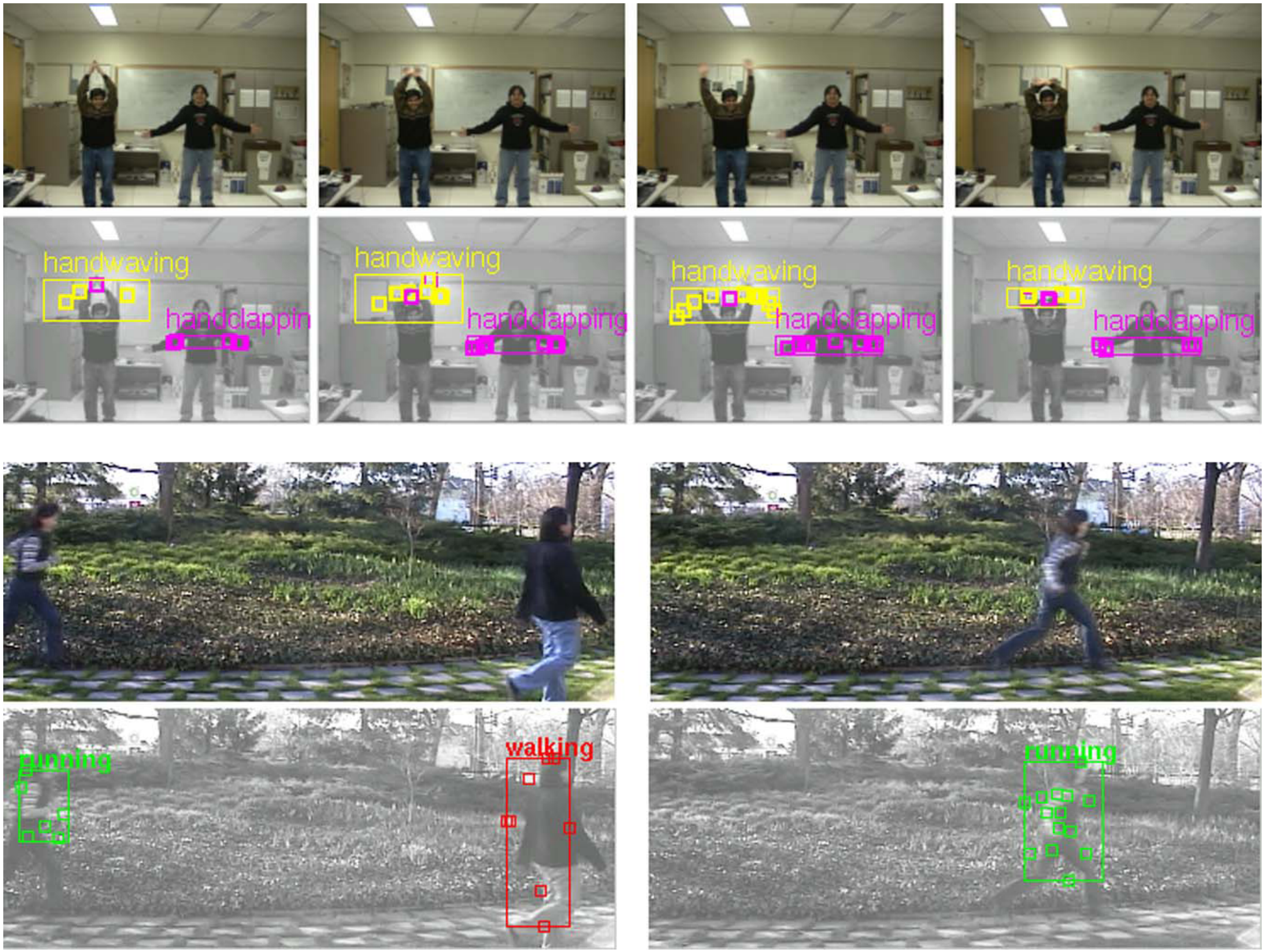
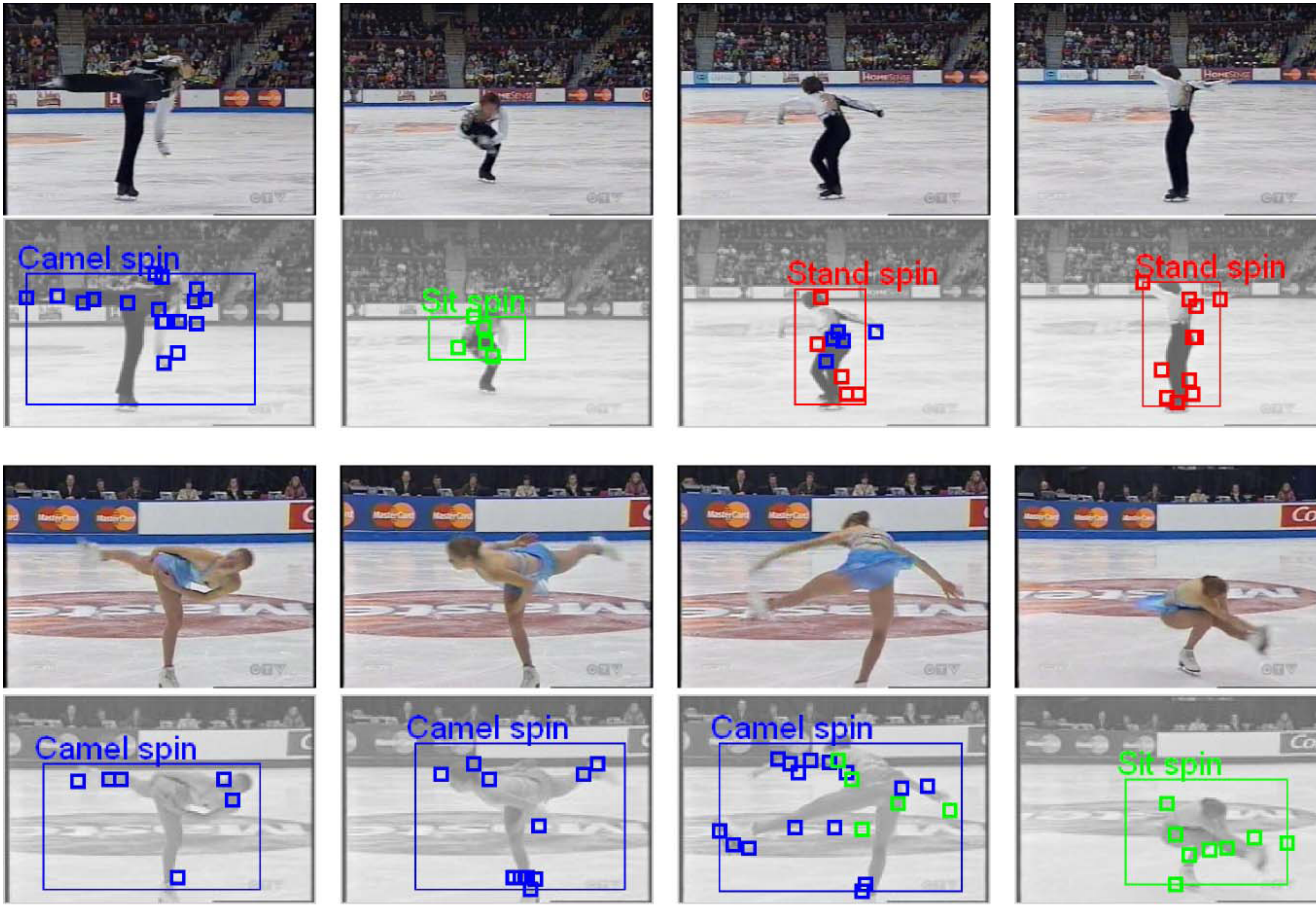


Fig. 10 Example frames from testing sequences in the KTH dataset. The spatial-temporal patches in each sequence are automatically colored according to action class that most likely generated its corresponding spatial-temporal word. Although some of the words are assigned to the wrong topic, most interest points are assigned to the correct action for each video. Consistently, the predicted action class corresponds to the actual ground truth. In addition, we usually observe that the second best ranked action class corresponds to a similar action: in the "jogging" example of the figure, the second best label is "running". The figure is best viewed in color and with PDF magnification

Multiple Actions



Multiple Actions



Summary: Generative models

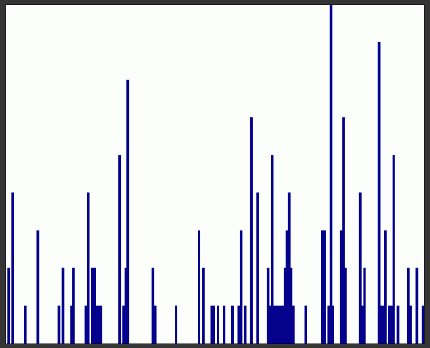
- Naïve Bayes
  - *Unigram models* in document analysis
  - Assumes conditional independence of words given class
  - Parameter estimation: frequency counting
- Probabilistic Latent Semantic Analysis
  - Unsupervised technique
  - Each document is a mixture of topics (image is a mixture of classes)
  - Can be thought of as matrix decomposition
  - Parameter estimation: Expectation-Maximization

Adding spatial information

- Computing bags of features on sub-windows of the whole image
- Using codebooks to vote for object position
- Generative part-based models

Spatial pyramid representation

- Extension of a bag of features
- Locally orderless representation at several levels of resolution



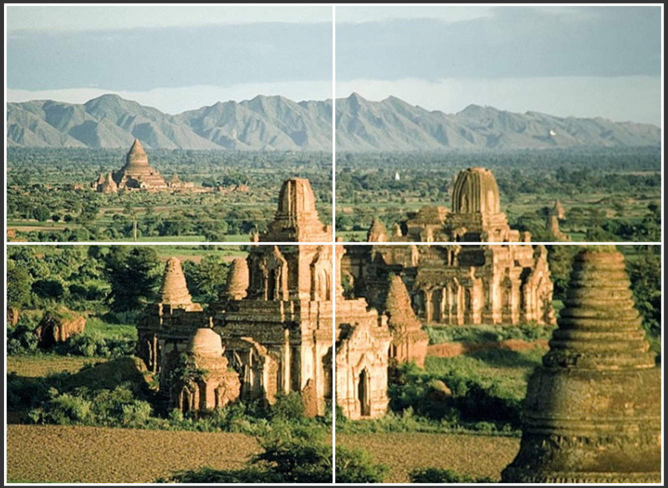
level 0

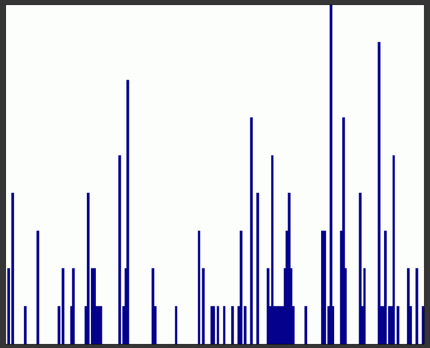
Lazebnik, Schmid & Ponce (CVPR 2006)

Slide: S. Lazebnik

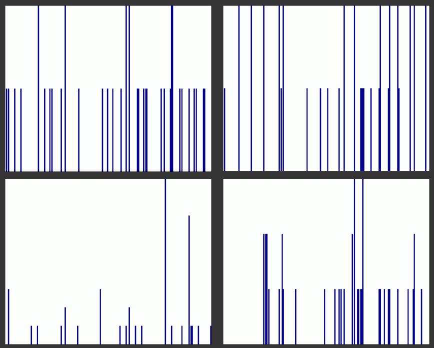
Spatial pyramid representation

- Extension of a bag of features
- Locally orderless representation at several levels of resolution





level 0



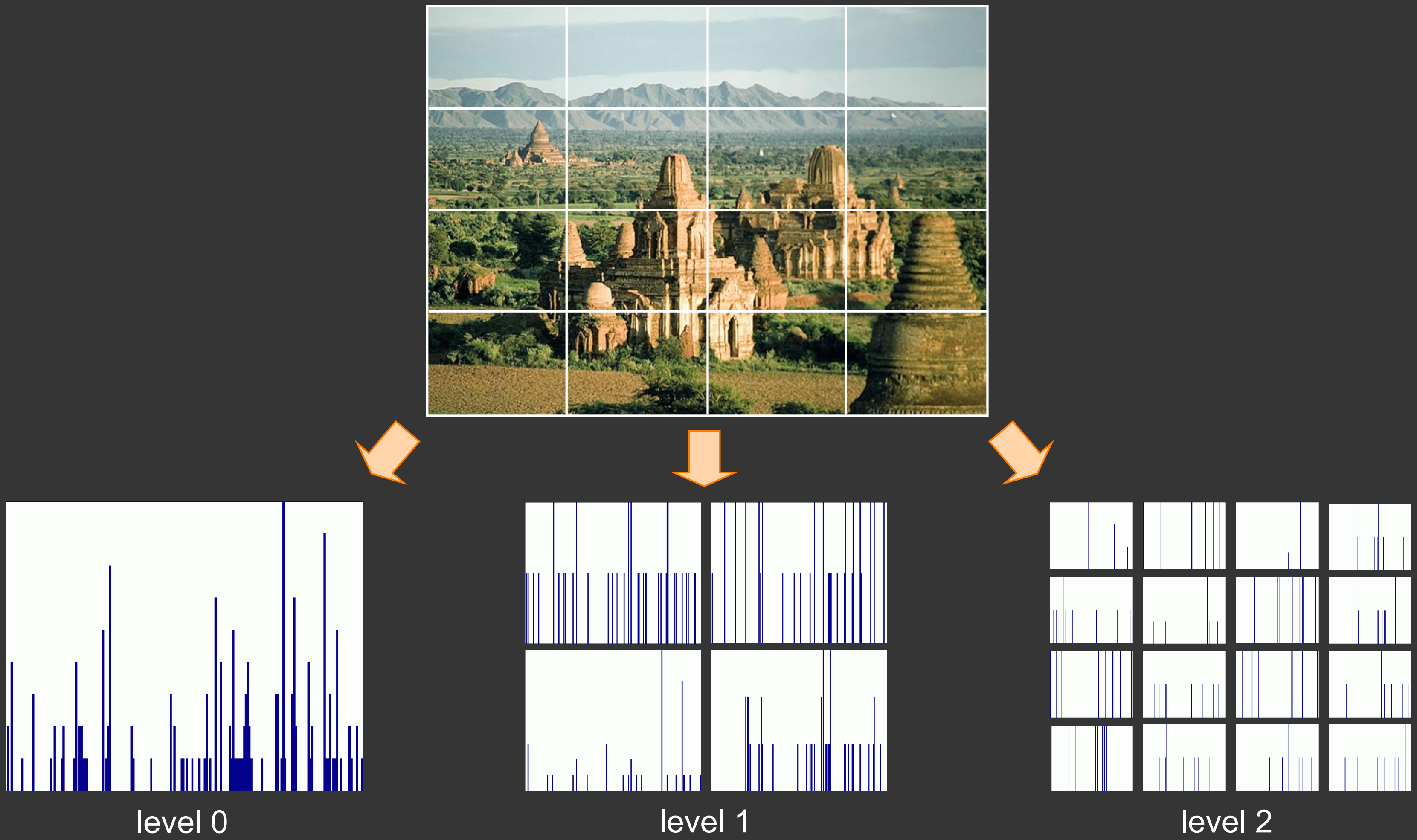
level 1

Lazebnik, Schmid & Ponce (CVPR 2006)

Slide: S. Lazebnik

Spatial pyramid representation

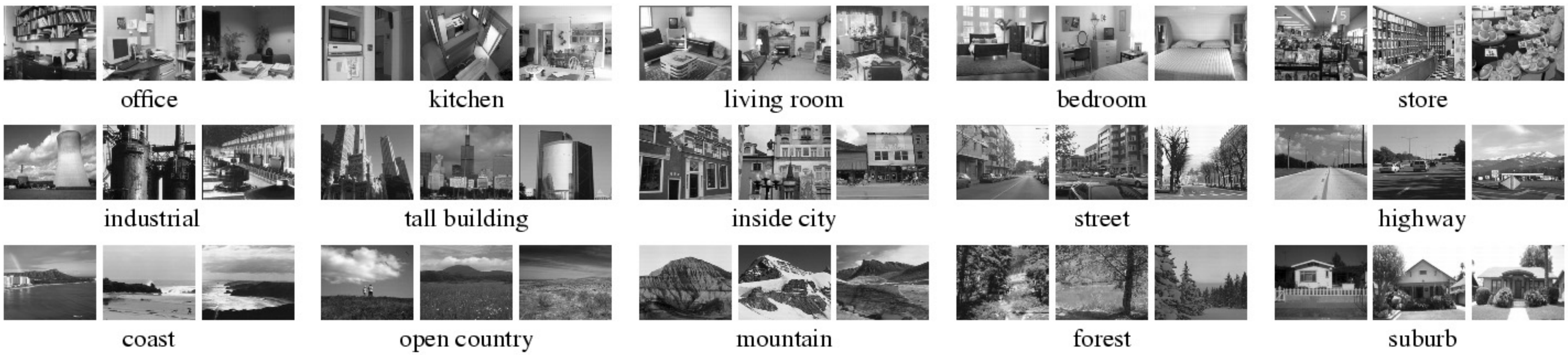
- Extension of a bag of features
- Locally orderless representation at several levels of resolution



Lazebnik, Schmid & Ponce (CVPR 2006)

Slide: S. Lazebnik

Scene category dataset



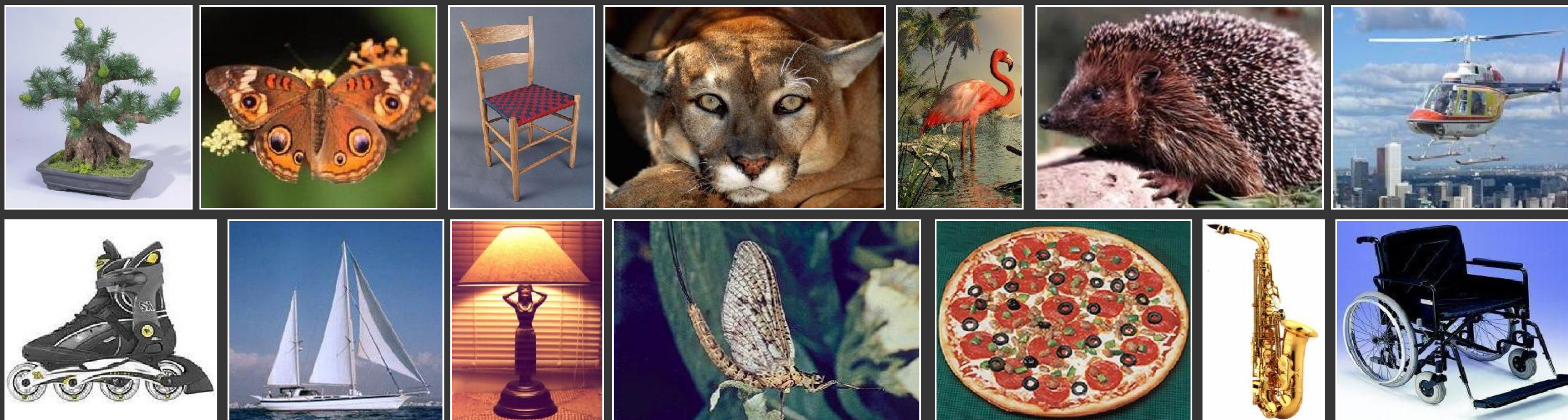
Multi-class classification results
(100 training images per class)

	Weak features (vocabulary size: 16)		Strong features (vocabulary size: 200)	
Level	Single-level	Pyramid	Single-level	Pyramid
0 (1×1)	45.3 ±0.5		72.2 ±0.6	
1 (2×2)	53.6 ±0.3	56.2 ±0.6	77.9 ±0.6	79.0 ±0.5
2 (4×4)	61.7 ±0.6	64.7 ±0.7	79.4 ±0.3	**81.1** ±0.3
3 (8×8)	63.3 ±0.8	**66.8** ±0.6	77.2 ±0.4	80.7 ±0.3

Slide: S. Lazebnik

Caltech101 dataset

http://www.vision.caltech.edu/Image_Datasets/Caltech101/Caltech101.html



Multi-class classification results (30 training images per class)

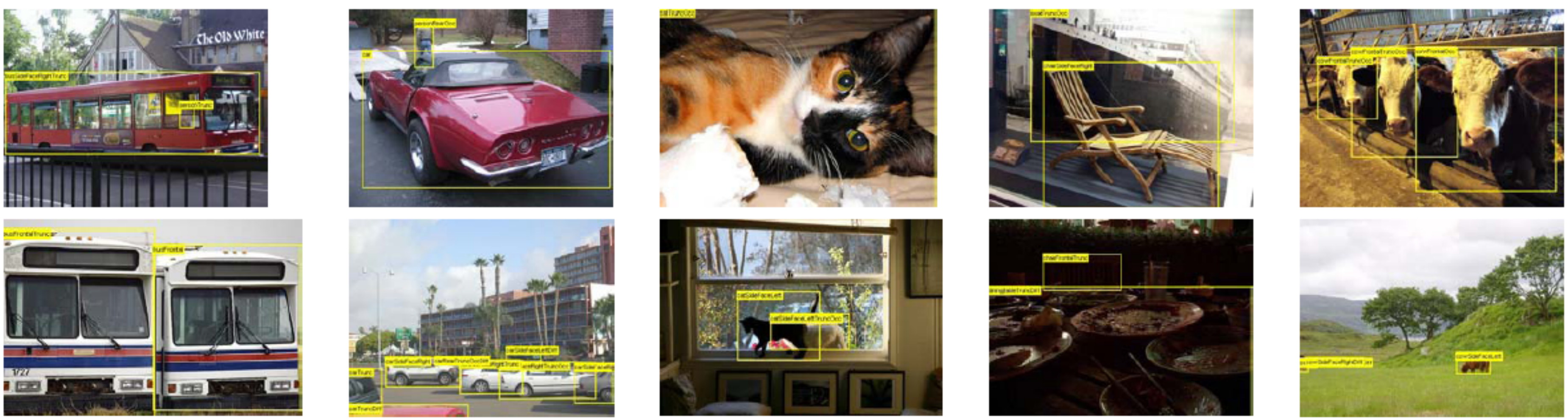
	Weak features (16)		Strong features (200)	
Level	Single-level	Pyramid	Single-level	Pyramid
0	15.5 ±0.9		41.2 ±1.2	
1	31.4 ±1.2	32.8 ±1.3	55.9 ±0.9	57.0 ±0.8
2	47.2 ±1.1	49.3 ±1.4	63.6 ±0.9	**64.6** ±0.8
3	52.2 ±0.8	**54.0** ±1.1	60.3 ±0.9	64.6 ±0.7

Slide: S. Lazebnik

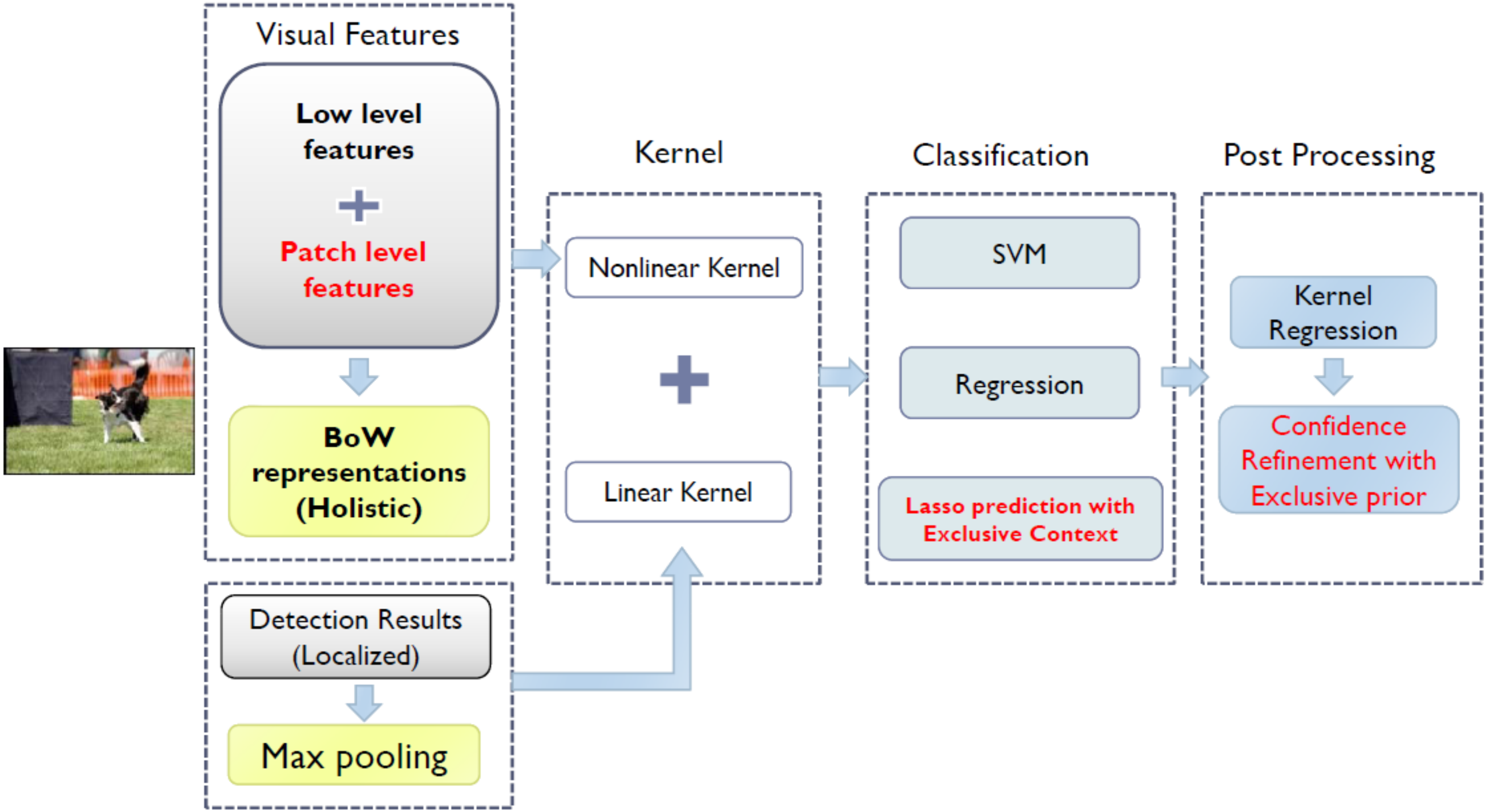
Examples from PASCAL VOC Challenge 2010

Aeroplane | Bicycle | Bird | Boat | Bottle

Bus | Car | Cat | Chair | Cow



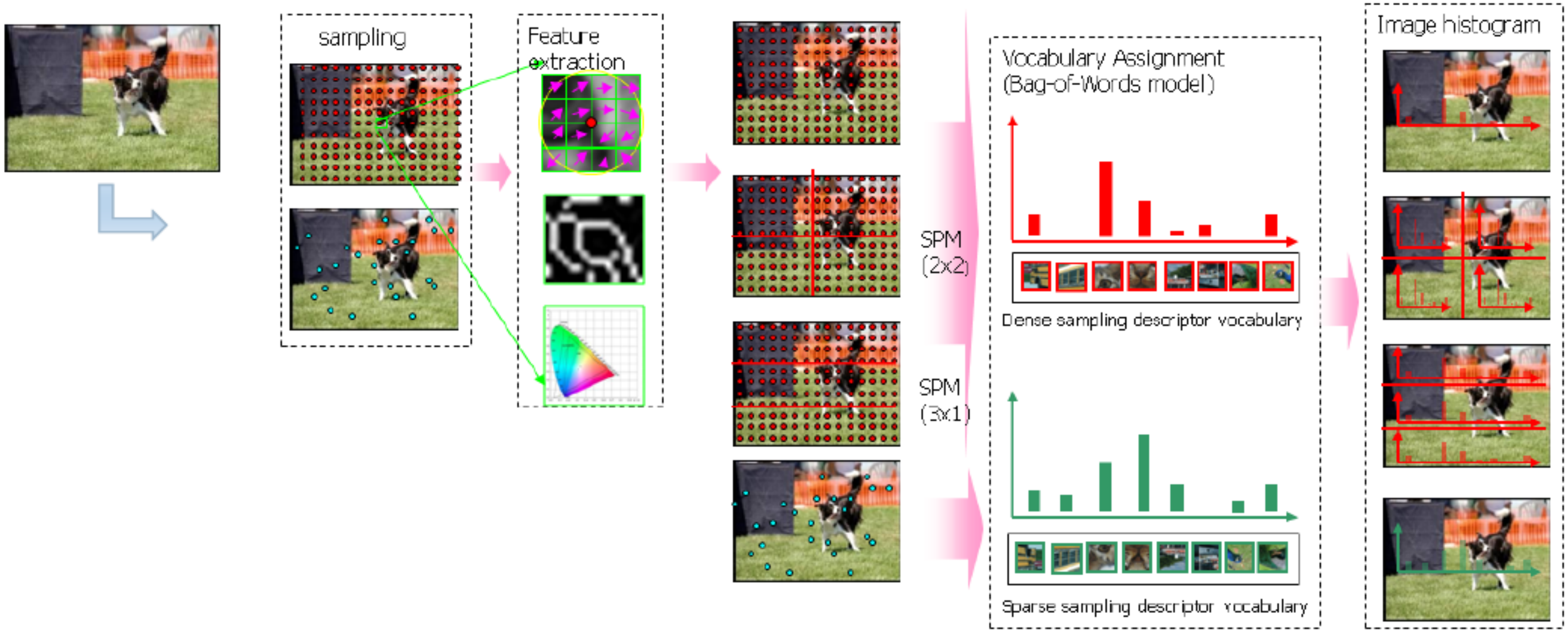
Framework



Boosting Classification with Exclusive Context, Yan et al. 2010

Low Level Features

- Low level features: SIFT and its variants, LBP, HOG.
- Dense sampling and interest point detector;
- Represented as Bags of Words;

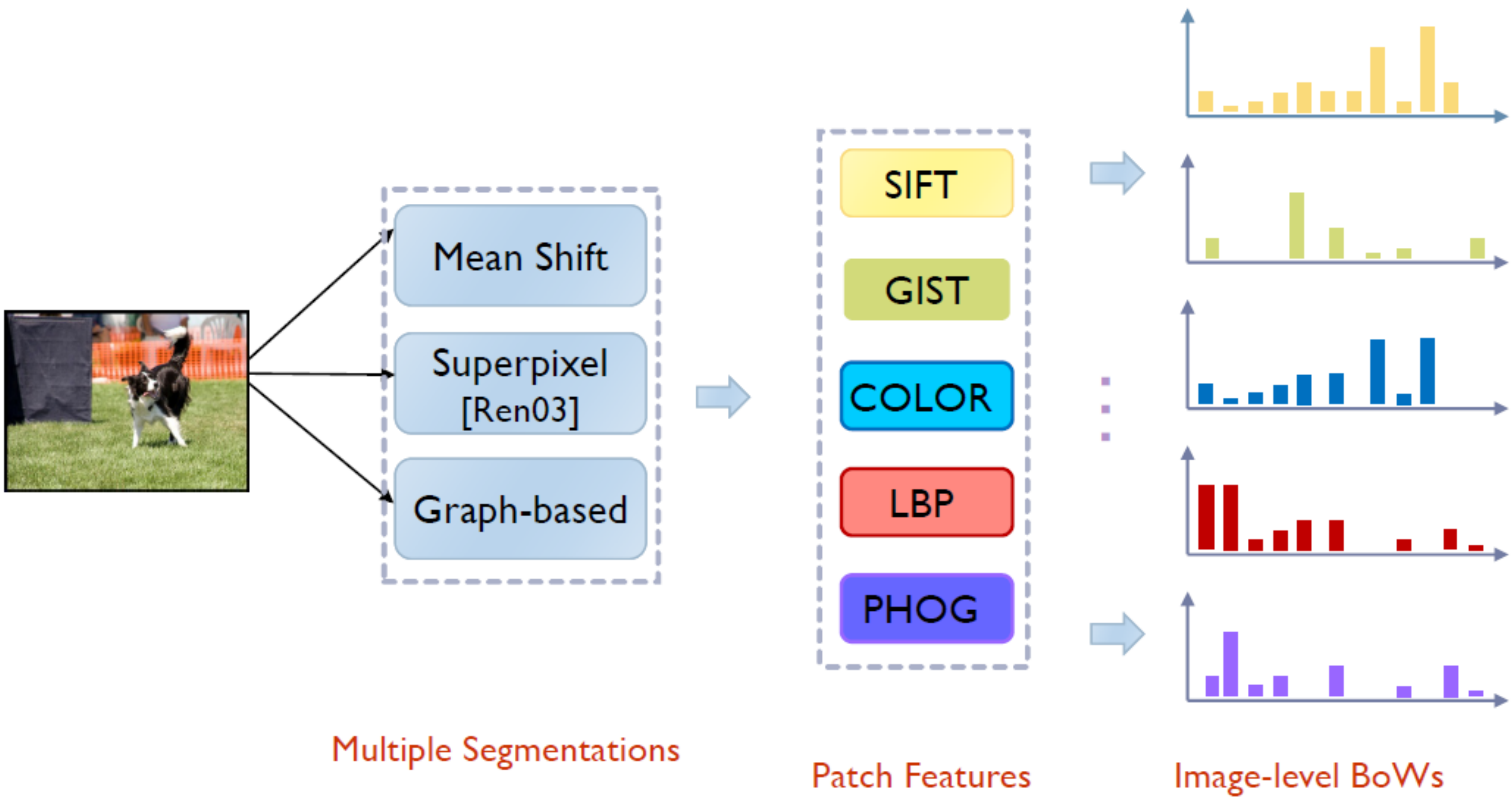


NUS National University of Singapore

Panasonic ideas for life

Boosting Classification with Exclusive Context, Yan et al. 2010

Patch Level Features



NUS National University of Singapore

Panasonic ideas for life

Boosting Classification with Exclusive Context, Yan et al. 2010

The results

	SVM	Exclusive	Fusing	Our Best	Other's Best
aeroplane	91.9	91.3	93	93	93.3
bicycle	77.1	77	79	79	77
bird	69.5	70	71.6	71.6	69.9
boat	74.7	75.6	77.8	77.8	77.2
bottle	52.5	50.7	54.3	54.3	53.7
bus	84.3	83.2	85.2	85.2	85.9
car	77.3	77.1	78.6	78.6	80.4
cat	76.2	75.4	78.8	78.8	79.4
chair	63	62.5	64.5	64.5	62.9
cow	63.5	62.6	64	64	66.2
diningtable	62.9	62.7	62.7	62.9	61.1
dog	65	64.6	69.6	69.6	71.1
horse	79.5	77.9	82	82	76.7
motorbike	83.2	81.8	84.4	84.4	81.7
person	91.2	91.1	91.6	91.6	90.2
pottedplant	45.5	44.8	48.6	48.6	53.3
sheep	65.4	64.2	64.9	65.4	66.3
sofa	55	53.2	59.6	59.6	58
train	87	86.3	89.4	89.4	87.5
tvmonitor	77.2	77.1	76.4	77.2	76.2
MAP	72.095	71.455	73.8		

NUS National University of Singapore

Panasonic ideas for life

Boosting Classification with Exclusive Context, Yan et al. 2010