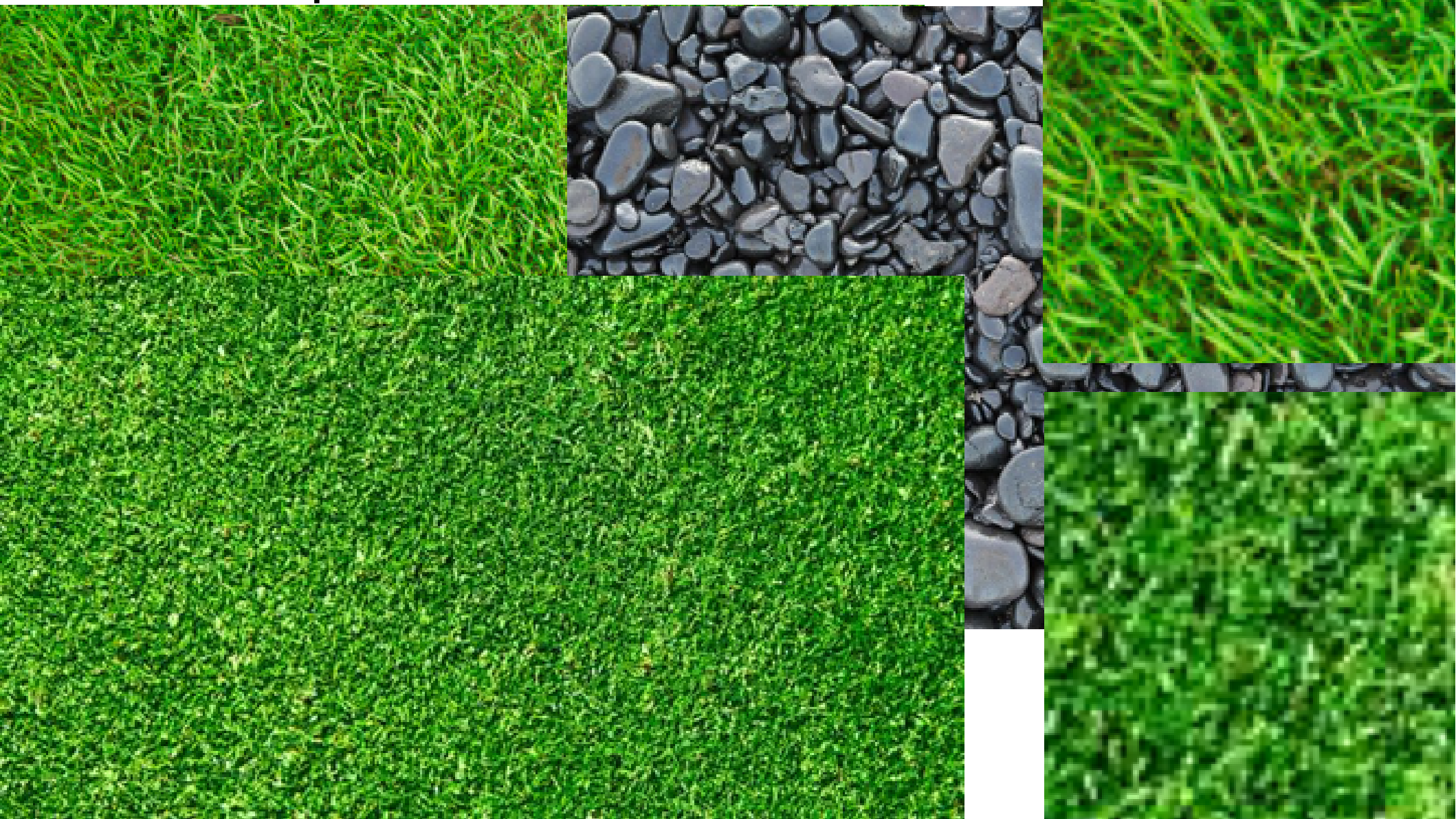
Computer Vision

Recognizing images, objects, scenes

Suyash P. Awate

Texture

- Statistics of appearance of patches
  - Colors patterns = texture

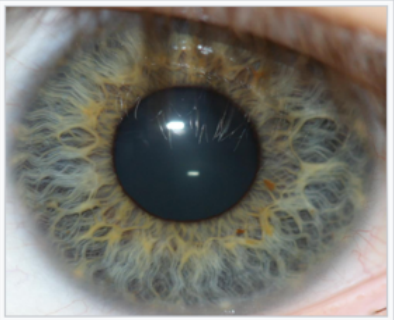
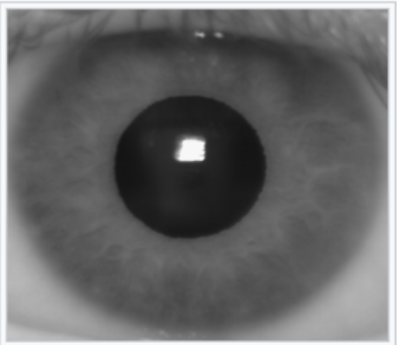
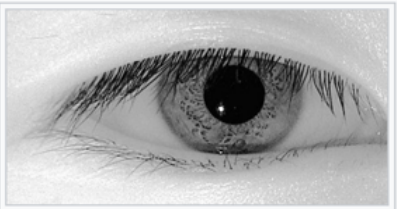


Texture Classification

Texture Modeli

SPIE

Daugman filed for a patent for his iris-recognition algorithm in 1991, while at the Univ. of Cambridge.

Visible wavelength iris image	Near infrared (NIR) version	NIR imaging extracts structure
		
Visible light reveals rich pigmentation details of an Iris by exciting melanin, the main colouring component in the iris.	Pigmentation of the iris is invisible at longer wavelengths in the NIR spectrum	Even "dark brown" eyes reveal rich iris texture in the NIR band, and most corneal specular reflections can be blocked.

600 million citizens of India are now enrolled with biometric ID

John Daugman

The most ambitious biometric deployment in history, to enroll the iris patterns and other identifying data of all 1.2 billion Indian citizens in three years, has now passed its halfway mark.

Biometric patte
and engineerin
rapid determin
and matching
technologies th
processing, cor
tistics, pattern
around the wo

John Daugman
University of Cambridge
Cambridge, United Kingdom

John Daugman is professor of Computer Vision and Pattern Recognition at Cambridge University. He is the inventor of automatic iris recognition, and he serves as chief scientist for this technology with the Morpho division of the French defense, aerospace, and security group SAFRAN. His awards include the Order of the British Empire (OBE), and induction into the US National Inventors Hall of Fame.

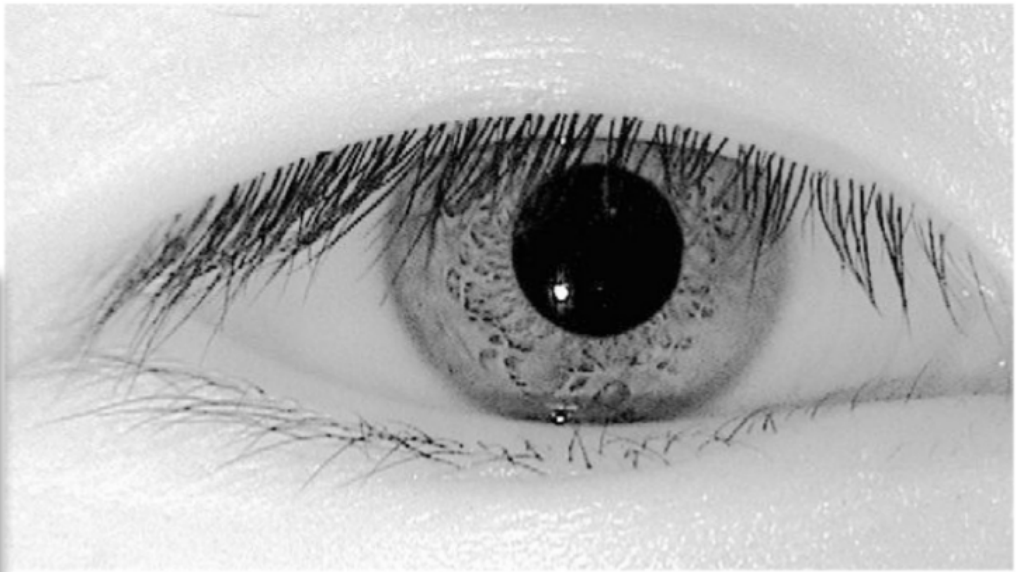
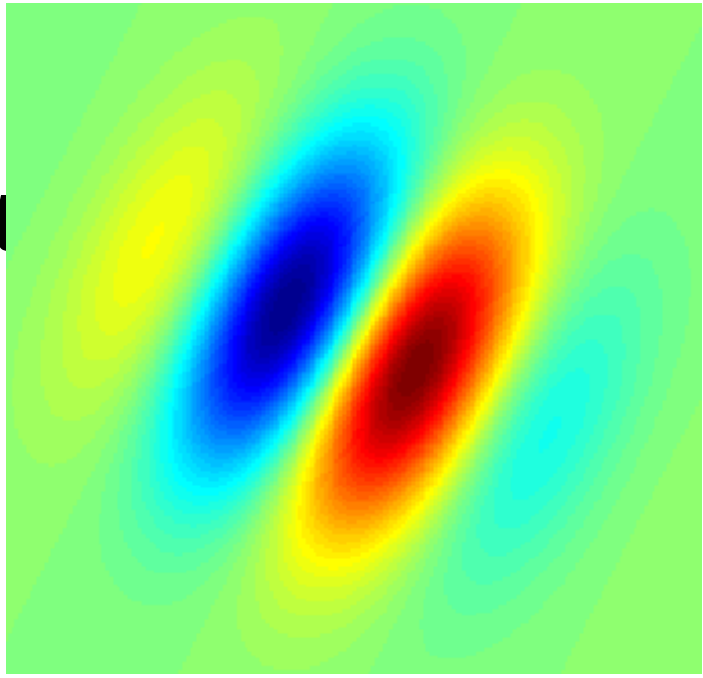


Figure 1. *Complex texture of a darkly pigmented iris when it is imaged in the near-IR band (NIR, 700–900nm).*

Texture Modeling



- Filter bank to model / represent text
  - Gabor filters
  - Each filter = sinusoid .* Gaussian
  - Gabor filter bank = collection of filters
    - Various patterns, scales, orientations
    - Lambda = wavelength; theta = orientation of normal to “stripes”; psi = phase; gamma = spatial aspect ratio

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \cos\left(2\pi \frac{x'}{\lambda} + \psi\right)$$

where

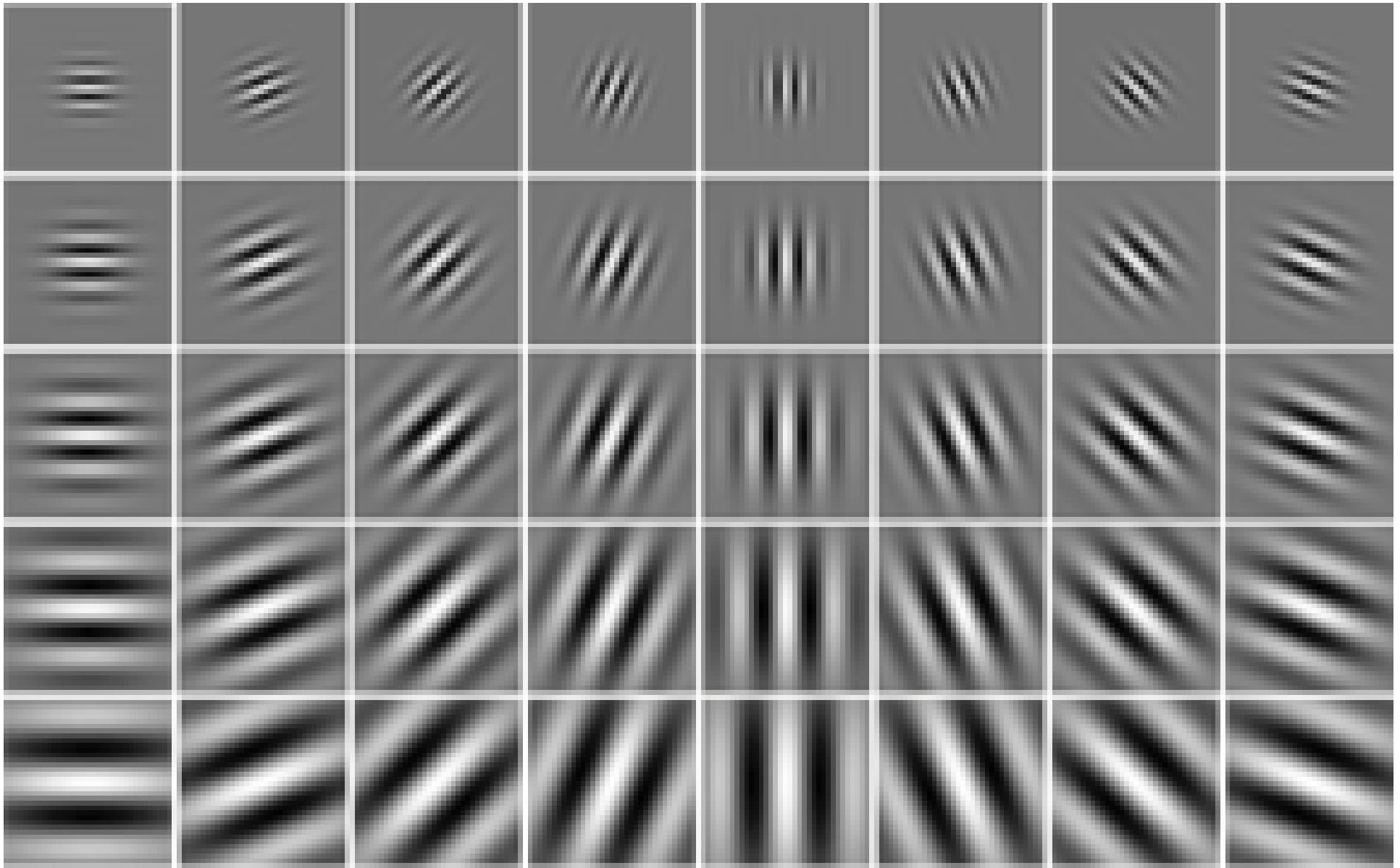
$$x' = x\cos\theta + y\sin\theta$$

and

$$y' = -x\sin\theta + y\cos\theta$$

Texture Modeling

- Filter bank to model / represent texture
  - Gabor filters



Texture Modeling

- Gabor filter

Complex

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \exp\left(i\left(2\pi\frac{x'}{\lambda} + \psi\right)\right)$$

Real

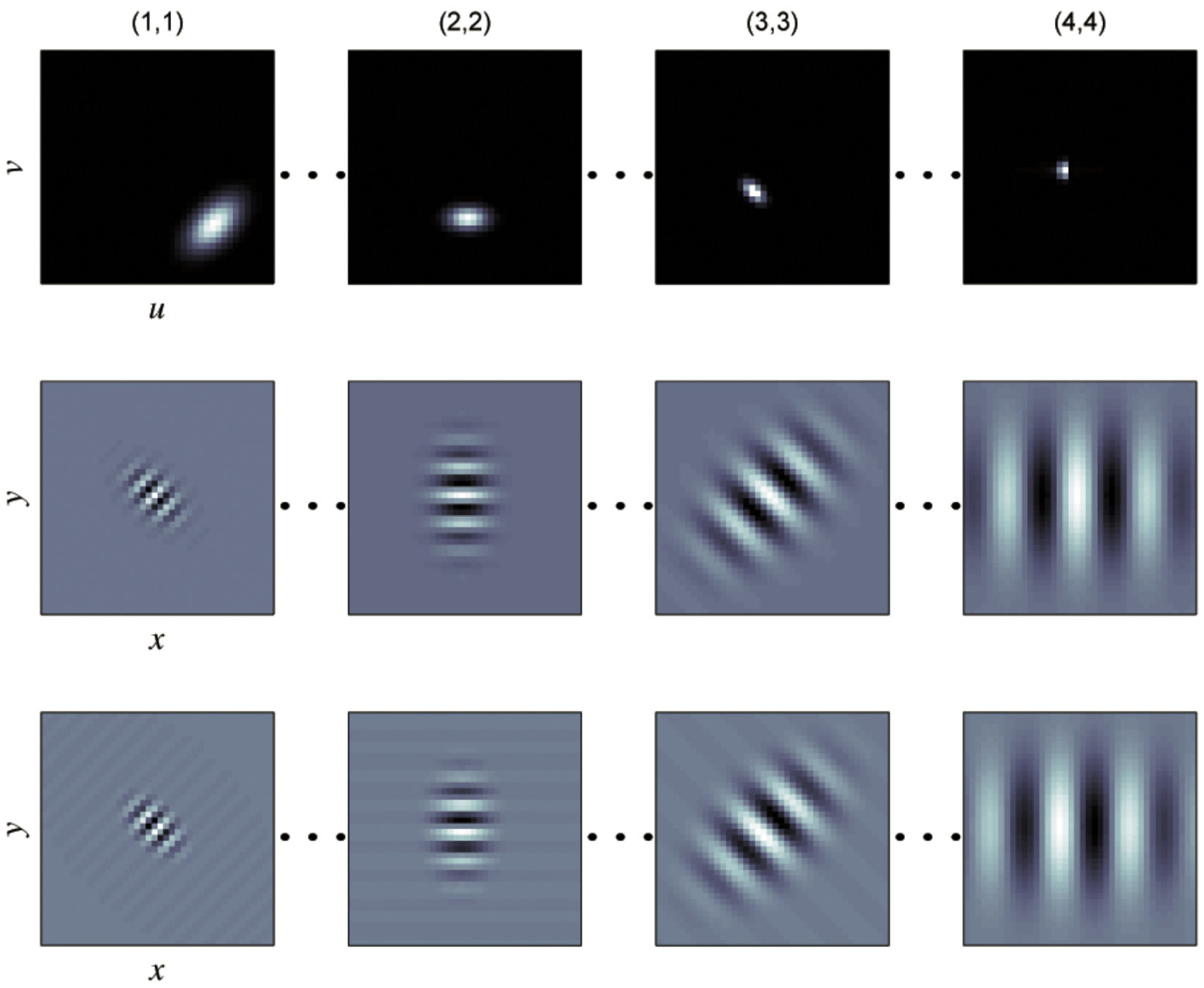
$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \cos\left(2\pi\frac{x'}{\lambda} + \psi\right)$$

Imaginary

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \sin\left(2\pi\frac{x'}{\lambda} + \psi\right)$$

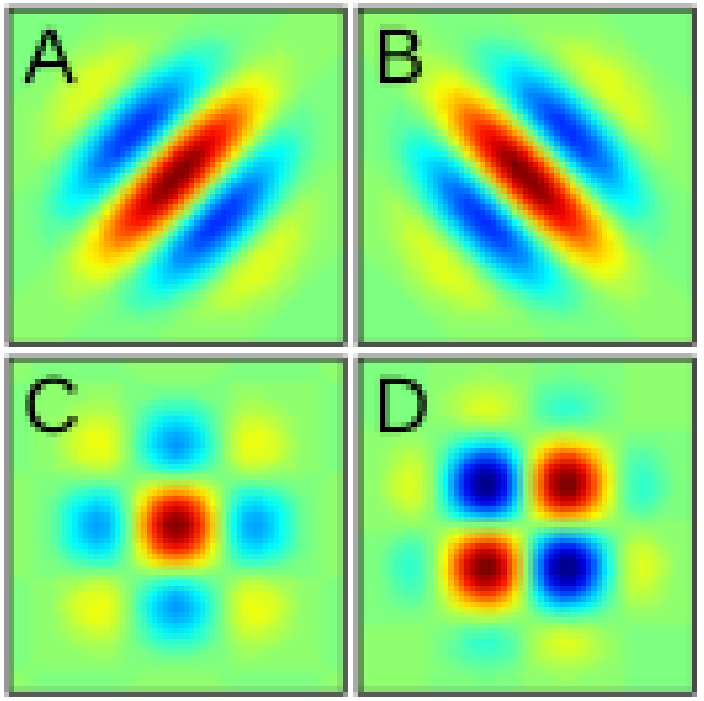
Texture Modeling

- Gabor filters in Fourier frequency domain

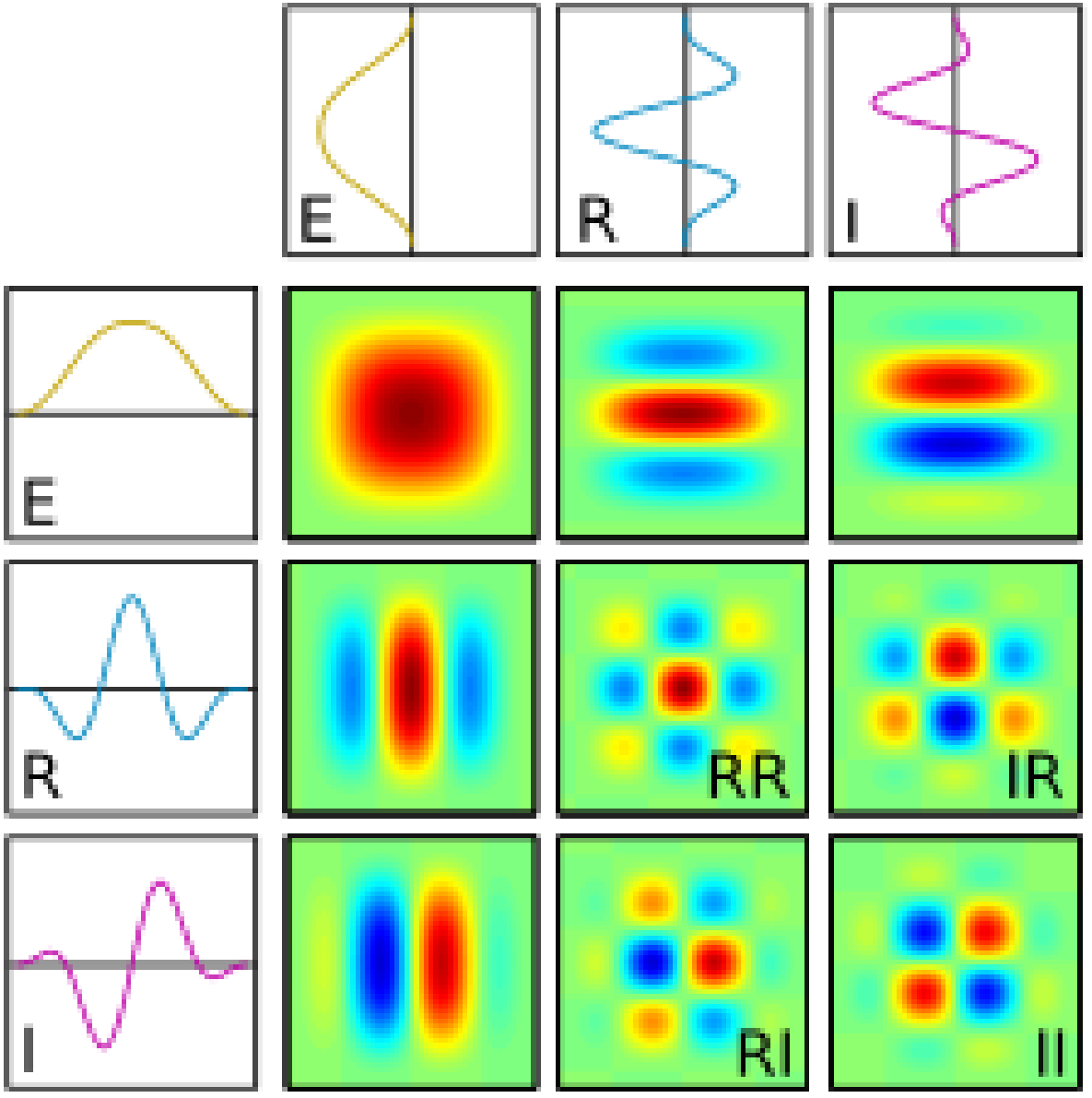


Texture Modeling

- 2D Gabor filters **not** separable



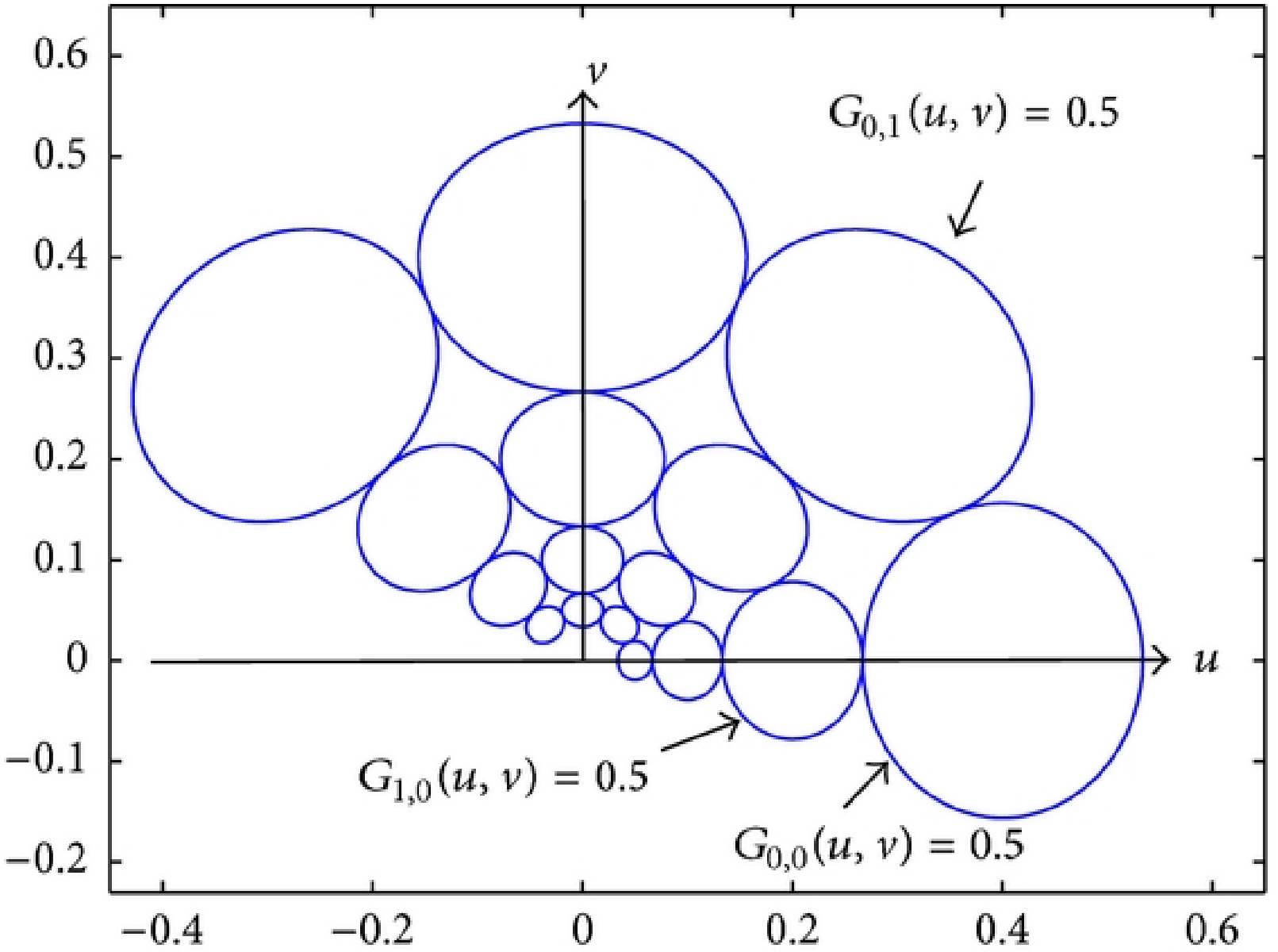
2D-Gabor filters: A, B
Separable filters: C, D



Separable combinations of 1D Gabor filters

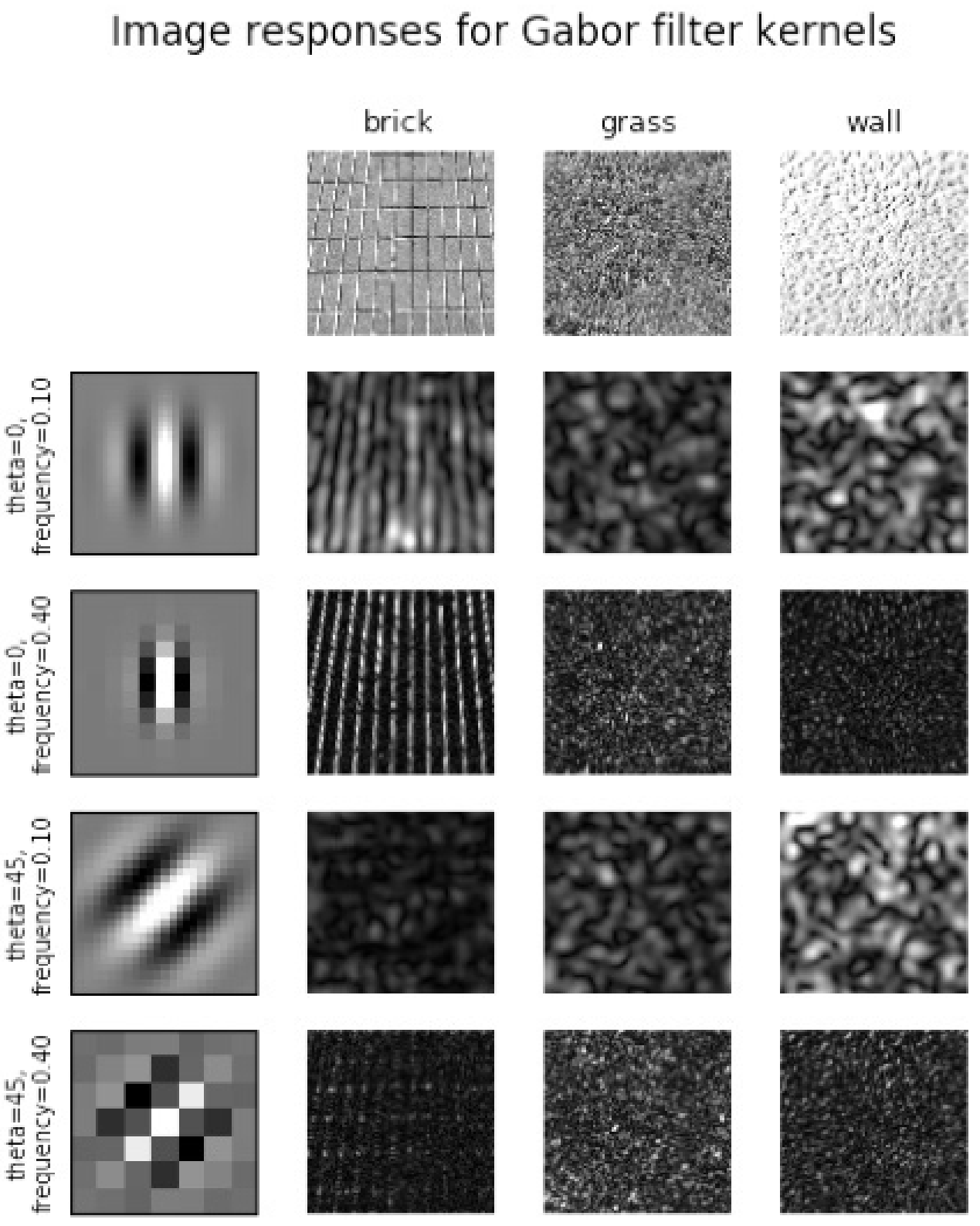
Texture Modeling

- Designing the Gabor filter bank in frequency domain



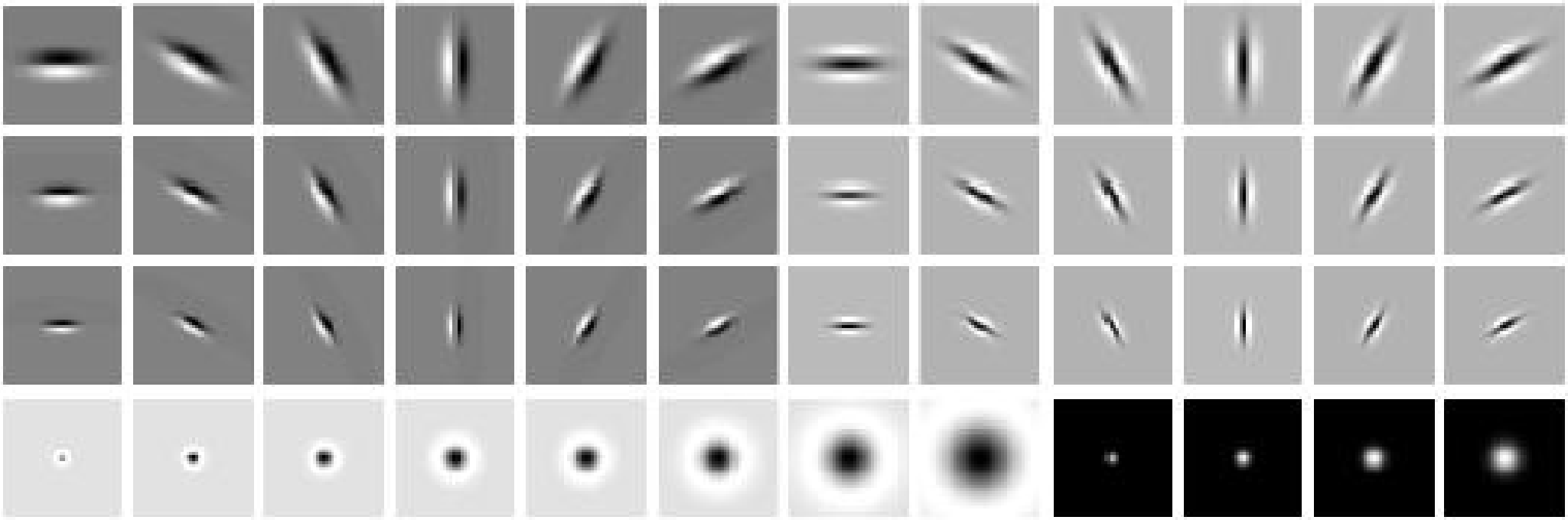
Texture Modeli

- Filter bank to model / represent texture
  - Gabor filter response at each pixel



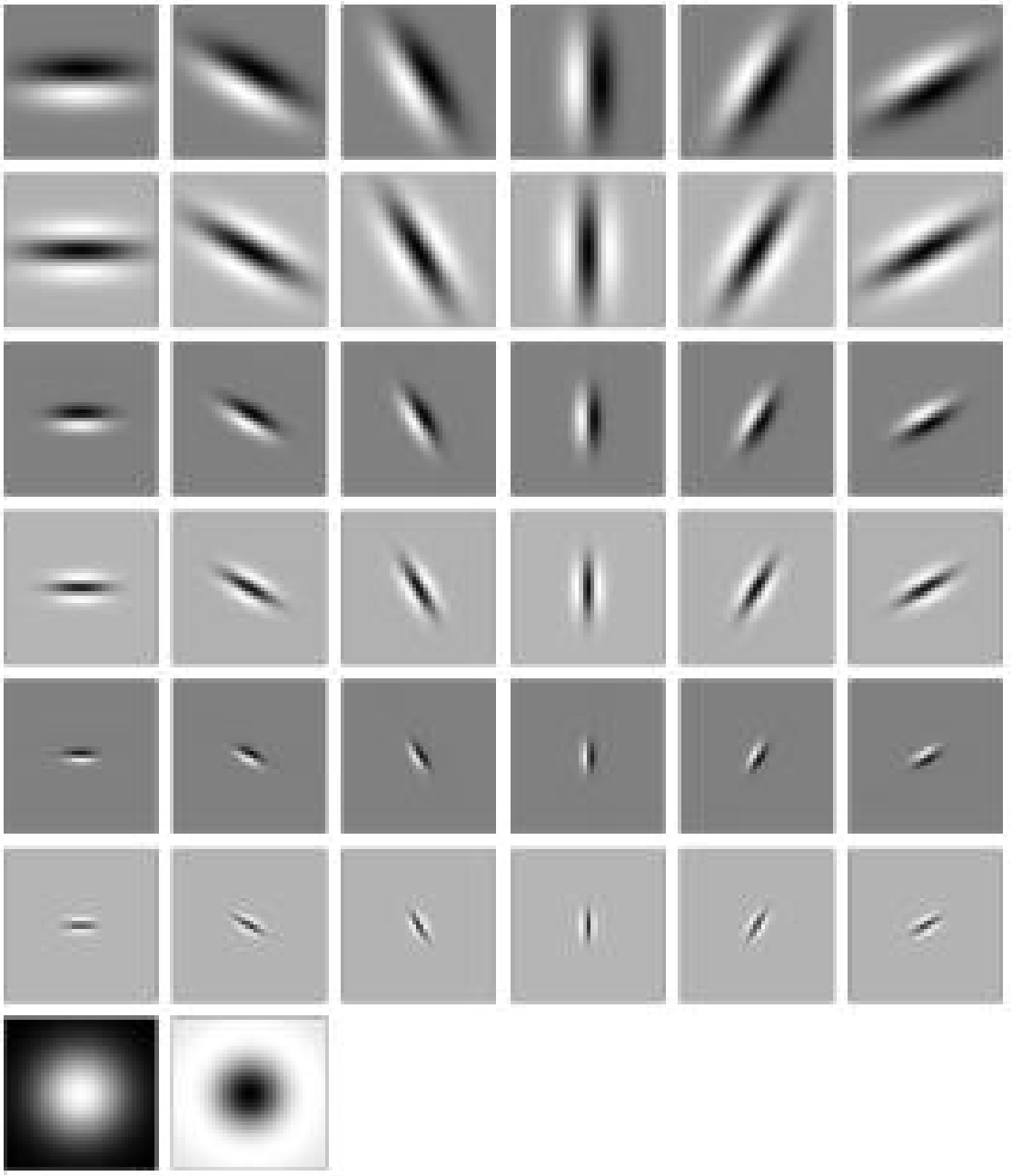
Texture Modeling

- Filter bank to model / represent texture
  - [Leung and Malik 2001 IJCV]
    - “Edge” (d/dx G(x,y)) filters (3 scales, 6 orientations)
    - “Bar” (d^2/dx^2 G(x,y)) filters (3 scales, 6 orientations)
    - LoG (8 scales)
    - Gaussians (4 scales)



Texture Modeling

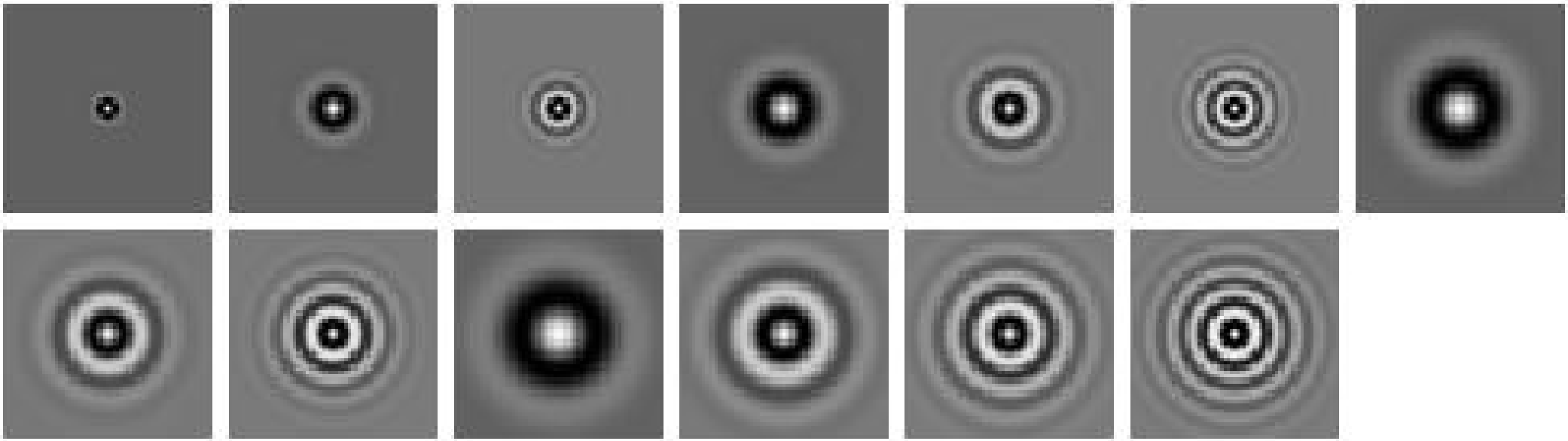
- Filter bank to model / represent texture
  - MR8 [ECCV 2002]
    - MR = Maximum response
    - “Edge” filter
      - 3 scales
      - 6 orientations
    - “Bar” filter
      - 3 scales
      - 6 orientations
    - Gaussian
    - LoG



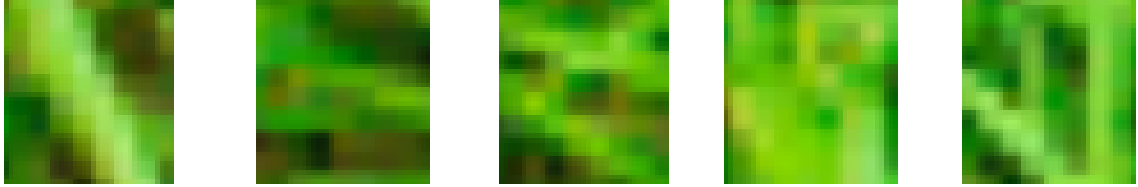
Texture Modeling

- Filter bank to model / represent texture
  - [Schmidt 2001 IEEE CVPR]
    - 13 rotationally invariant (isotropic) filters

$$F(r,\sigma,\tau) = F_0(\sigma,\tau) + \cos\left(\frac{\pi\tau r}{\sigma}\right) e^{-\frac{r^2}{2\sigma^2}}$$



Texture Modeling



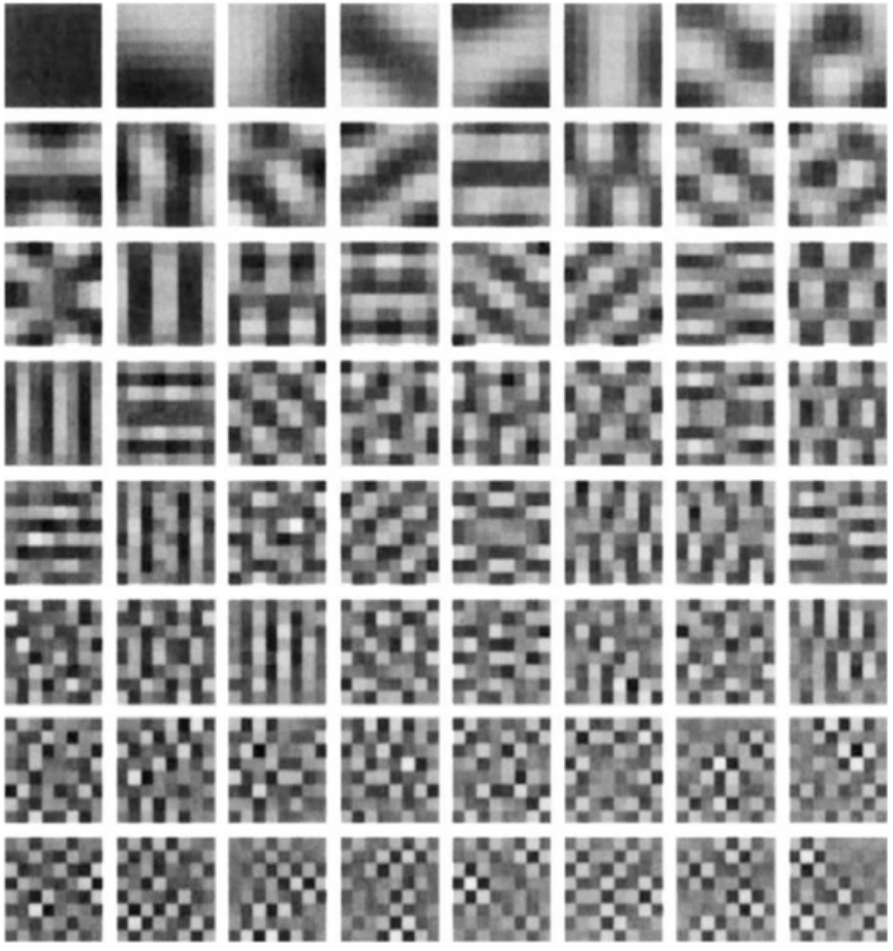
- Filter bank to model / represent texture
  - Data-driven texture [1996 Nature]

Emergence of simple-cell receptive field properties by learning a sparse code for natural images

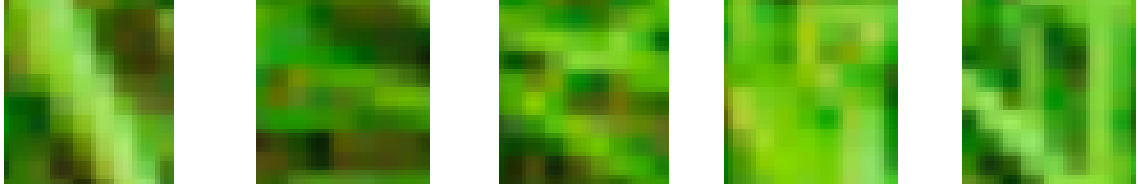
Bruno A. Olshausen* & David J. Field

Department of Psychology, Uris Hall, Cornell University, Ithaca, New York 14853, USA

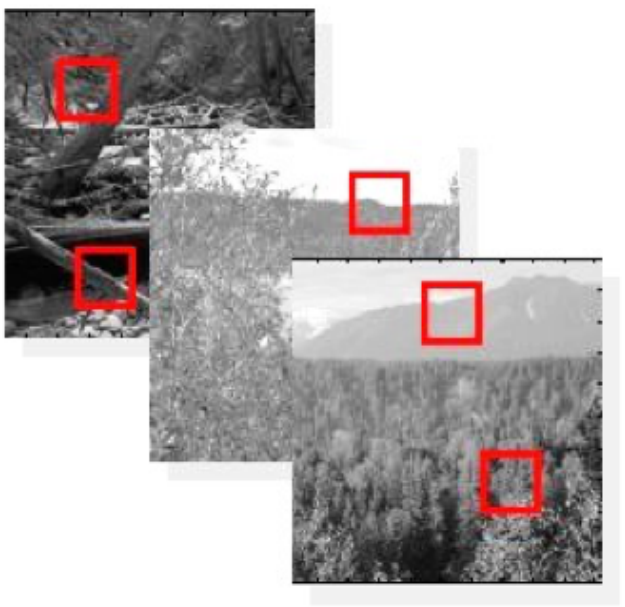
THE receptive fields of simple cells in mammalian primary visual cortex can be characterized as being spatially localized, oriented[1–4] and bandpass (selective to structure at different spatial scales), comparable to the basis functions of wavelet transforms[5,6]. One approach to understanding such response properties of visual neurons has been to consider their relationship to the statistical structure of natural images in terms of

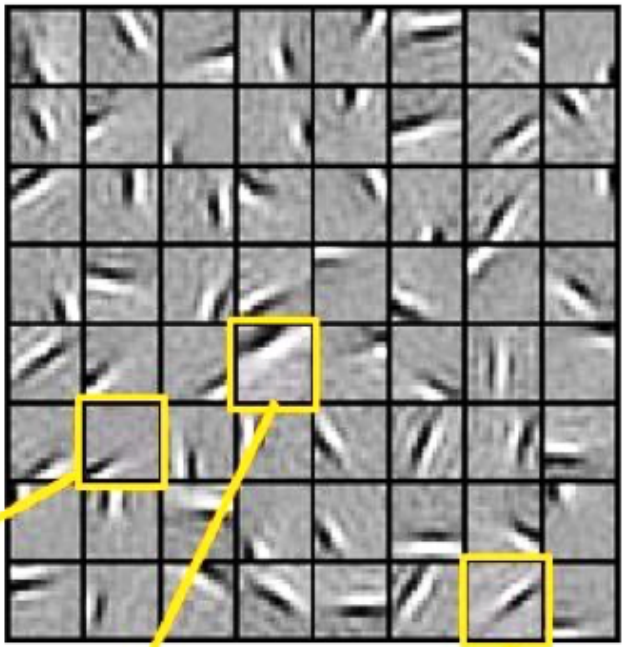


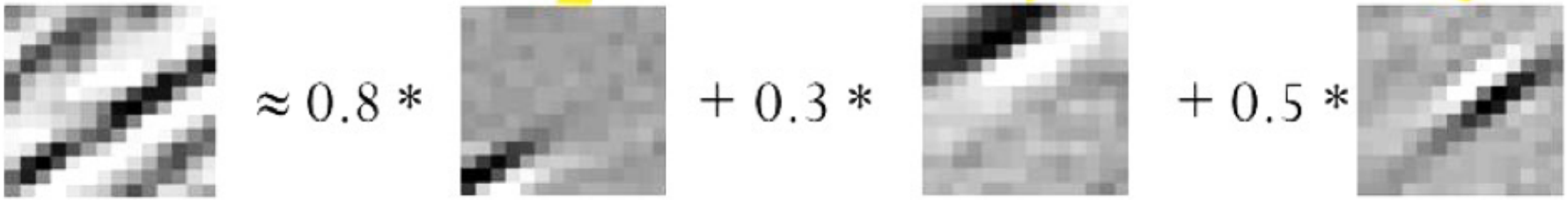
Texture Modeling



- Filter bank to model / represent texture
  - Data-driven texture





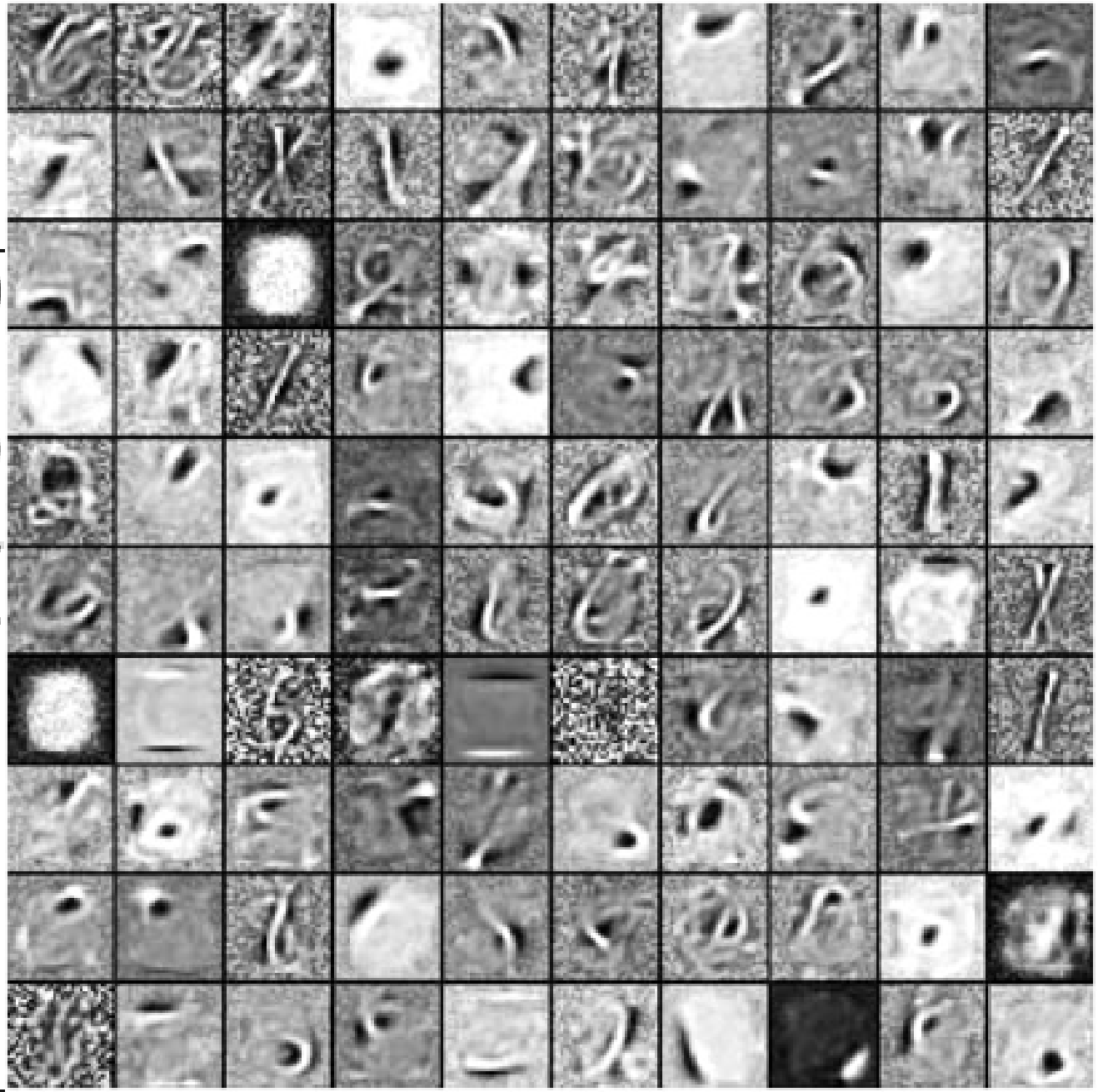


$[a_1, \ldots, a_{64}] = [0, 0, \ldots, 0, \mathbf{0.8}, 0, \ldots, 0, \mathbf{0.3}, 0, \ldots, 0, \mathbf{0.5}, 0]$

(feature representation)

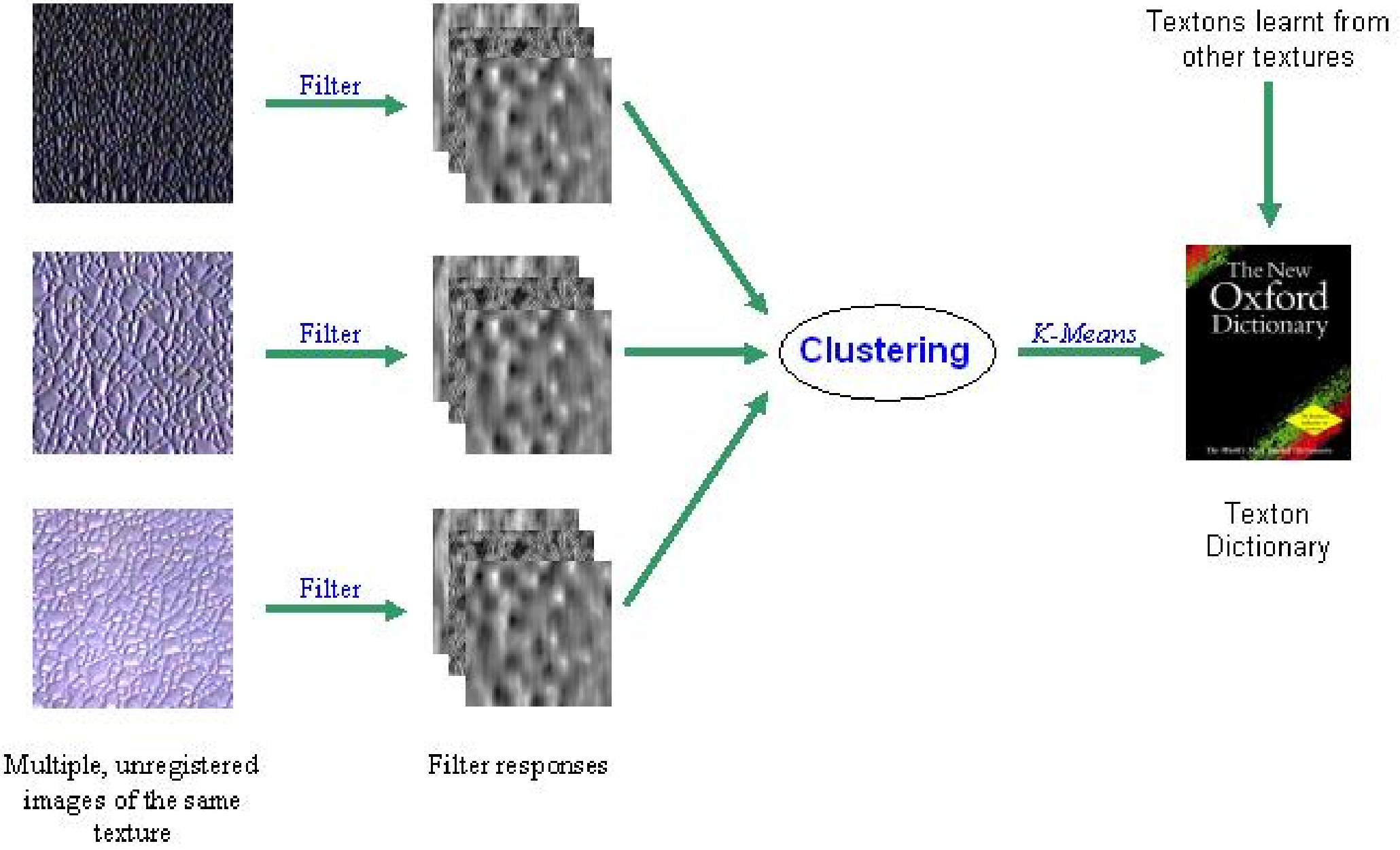
Texture Modeling

- Learned filters using a deep convolutional neural network
  - [2010+]



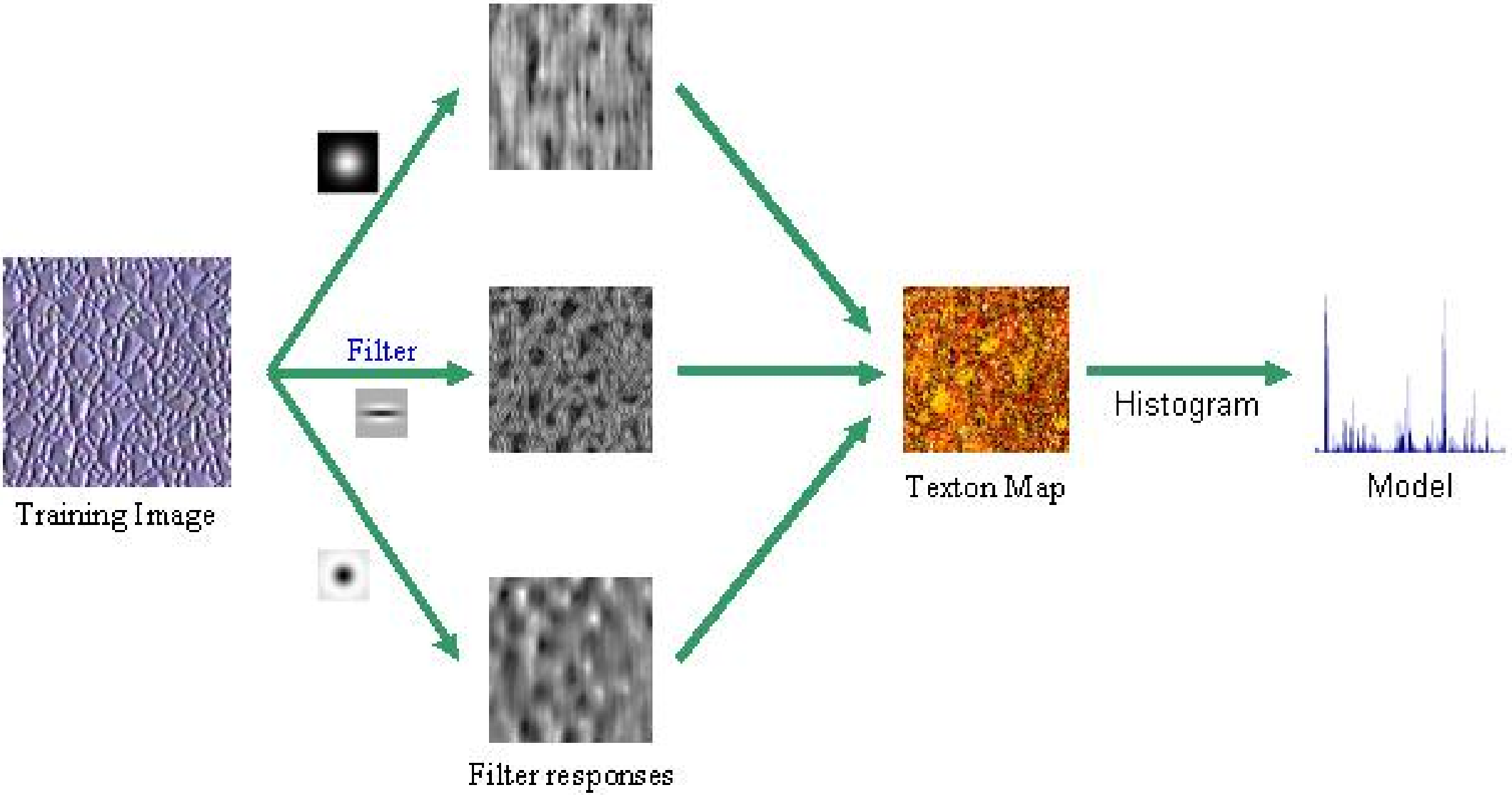
Texture Classification

- (1/3) Learning a dictionary (of textons)



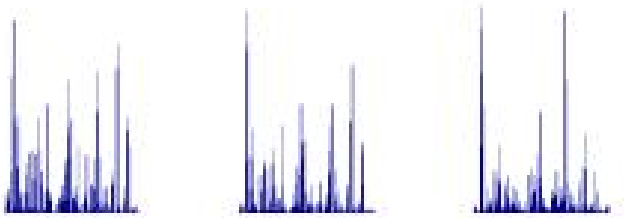
Texture Classification

- (2/3) Building a model



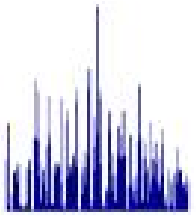
Texture Classification

- (3/3) Classifying unseen images
  - Nearest neighbor
  - Nearness → chi-square distance



Plastic

Novel Image



Model

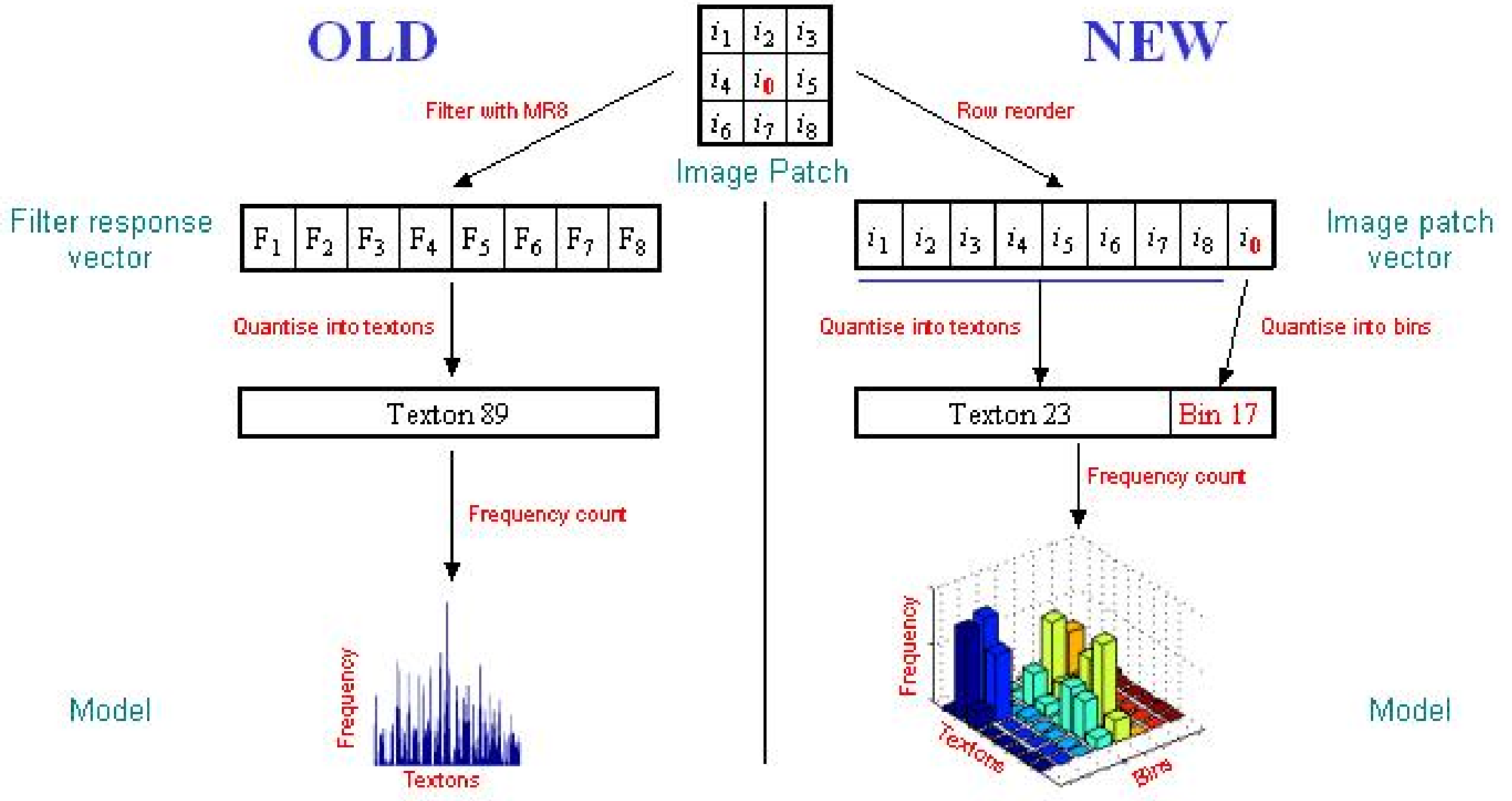
χ^2 =

Grass

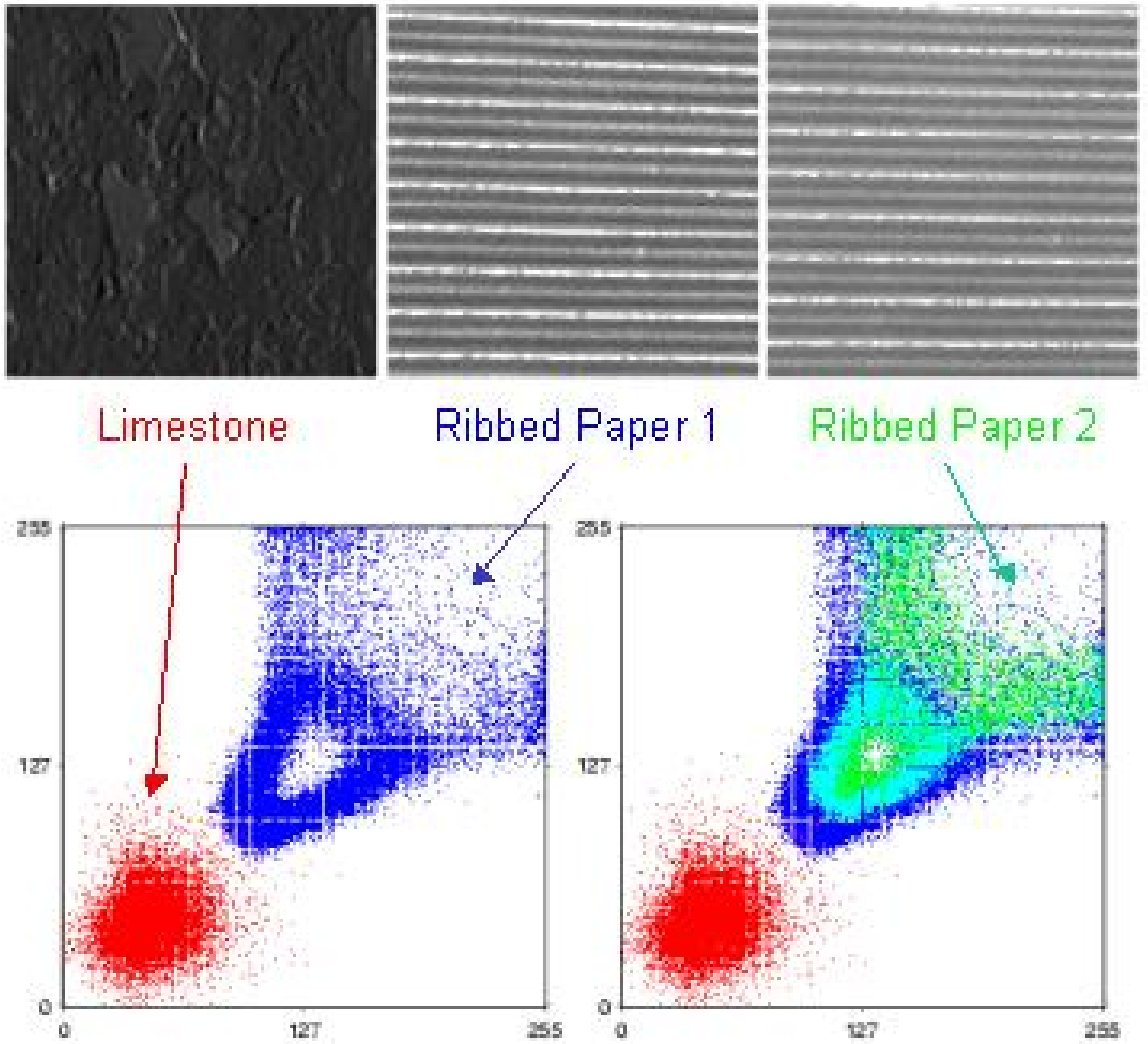
Texture Classification

- Replace filter responses by patch [Verma,Zissermann]

Model changes from joint PDF of filter responses to joint PDF of raw pixel intensities computed over all N x N patches in image:



Why Should Small Image Patches Work?



Co-occurrence plots: $I(x)$ vs $I(x + (2,2))$

Distributions are clearly distinct ⇒ 3x3 patches are sufficient for classification

Texture Modeling

- How good is the model ?
  - Given image, we use model to encode (quantize) information into the index of the nearest texton at each pixel
  - Have we lost information in the quantization ?
  - How can we measure or visualize information loss ?

Texture Modeling

- Consider image with N pixels
  - Vectorized image V is Nx1 column vector
- Assume a filter bank with F filters
- Consider a linear system (matrix M) that takes vectorized image and produces all feature responses at each pixel
  - F responses at 1st pixel ← F x N matrix
  - F responses at 2nd pixel ← F x N matrix
  - …
  - Append all rows → M → NF x N matrix **(very sparse)**
- Then, R = MV has size NF x 1
  - R contains all filter responses at all pixels

Texture Modeling

- Can we compute (regularized) inverse of the feature responses R through the matrix M ?
  And hope to get the original image back ?

- Quantization replaces the actual filter response by the filter response associated with nearest “texton”
  - Call this quantized set of feature responses by R*

- Can we compute (regularized) inverse of the feature responses R* through the matrix M ?
  And hope to get back close to the original image ?

Texture Modeling

- Illustration of K-means clustering and reconstruction from filter responses with K = 20. **(a)** Original image.
  (b) K-means centers reconstructed as local filters. These centers correspond to the dominant features in the image: bars and edges at various orientations and phases;
  (c) Reconstruction of the quantized image.
  Close resemblance between (a) and (c) suggests that quantization doesn't introduce much error perceptually.

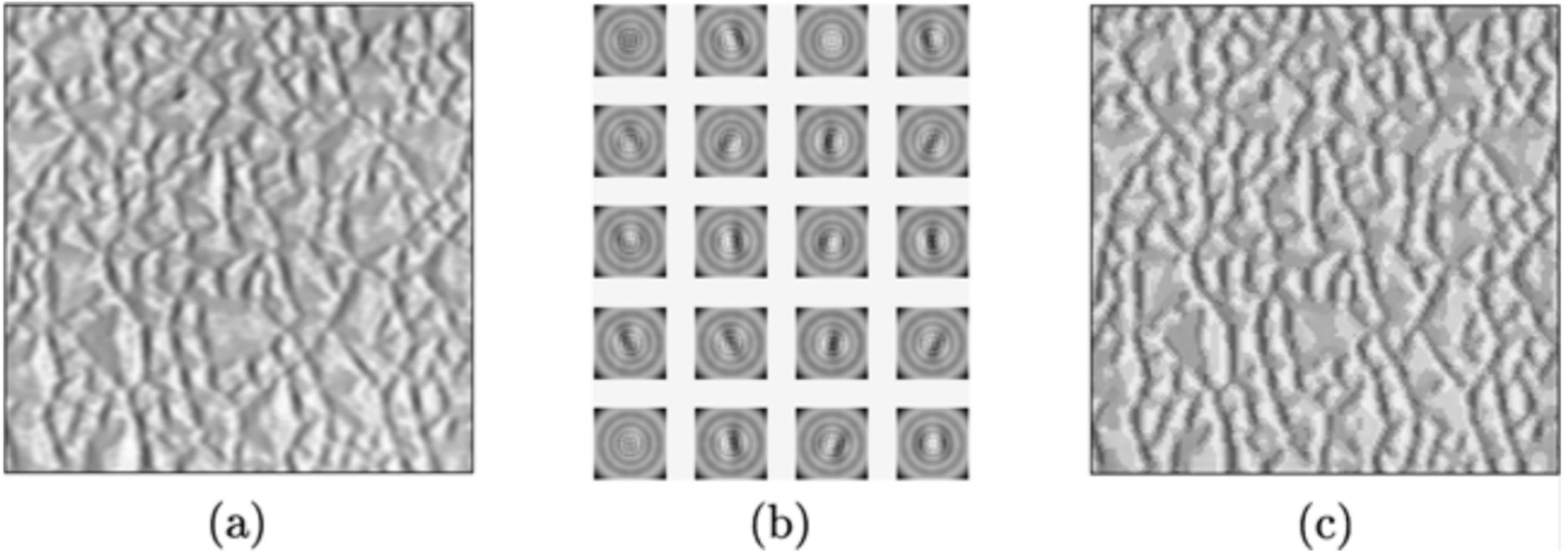


Image Classification

- Difficult problem: object recognition

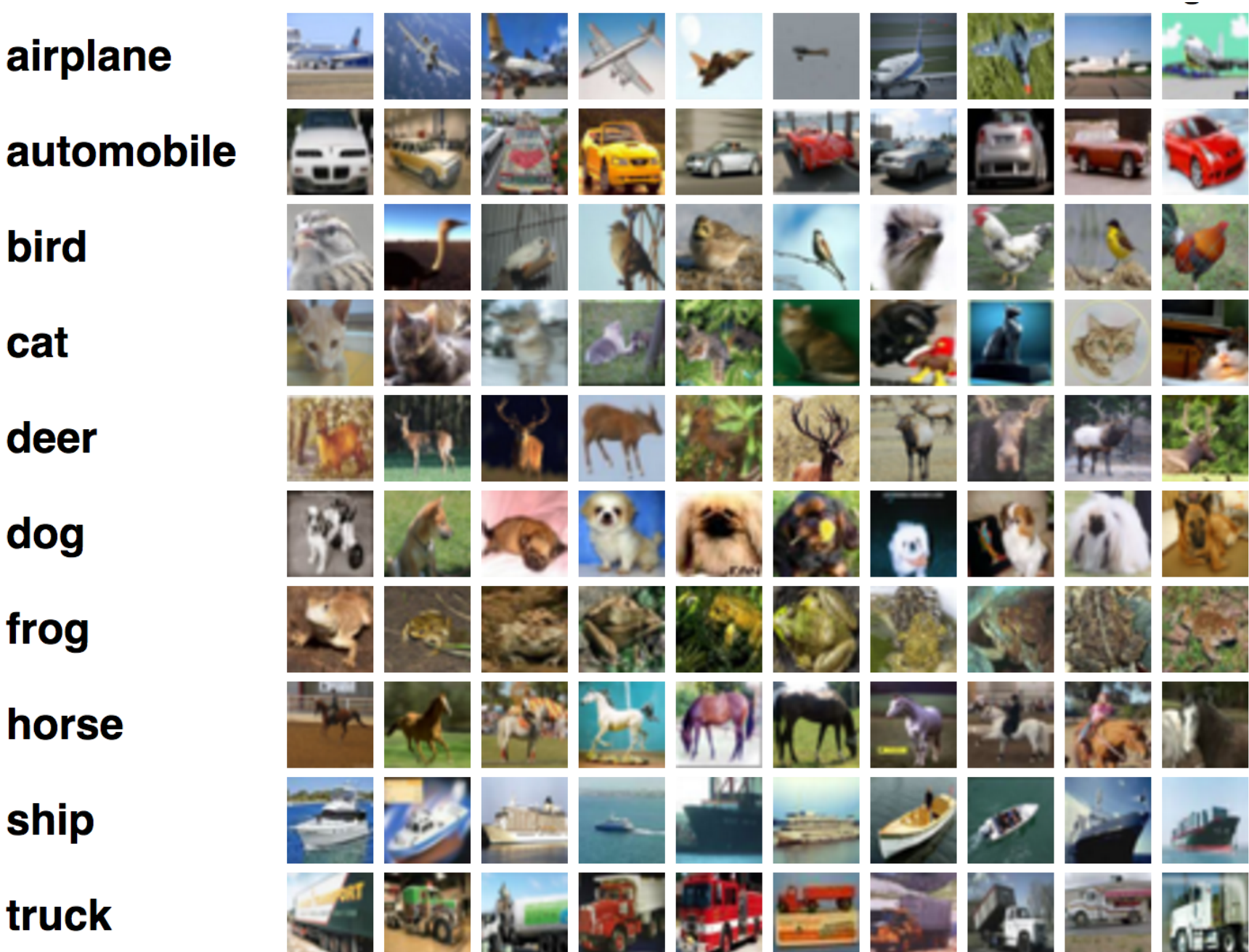


Image Classification

- Other difficult problems

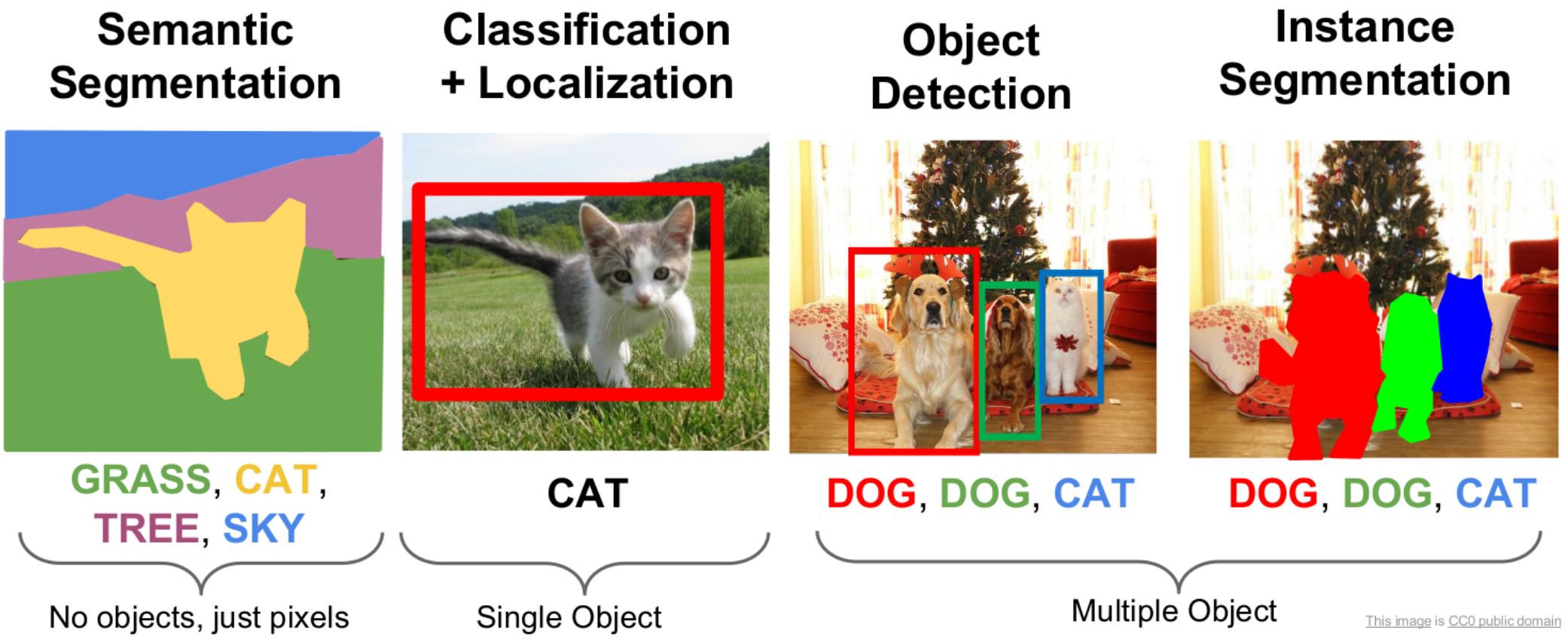
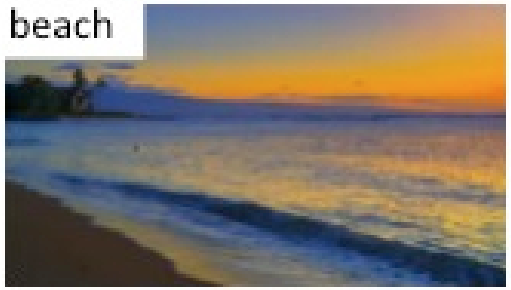
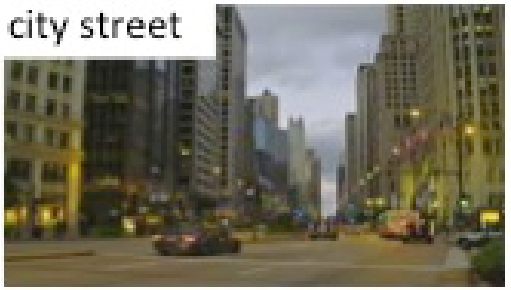
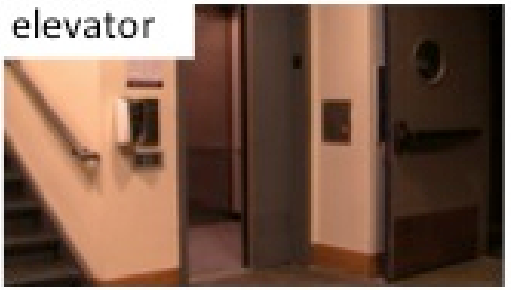


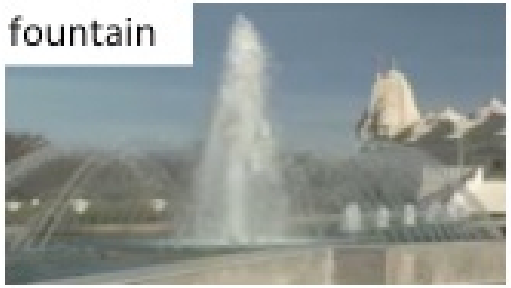
Image Classification

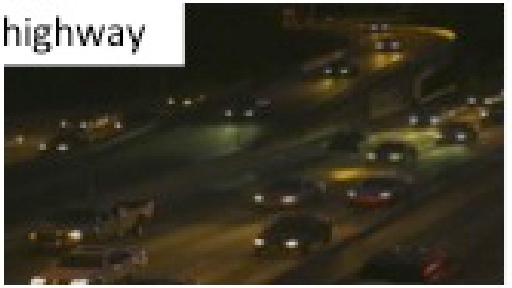
- More difficult problems: scene classification

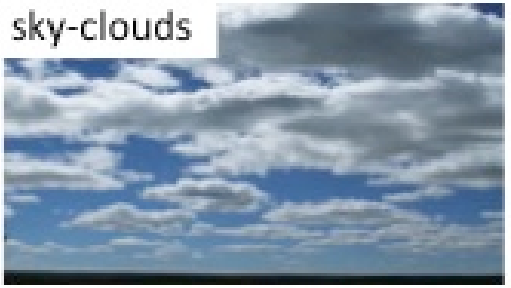












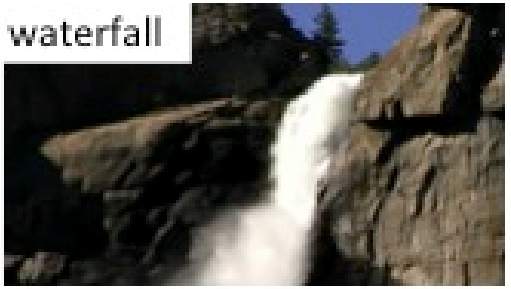


Image Classification

- Why difficult ?
  - Very many classes
  - Lot of intra-class variation

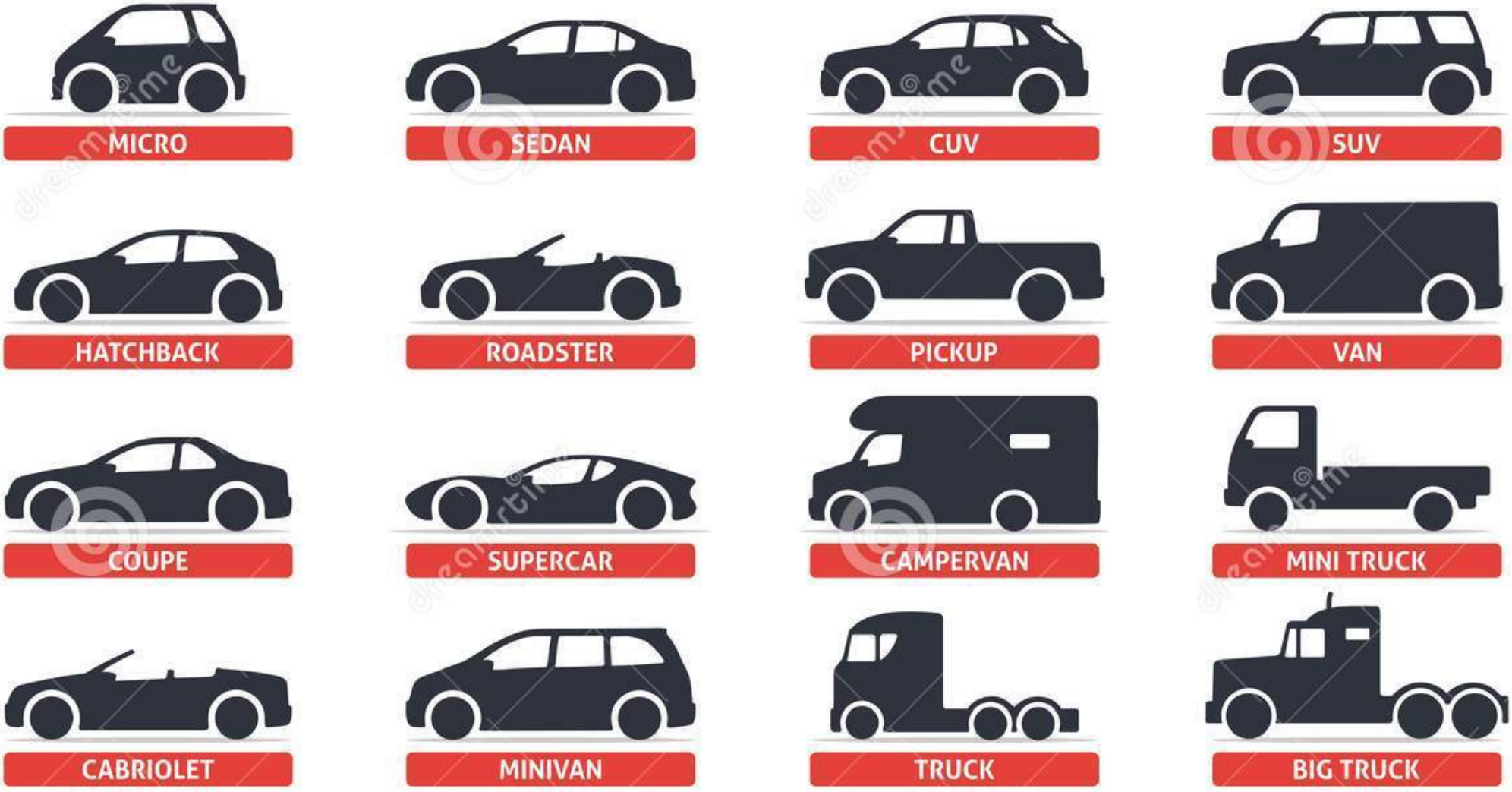


Image Classification

- What do you see ?

Image Classification

- What do you see ?

Image Classification

- What do you see ?

Image Classification

- Text / document classification
  - "**Bag of words**" (BoW) model
  - Certain classes of documents (e.g., sports related articles) have a certain set of keywords occurring frequently

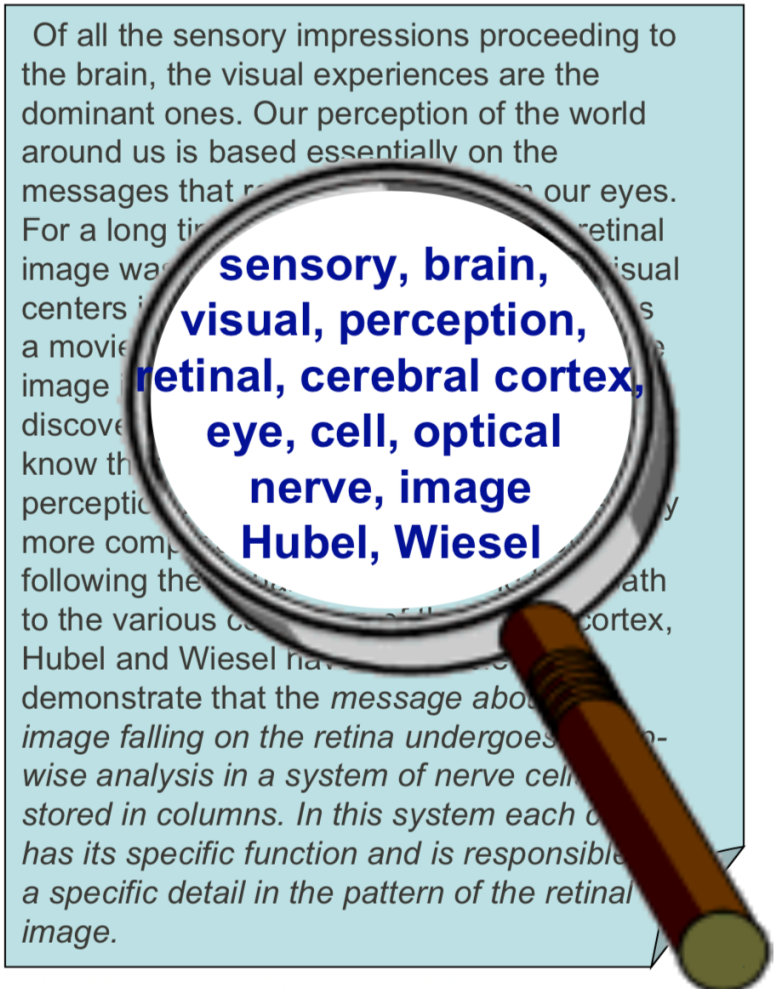


Image Classification

- Bag of ‘visual’ words
  - Visual word ↔ texton

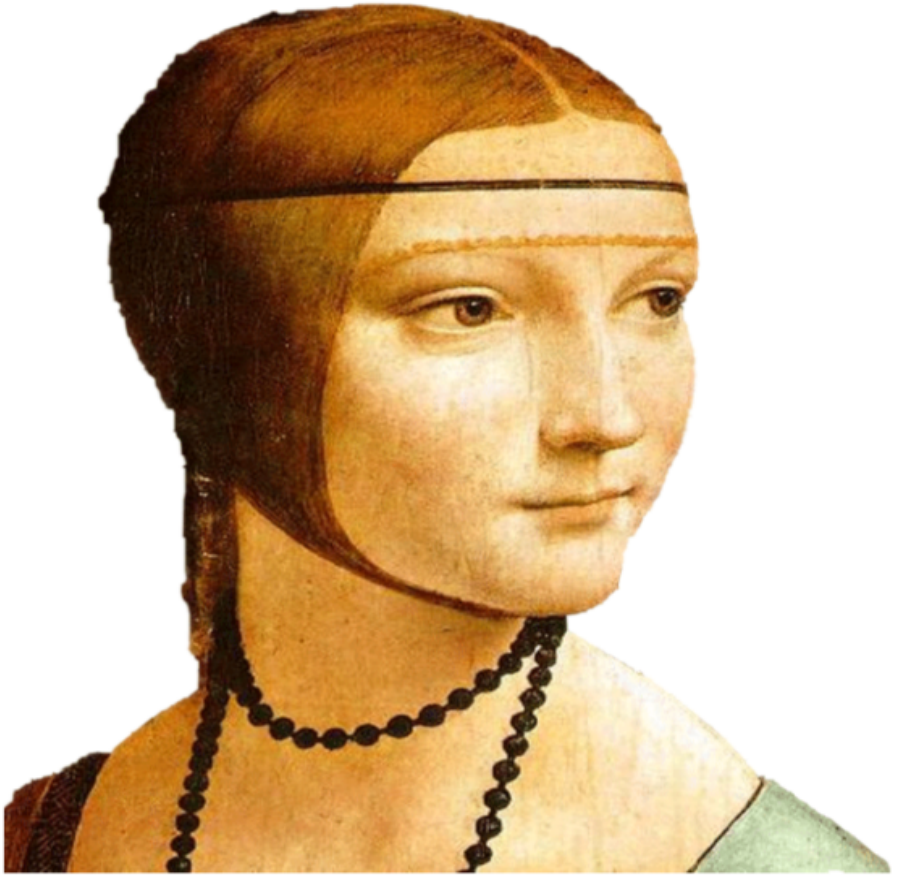


Image C

- Bag of ‘visual’ words model

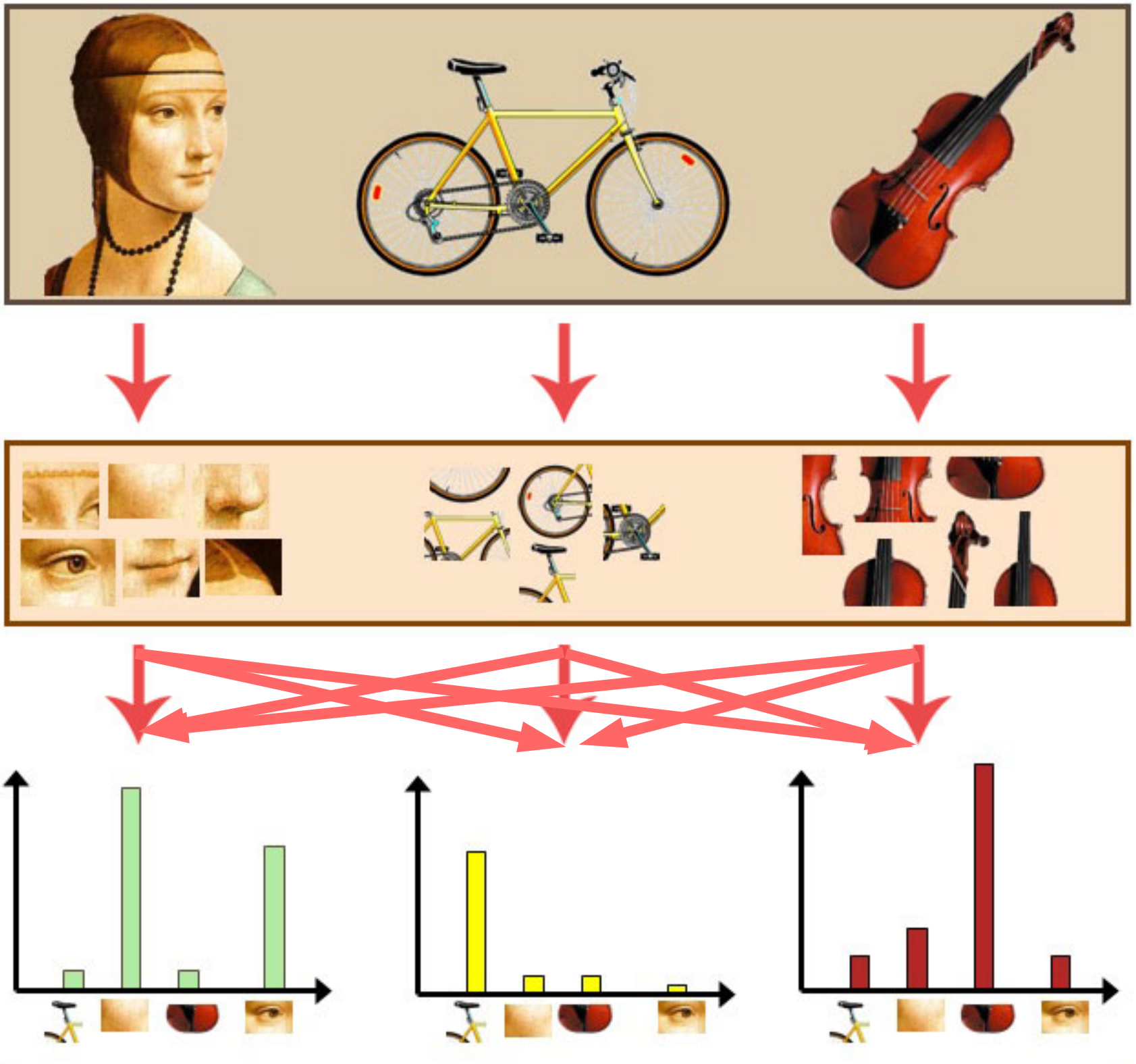
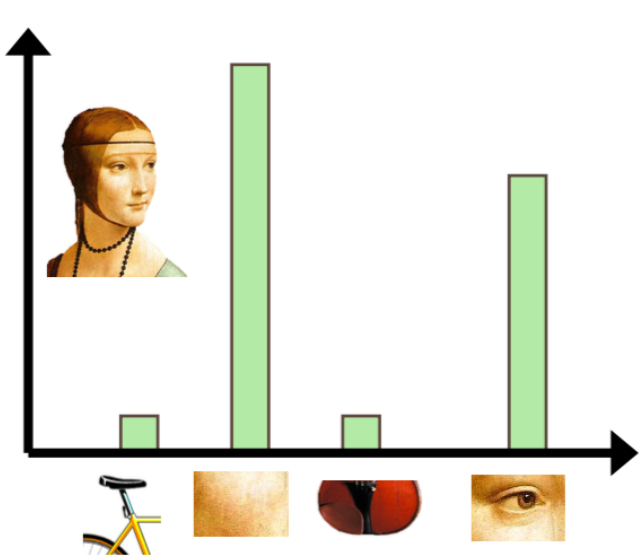
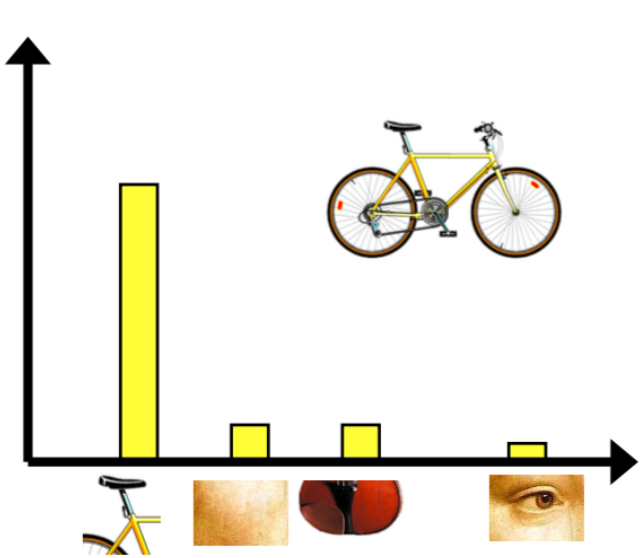
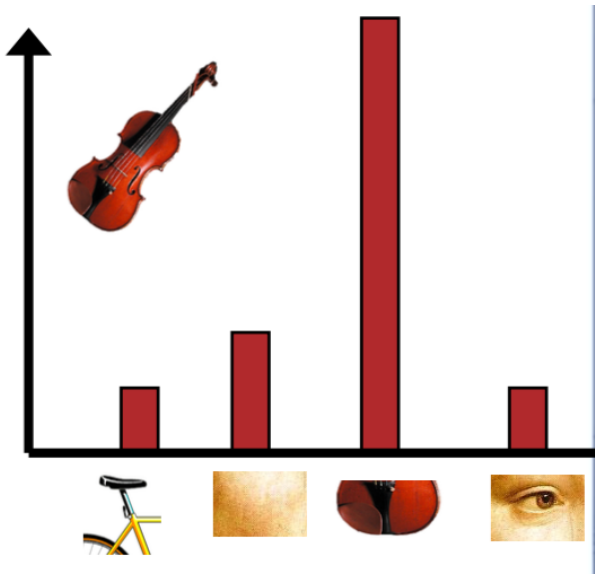


Image Classification

- Images from ‘face’ class, ‘bicycle’ class, ‘violin’ class have different compositions in terms of “words”
  - One way to characterize composition → histogram







Model Learning -- Recognition

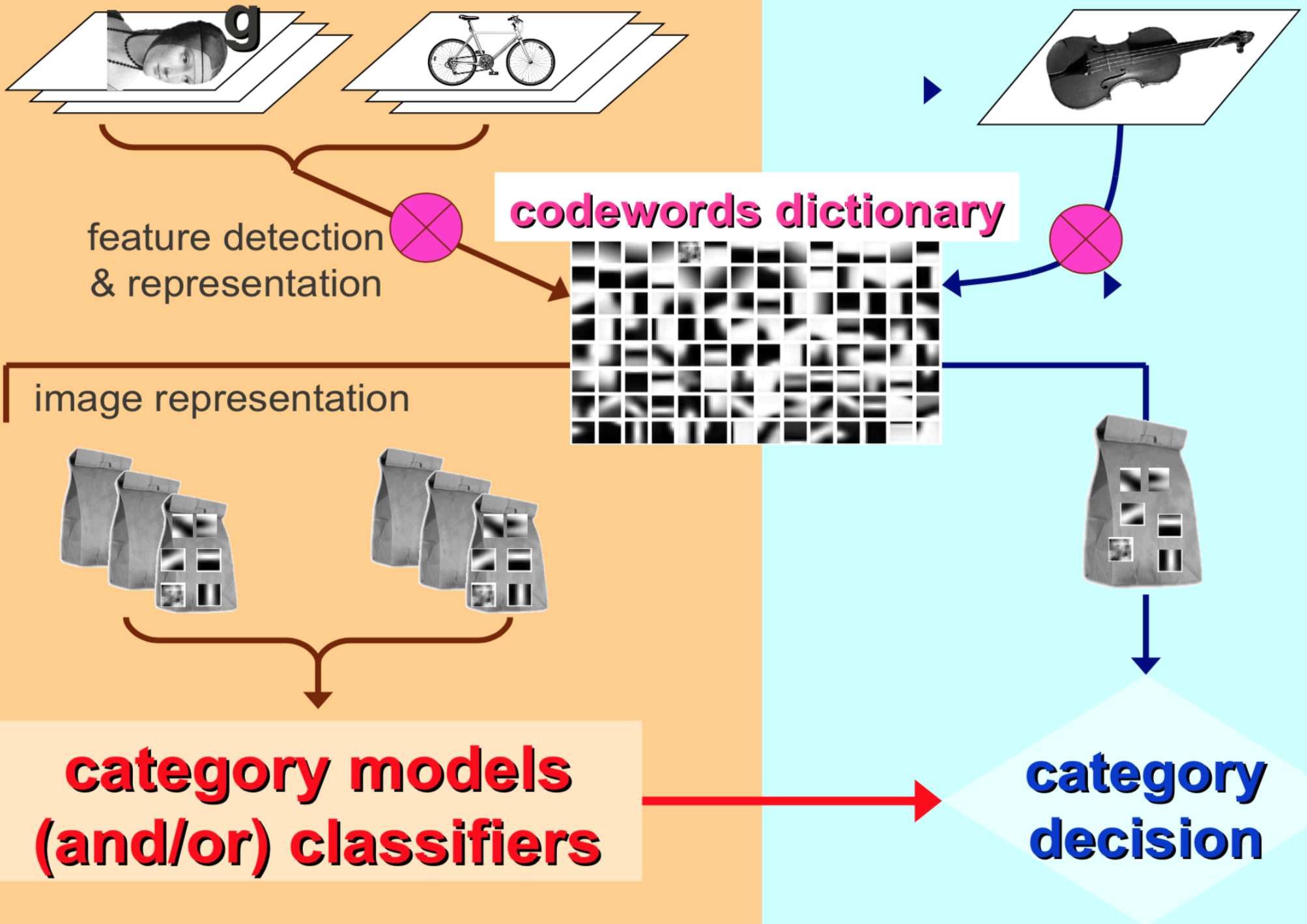
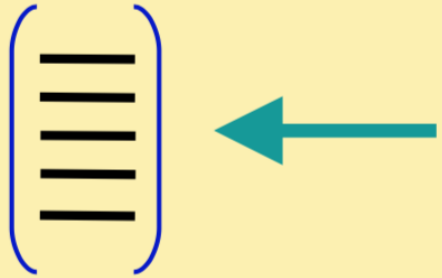


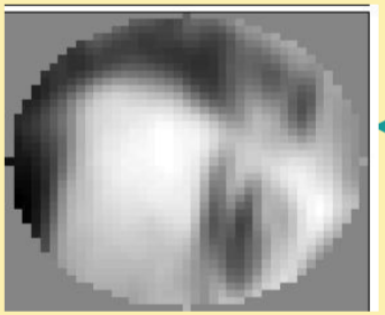
Image Classification

- (1) Feature detection and representation
  - (a) Detect keypoints and extract features from keypoints



Compute SIFT descriptor

[Lowe'99]



Normalize patch

Detect patches

[Mikojaczyk and Schmid '02]

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

[Sivic & Zisserman, '03]

Image Classification

- (1) Feature detection and representation
  - (a) Detect keypoints and extract features from keypoints
  - (b) Extract features from a dense tessellation of the image
    - e.g., dense SIFT

SIFT descriptors of 16x16 patches sampled on a regular grid, quantized to form visual vocabulary (size 200, 400)

Image Classification

- (2) Visual “codewords” / dictionary learning

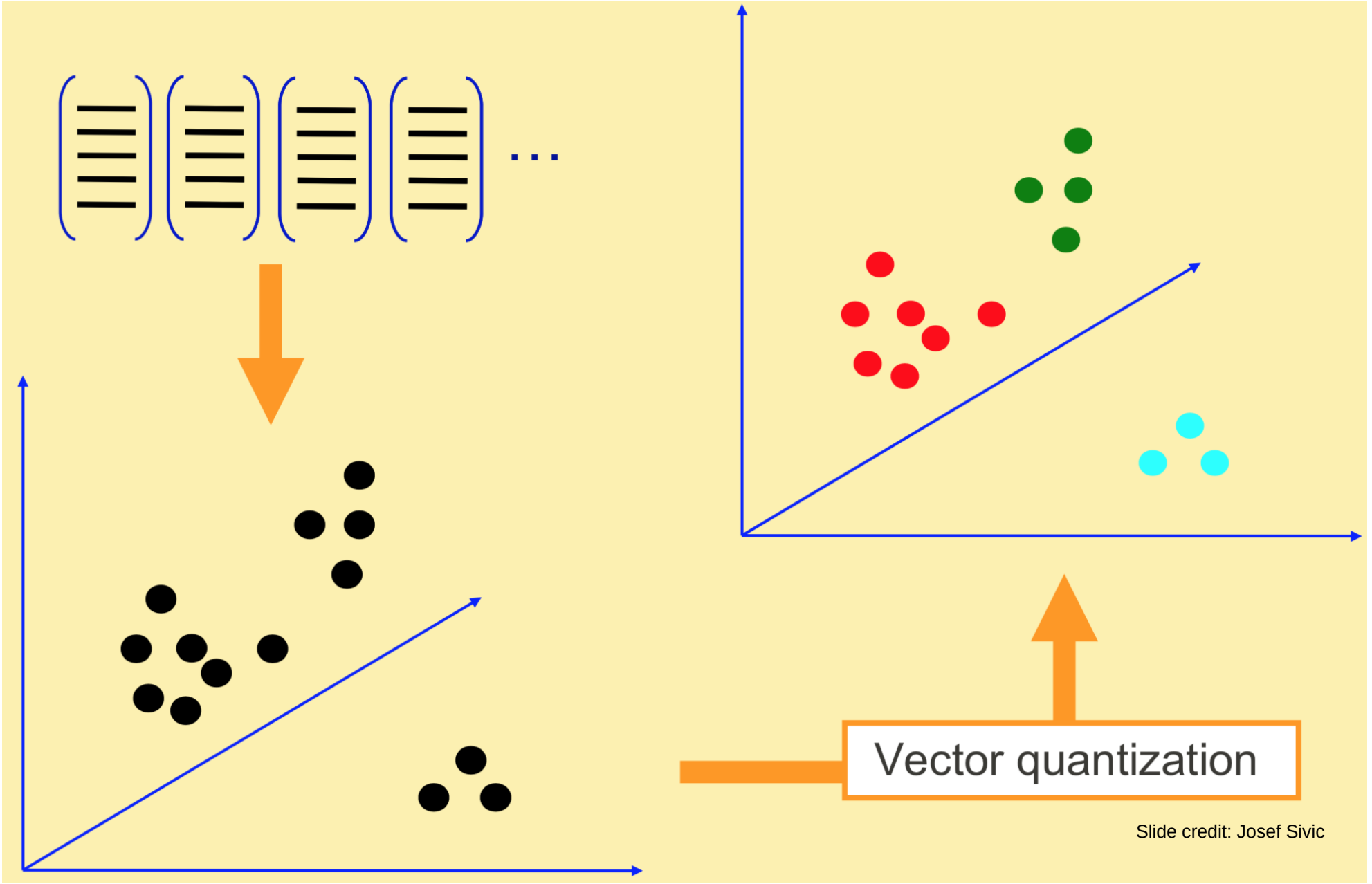
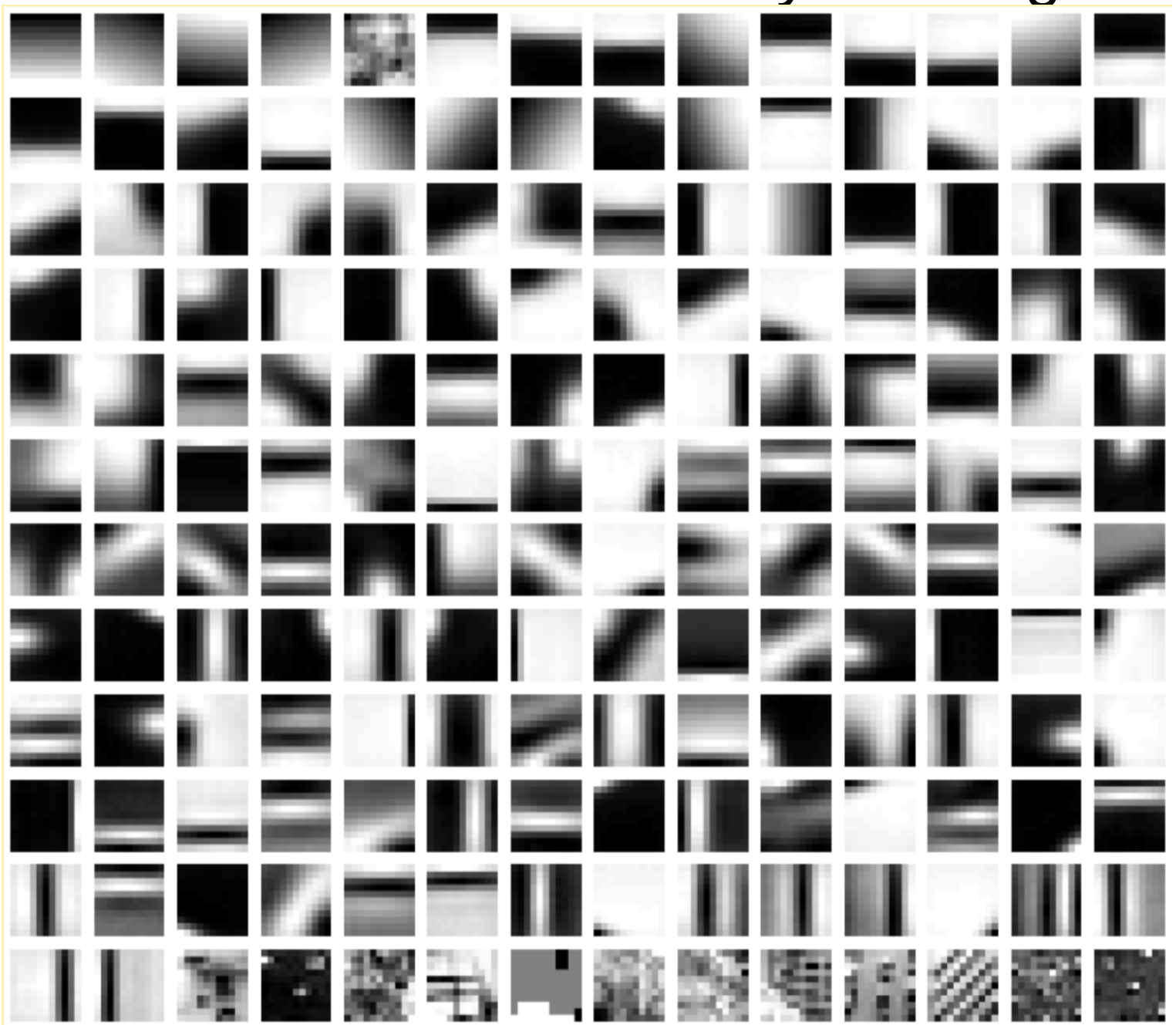


Image Classification

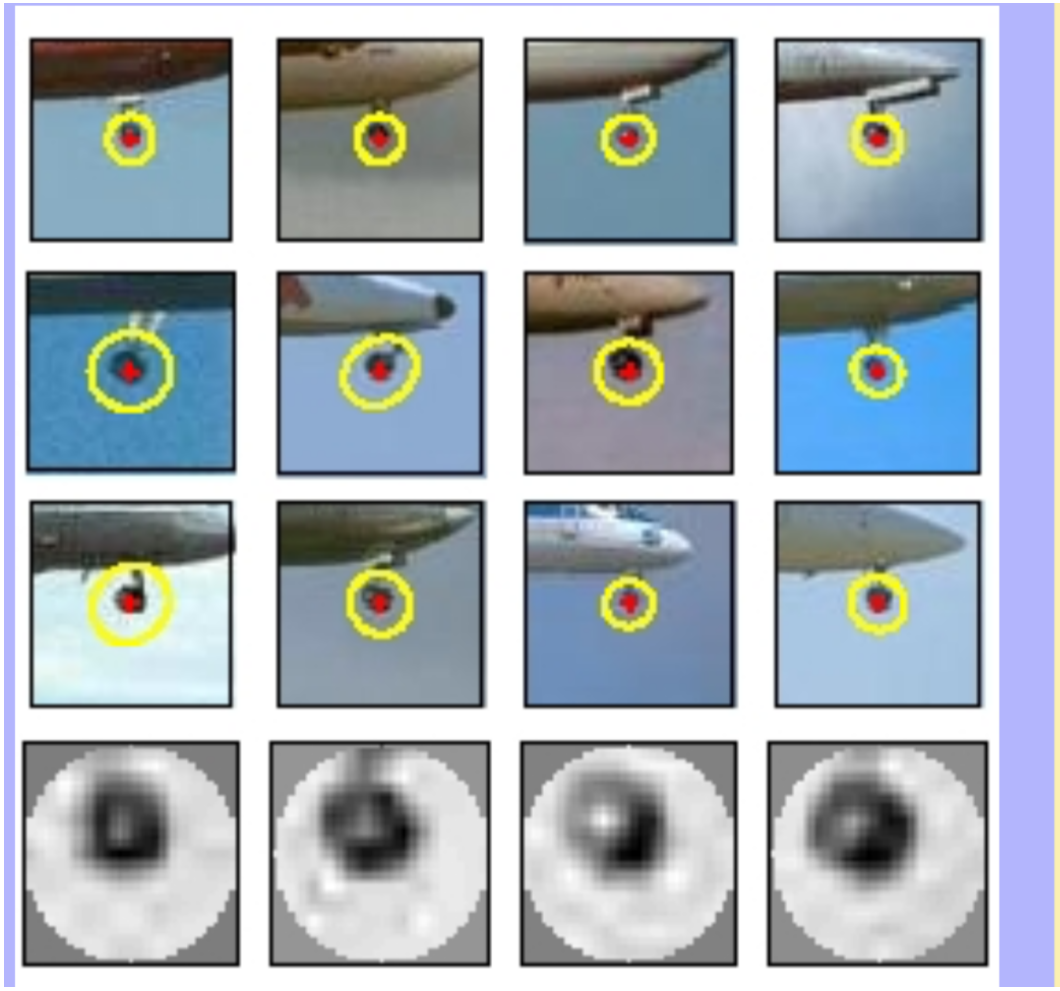
- (2) Visual “codewords” / dictionary learning



Fei-Fei et al. 2005

Image Classification

- (2) Visual “codewords” / dictionary learning
  - Examples



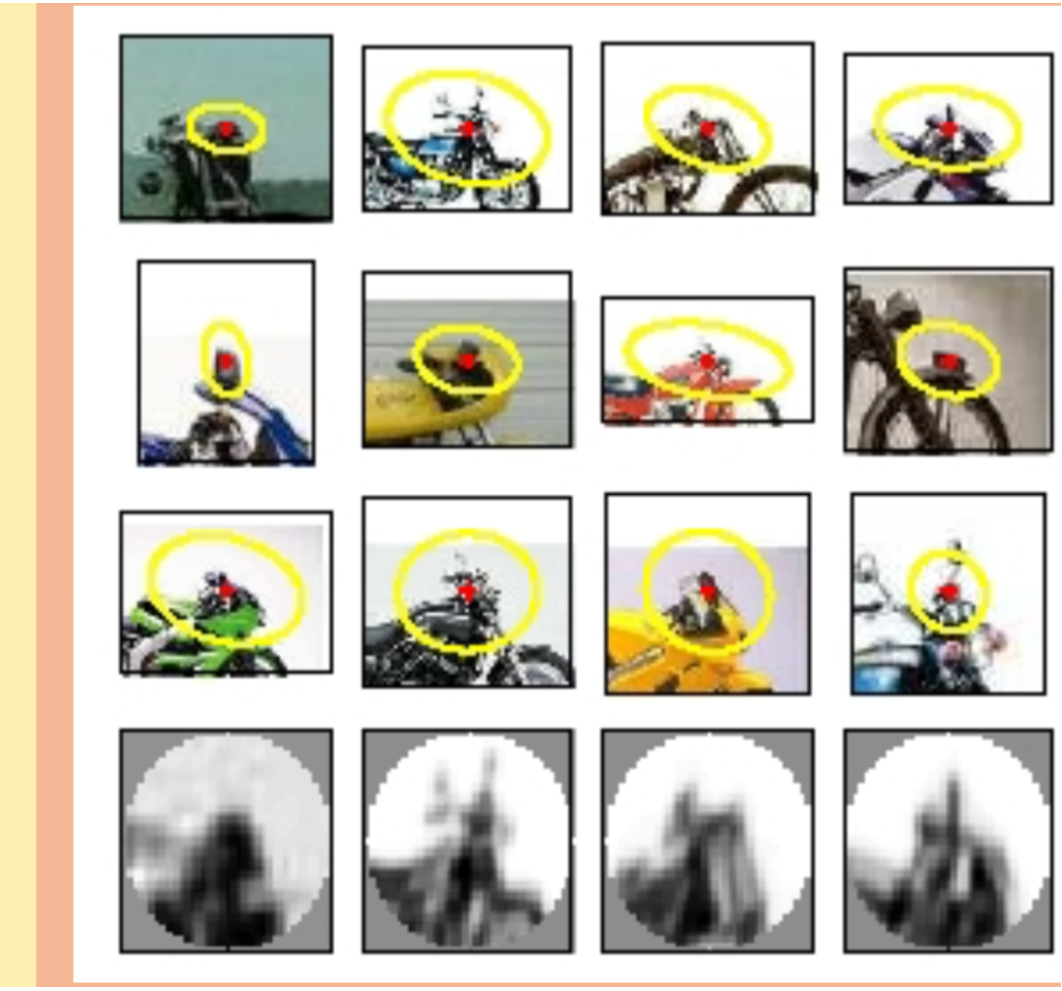


Image Classification

- (1) Feature detection and representation
- (2) Visual “codewords” / dictionary learning

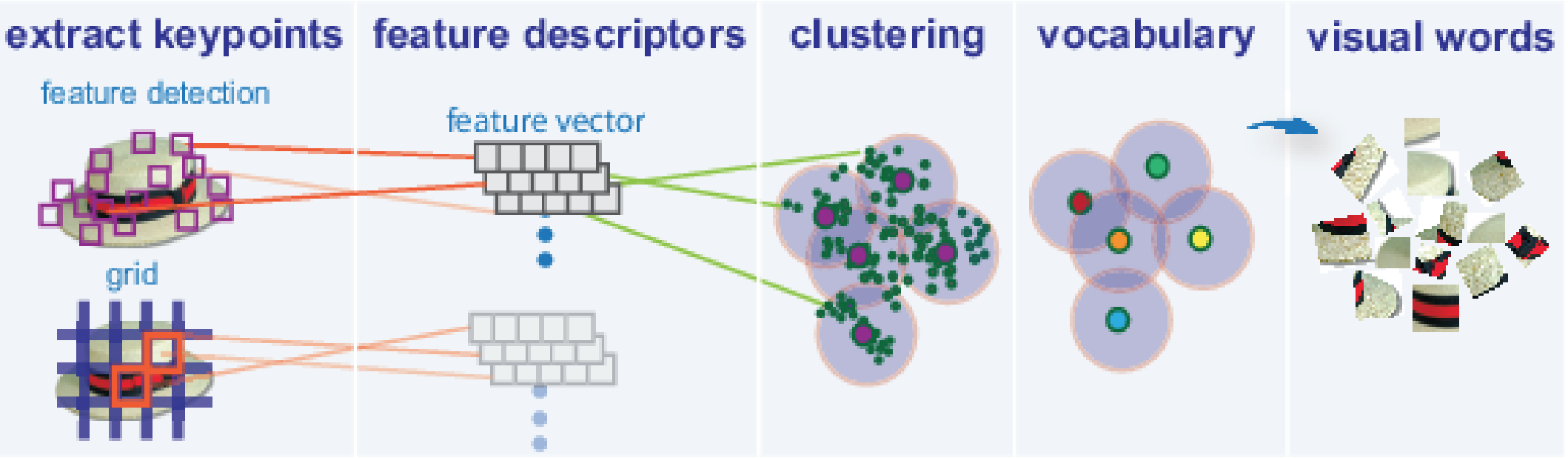


Image Classification

- (3) Image representation

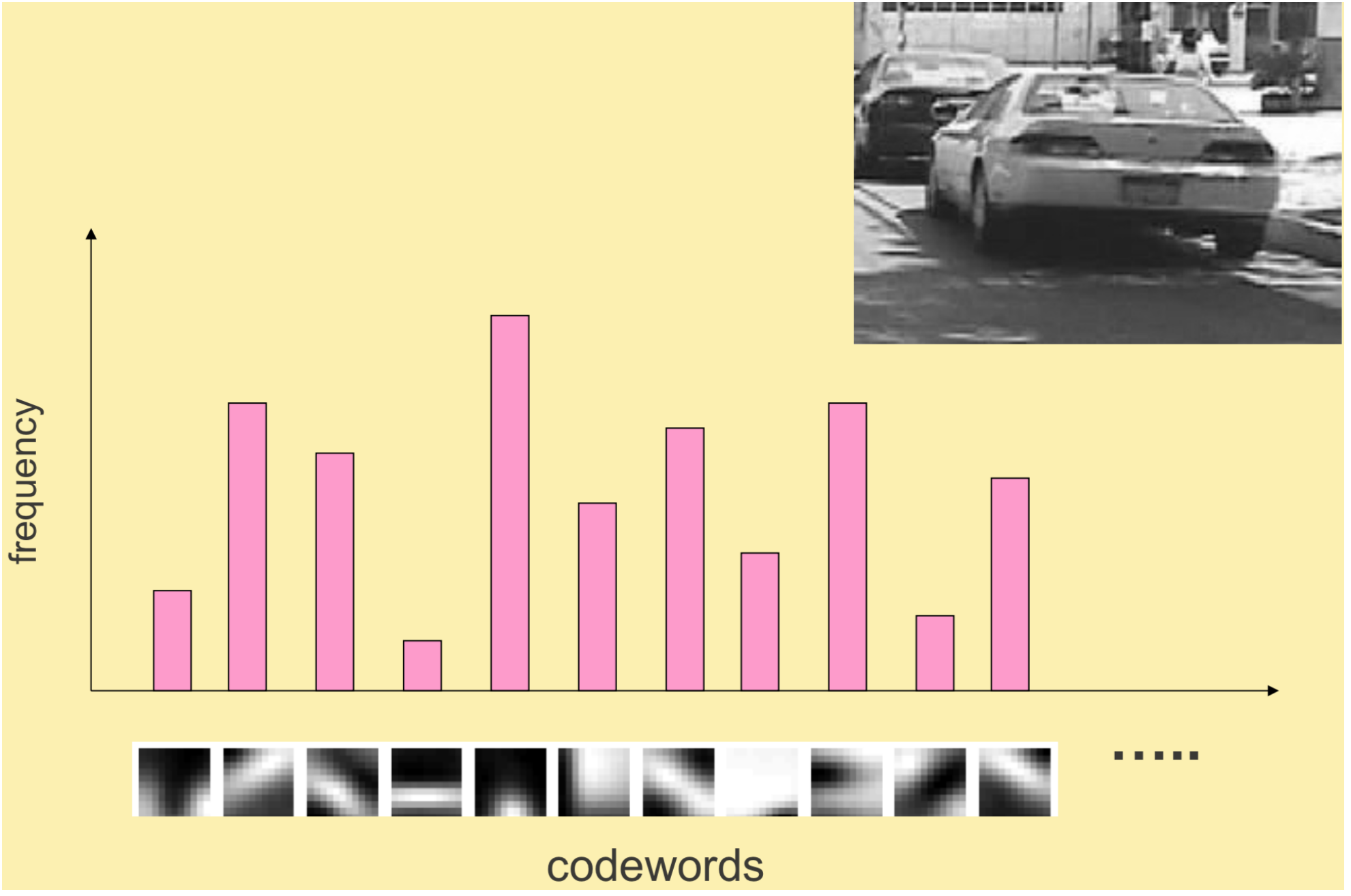


Image Classification

- (4) Recognition
  - Define a similarity measure between histograms
    - e.g., chi-square distance
  - Learn any classifier
    - K nearest neighbor
    - Support vector machine

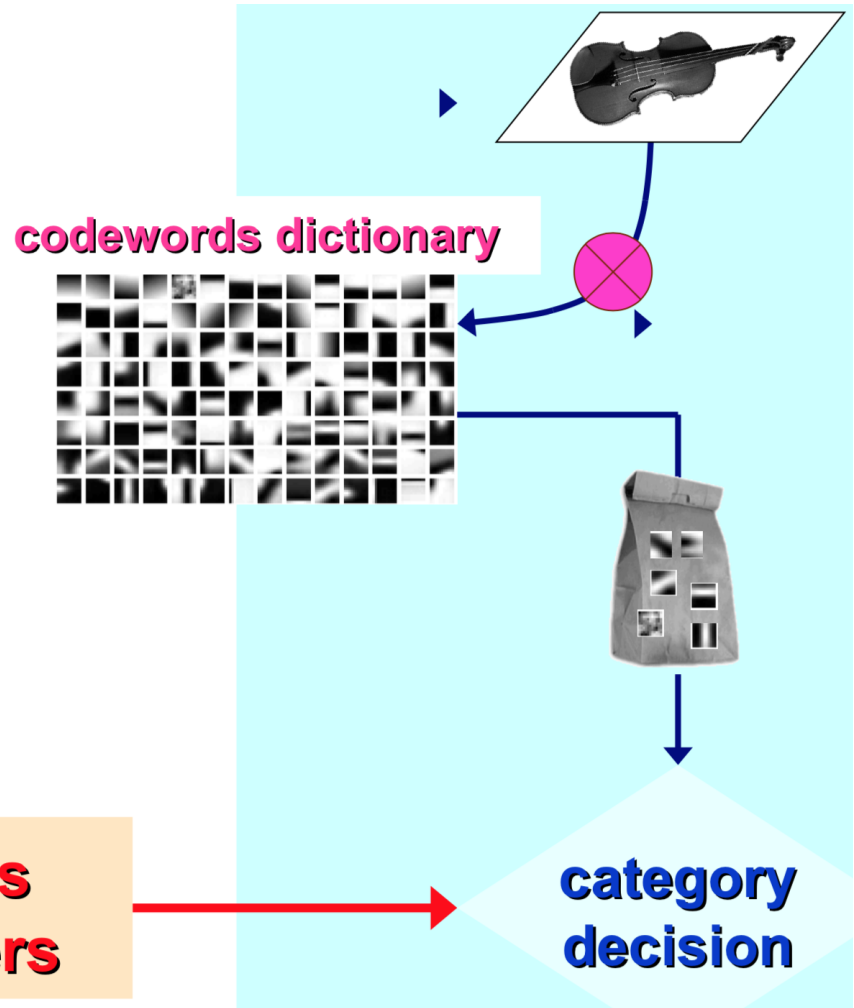


Image Classification

- More variations
  - For a feature vector,
    instead of assigning a single texton index,
    assign soft memberships to each texton

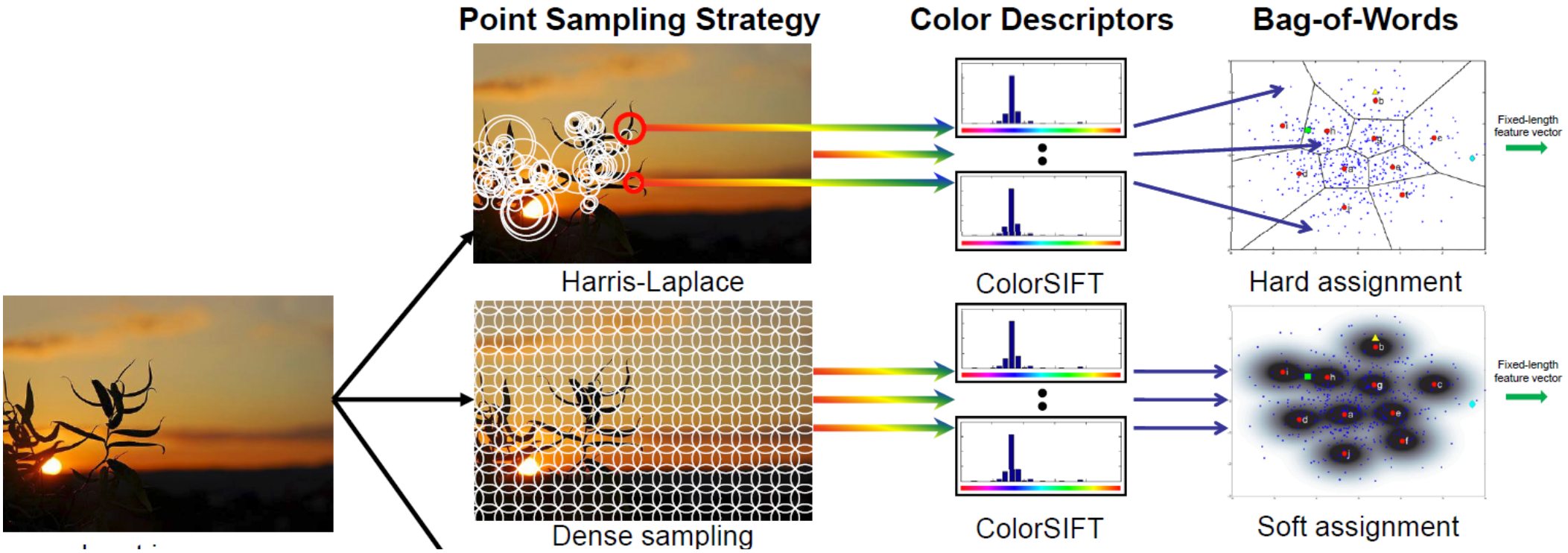
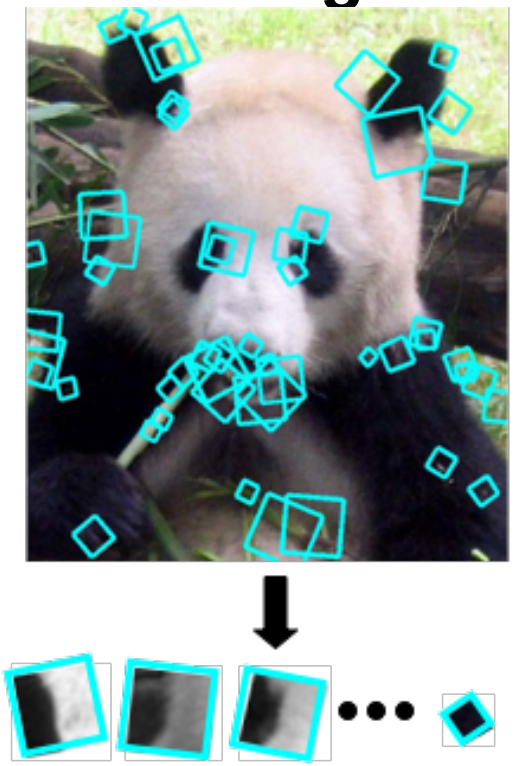
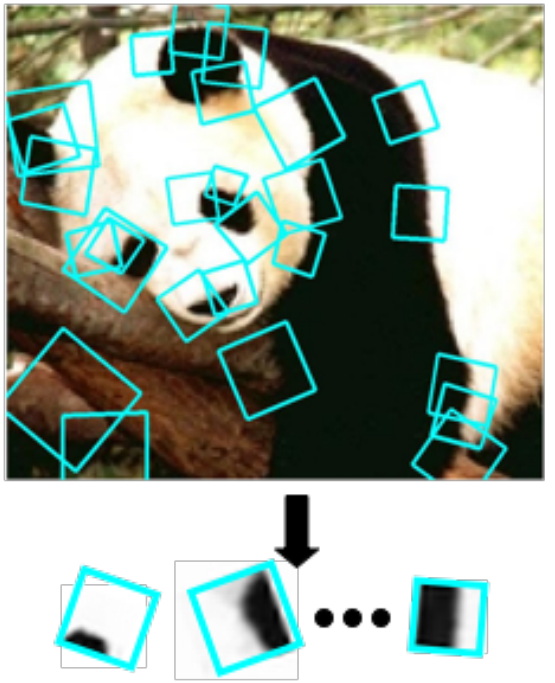


Image Classification

- How to **define a similarity measure** between 2 unordered sets of (feature) vectors ?
  - May, or may not, quantize feature vectors by textons
  - But, instead of building texton histogram, find a **matching of features in the feature-space itself**



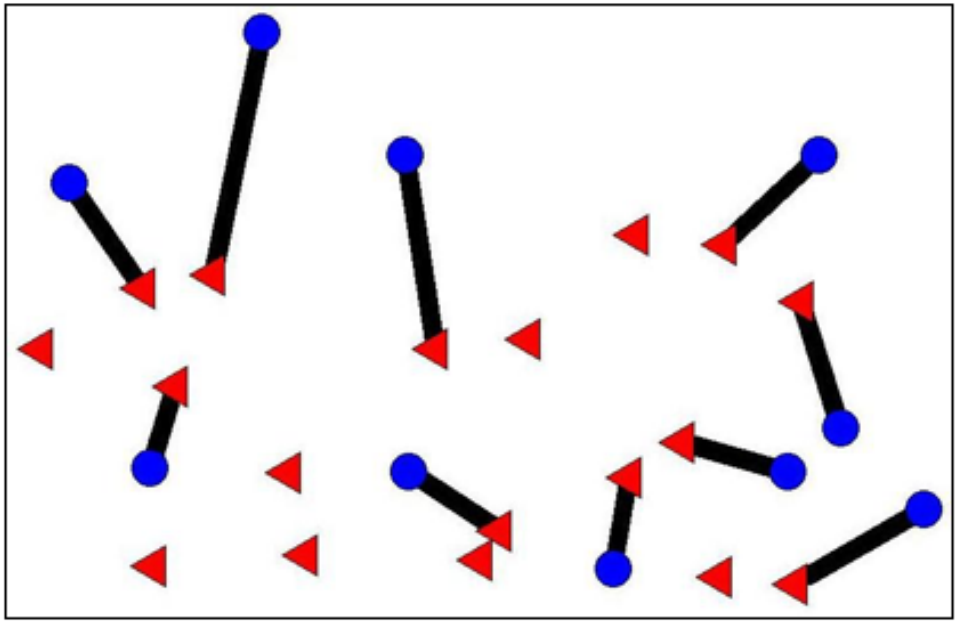
$$\mathbf{X} = \{\vec{\mathbf{x}}_1, \ldots, \vec{\mathbf{x}}_m\}$$

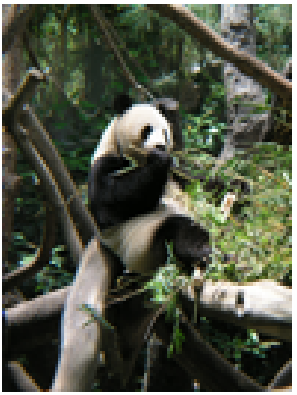
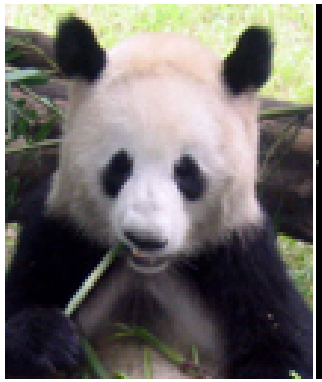
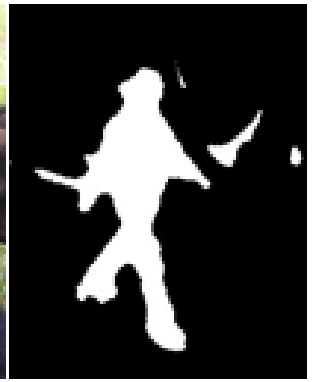


$$\mathbf{Y} = \{\vec{\mathbf{y}}_1, \ldots, \vec{\mathbf{y}}_n\}$$

Image Classification

- Compare sets by computing a **partial matching** between their features (in feature space)
  - Robust to clutter, segmentation errors, occlusion



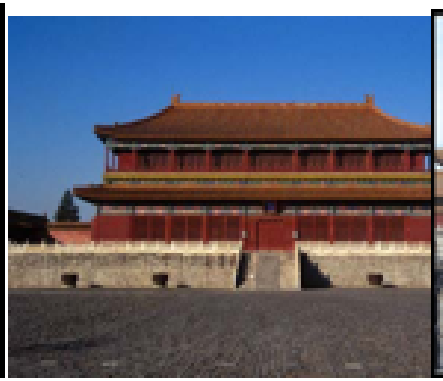
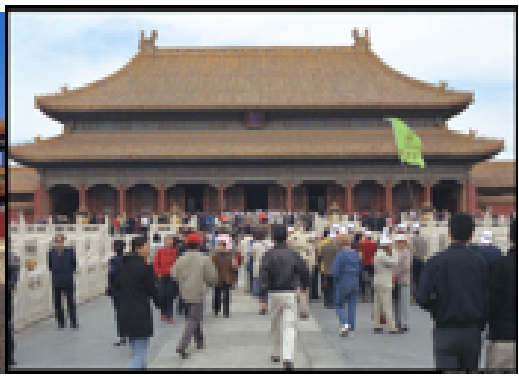
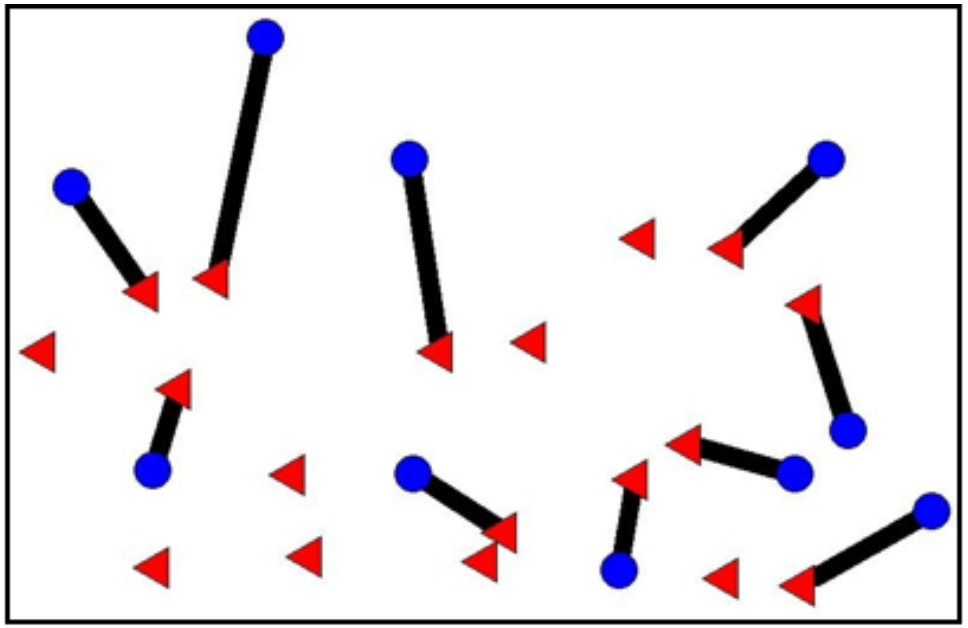



Image Classification

- **Optimal** partial matching
  - S(.,.) is some similarity measure in feature space
  - This is a hard problem
  - Is there a faster approximate way ?

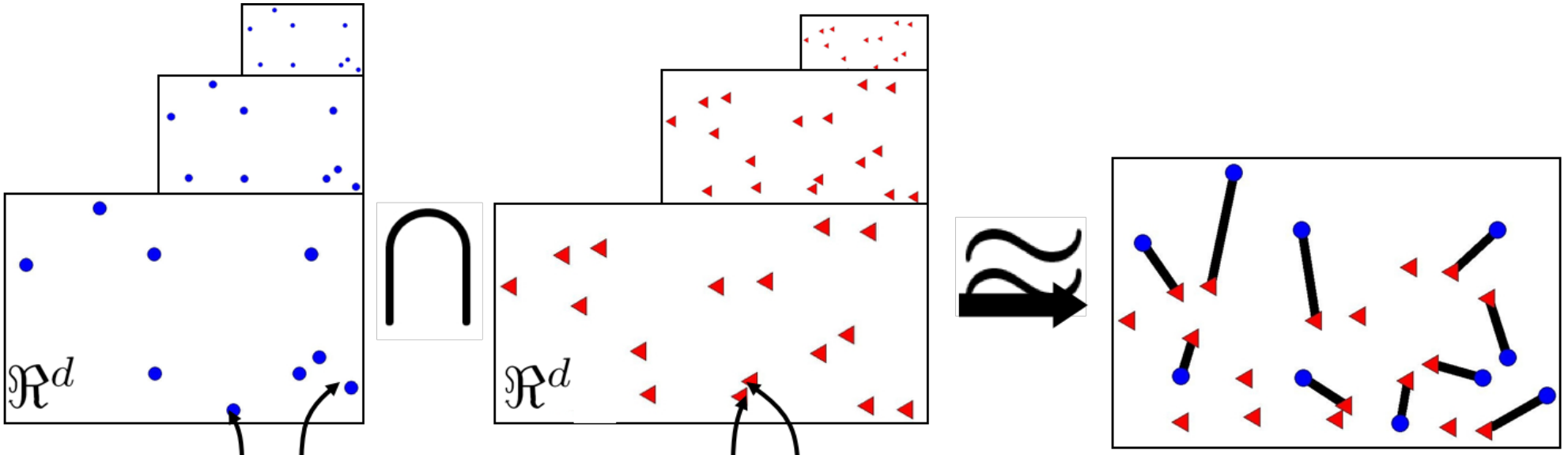


optimal partial matching

$$\max_{\pi:\mathbf{X}\to\mathbf{Y}} \sum_{\mathbf{x}_i\in\mathbf{X}} \mathcal{S}(\mathbf{x}_i, \pi(\mathbf{x}_i))$$

Image Classification

- **Pyramid match kernel (PMK)** [Grauman+Darrell 2005 ICCV, 2007 JMLR]



optimal partial matching

$$\max_{\pi:\mathbf{X}\to\mathbf{Y}} \sum_{\mathbf{x}_i\in\mathbf{X}} \mathcal{S}(\mathbf{x}_i, \pi(\mathbf{x}_i))$$

$$\mathbf{X} = \{\vec{\mathbf{x}}_1, \ldots, \vec{\mathbf{x}}_m\}$$

$$\vec{\mathbf{x}}_i \in \Re^d$$

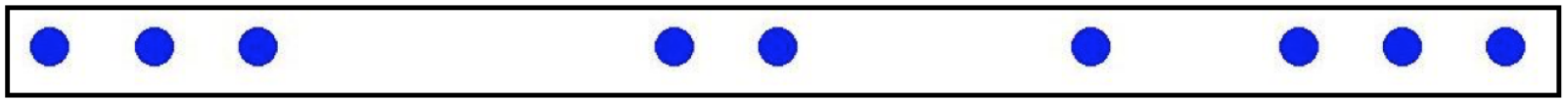
$$\mathbf{Y} = \{\vec{\mathbf{y}}_1, \ldots, \vec{\mathbf{y}}_n\}$$

$$\vec{\mathbf{y}}_i \in \Re^d$$

Image Classification

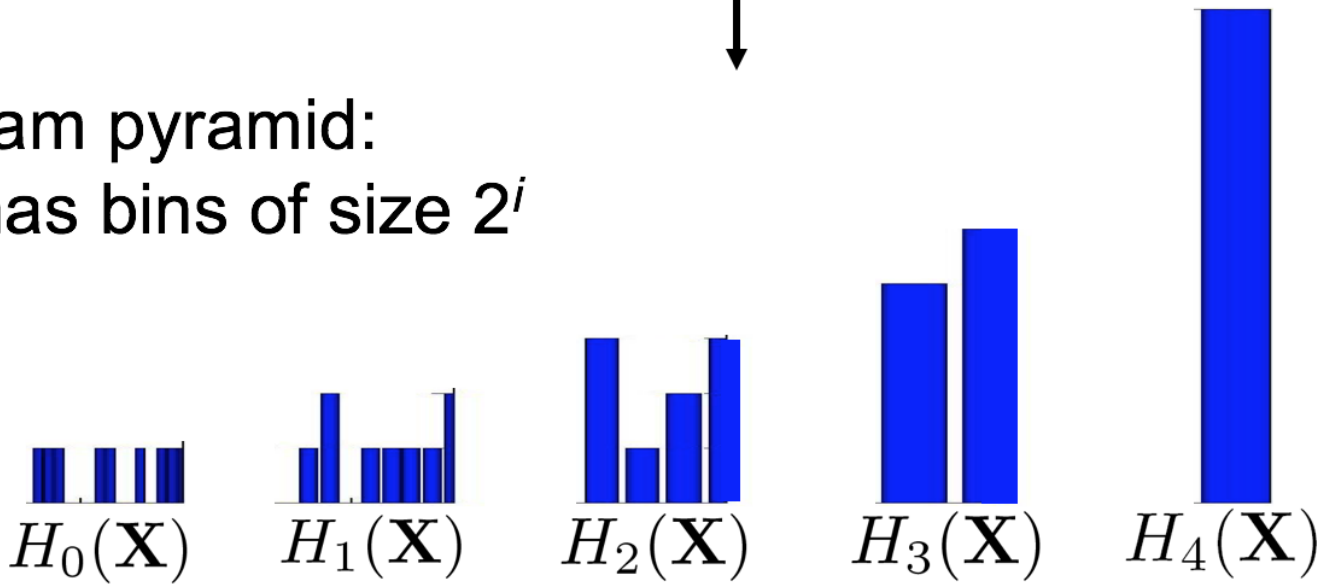
- Pyramid match kernel (PMK)
  - Bin the feature space: 16 bins, 8 bins, 4 bins, 2 bins, 1 bin

$$\mathbf{X} = \{\vec{\mathbf{x}}_1, \ldots, \vec{\mathbf{x}}_m\}, \quad \vec{\mathbf{x}}_i \in \Re^d$$



$d = 1$

Histogram pyramid:
level *i* has bins of size 2^i



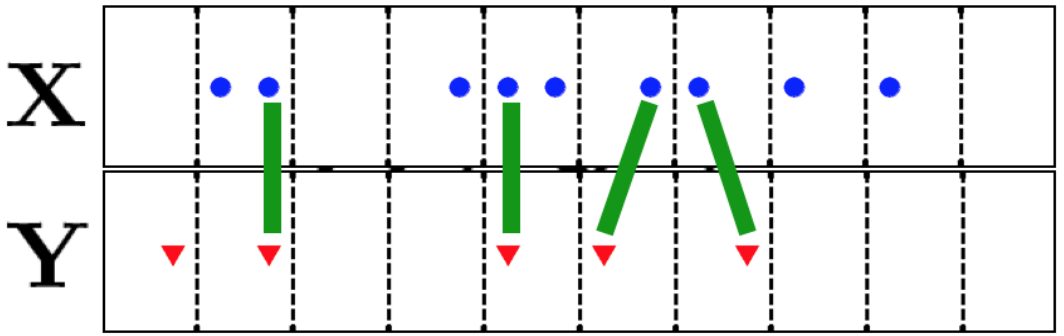
$$\Psi(\mathbf{X}) = [H_0(\mathbf{X}), \ldots, H_L(\mathbf{X})]$$

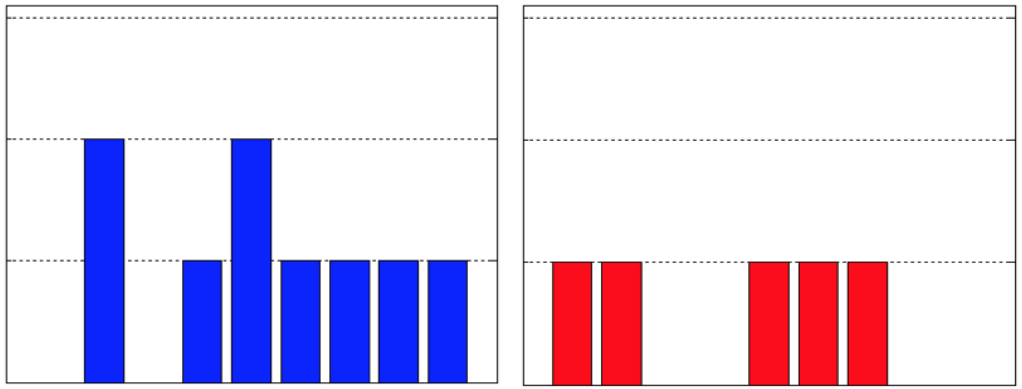
Image Classification

- Pyramid match kernel (PMK)
  - Count "**matches**" (= points in same bin 'j') at each scale

Histogram intersection

$$\mathcal{I}\left(H(\mathbf{X}), H(\mathbf{Y})\right) = \sum_{j=1}^{r} \min\left(H(\mathbf{X})_j, H(\mathbf{Y})_j\right)$$





$H(\mathbf{X})$ $H(\mathbf{Y})$ $\mathcal{I}\left(H(\mathbf{X}), H(\mathbf{Y})\right) = 4$

Image Classification

- Pyramid match kernel (PMK)
  - Count “new” matches at each scale ‘i’

matches at this level

matches at previous level

$$N_i = \mathcal{I}\left(H_i(\mathbf{X}), H_i(\mathbf{Y})\right) - \mathcal{I}\left(H_{i-1}(\mathbf{X}), H_{i-1}(\mathbf{Y})\right)$$

Difference in histogram intersections across levels counts *number of new pairs* matched

Image Classification

- Pyramid match kernel (PMK)
  - Weight new matches at each scale ‘i’ differently
    - Finer scale → smaller bin → larger weight
    - Match at finer scale → more significant match

histogram pyramids

$$K_\Delta\left(\overbrace{\Psi(\mathbf{X}), \Psi(\mathbf{Y})}\right) = \sum_{i=0}^{L} \frac{1}{2^i} \Big(\underbrace{\mathcal{I}\left(H_i(\mathbf{X}), H_i(\mathbf{Y})\right) - \mathcal{I}\left(H_{i-1}(\mathbf{X}), H_{i-1}(\mathbf{Y})\right)}_{\text{number of newly matched pairs at level } i} \Big)$$

measure of difficulty of a match at level *i*

Image Classification

- Pyramid match kernel (PMK)
  - Weight new matches at each scale ‘i’ differently
    - Finer scale → smaller bin → larger weight
    - Match at finer scale → more significant match
    - Un-normalized kernel

$$\tilde{\mathcal{P}}_\Delta(\Psi(\mathbf{Y}),\Psi(\mathbf{Z})) = \sum_{i=0}^{L-1} w_i \Big(I(H_i(\mathbf{Y}),H_i(\mathbf{Z})) - I(H_{i-1}(\mathbf{Y}),H_{i-1}(\mathbf{Z})) \Big)$$

    - Normalized kernel (avoid favoring larger input sets)

$$\mathcal{P}_\Delta(\mathbf{P},\mathbf{Q}) = \frac{1}{\sqrt{C}} \tilde{\mathcal{P}}_\Delta(\mathbf{P},\mathbf{Q}), \text{ where } C = \tilde{\mathcal{P}}_\Delta(\mathbf{P},\mathbf{P})\, \tilde{\mathcal{P}}_\Delta(\mathbf{Q},\mathbf{Q})$$

    - Un-normalized kernel can be written as:

$$\tilde{\mathcal{P}}_\Delta(\Psi(\mathbf{Y}),\Psi(\mathbf{Z})) = w_{L-1}\ I(H_{L-1}(\mathbf{Y}),H_{L-1}(\mathbf{Z})) + \sum_{i=0}^{L-2} (w_i - w_{i+1})\ I(H_i(\mathbf{Y}),H_i(\mathbf{Z}))$$

$$w_i \geq w_{i+1} \qquad w_i = \frac{1}{d2^i}$$

Image Classification

- Pyramid match kernel (PMK)
  - Histogram intersection is a Mercer kernel [Odone 2005 IEEE TIP]
    - Map for H with
      'r' bins
      'm' features

$$\mathcal{V}(H) = \left(\underbrace{\overbrace{1,\ldots,1}^{H^{(1)}}, \overbrace{0,\ldots,0}^{m-H^{(1)}}}_{\text{first bin}}, \ldots, \underbrace{\overbrace{1,\ldots,1}^{H^{(r)}}, \overbrace{0,\ldots,0}^{m-H^{(r)}}}_{\text{last bin}} \right)$$

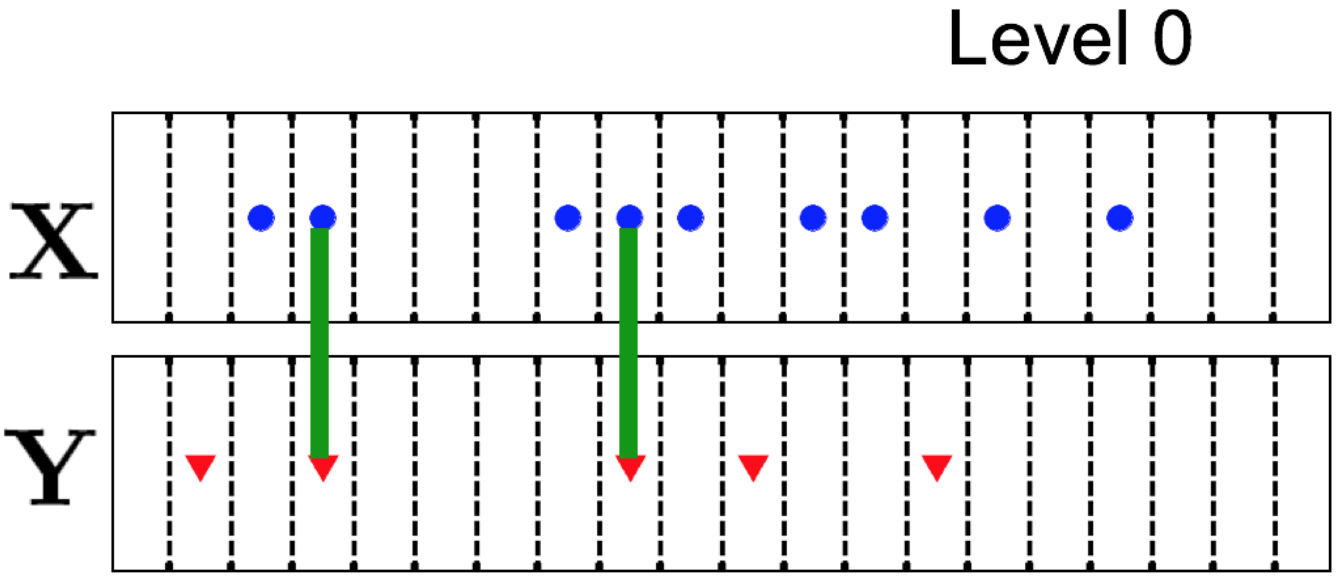
    - If num of features m_H depends on H, then $m \leftarrow \max_H (m_H)$
  - PMK is a Mercer kernel [Grauman 2007 JMLR]
    - Mercer kernel class is closed under scaling and addition
  - Normalize kernel to avoid giving large similarities to (favoring) large sets
    - Normalization still keeps it a Mercer kernel

Image Classification

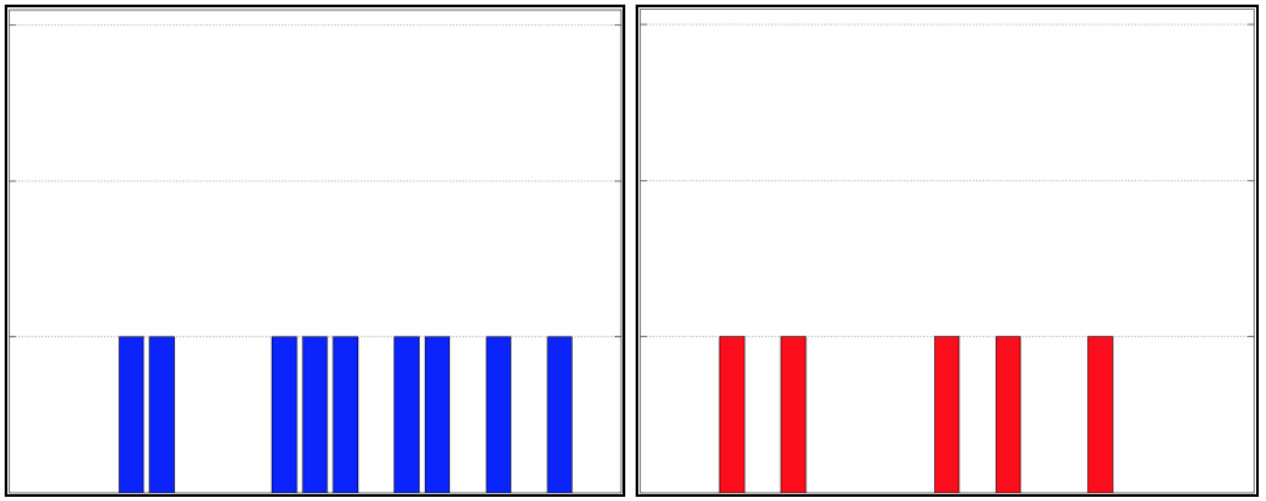
- Computational complexity O(dmL)
  - http://jmlr.csail.mit.edu/papers/volume8/grauman07a/grauman07a.pdf
  - Histogram pyramid is very sparse
  - L levels of histogram, m features in d dimensions
  - Intersection of histograms sorted by bin index takes time linear in # non-zero entries (not actual # bins)
    - Run one pointer down each of the 2 lists
    - One pointer may not advance until other is incremented down to an index at least as high as the other
  - Sorting takes linear time via radix-sort with counting sort
- Typical values
  - d: 5–12 (e.g., PCA on popular descriptors like SIFT)
  - m: 300–3000, L: 9

Image Classification

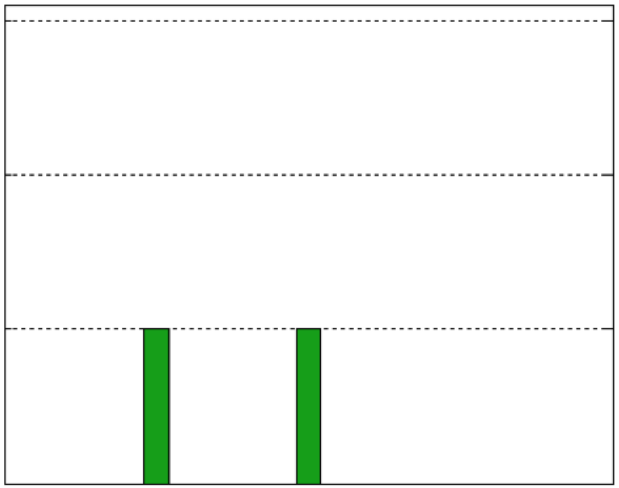
- Pyramid match kernel (PMK)



$$N_0 = 2$$
$$w_0 = 1$$



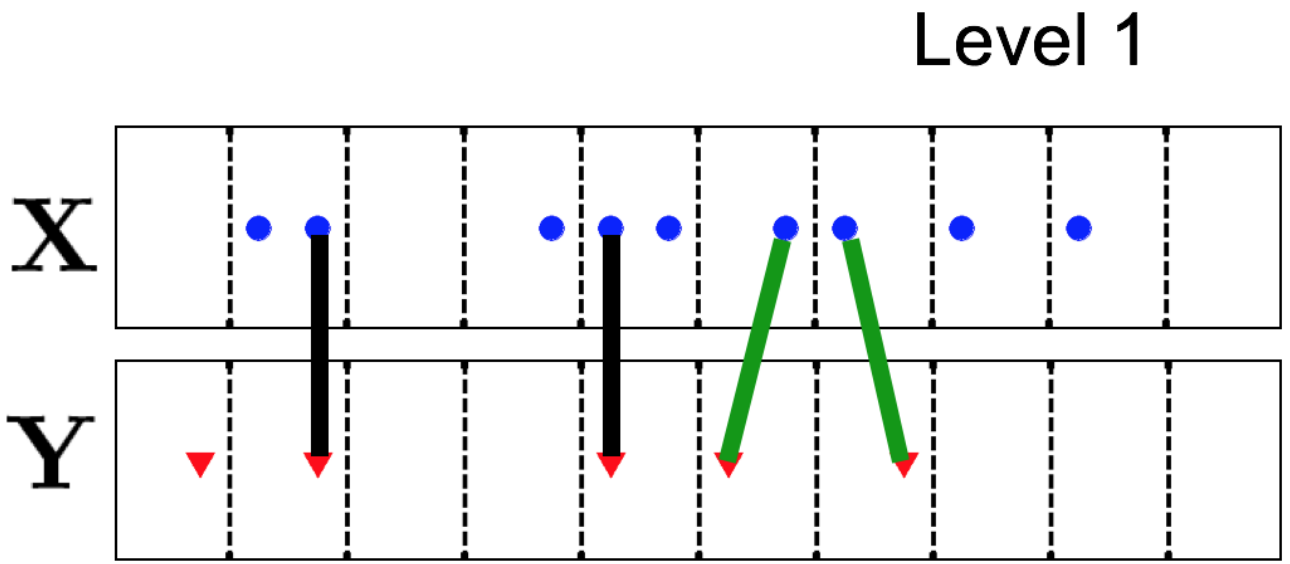
$H_0(\mathbf{X})$ $H_0(\mathbf{Y})$



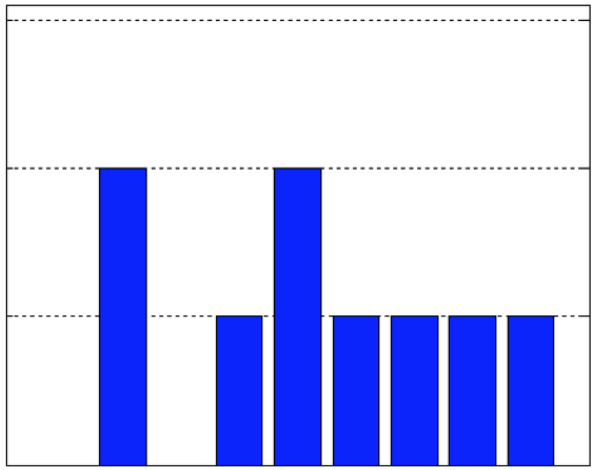
$\mathcal{I}_0 = 2$

Image Classification

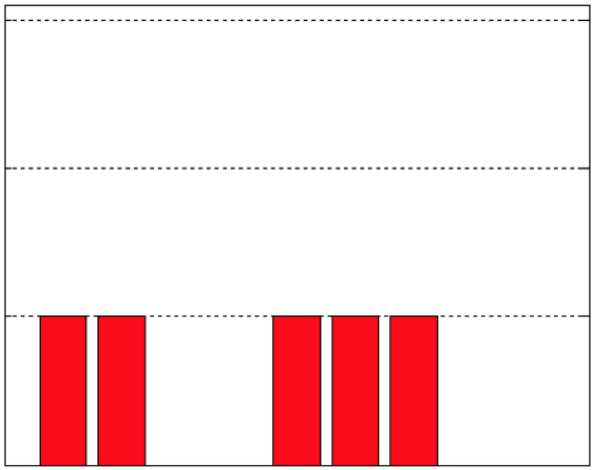
- Pyramid match kernel (PMK)



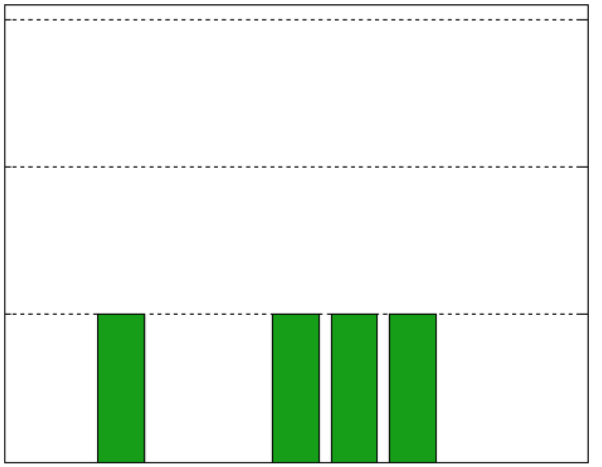
$$N_1 = 4 - 2 = 2$$
$$w_1 = \frac{1}{2}$$



$H_1(\mathbf{X})$



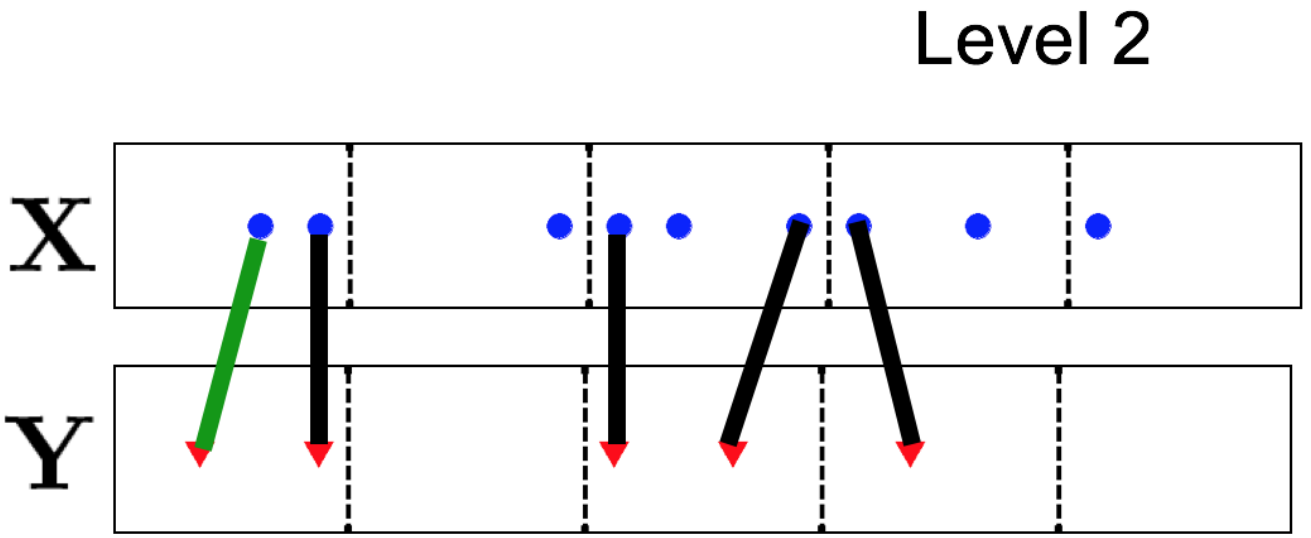
$H_1(\mathbf{Y})$



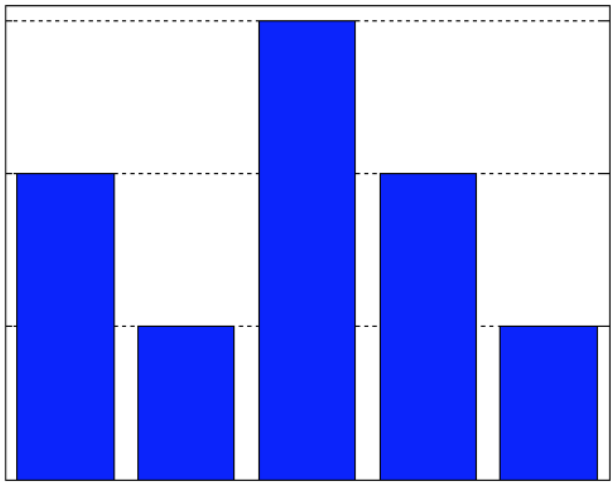
$\mathcal{I}_1 = 4$

Image Classification

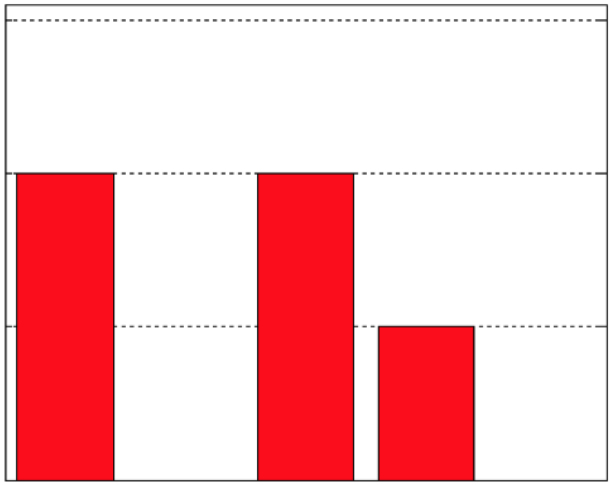
- Pyramid match kernel (PMK)



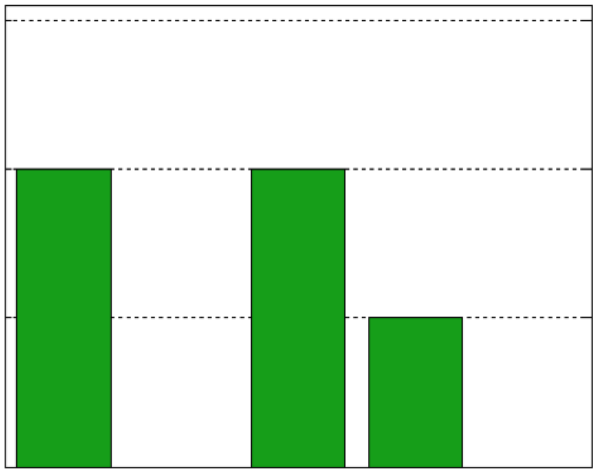
$$N_2 = 5 - 4 = 1$$
$$w_2 = \frac{1}{4}$$



$H_2(\mathbf{X})$



$H_2(\mathbf{Y})$

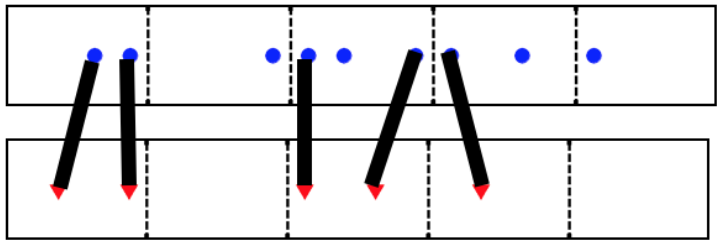


$\mathcal{I}_2 = 5$

Image Classification

- Pyramid match kernel (PMK)

pyramid match

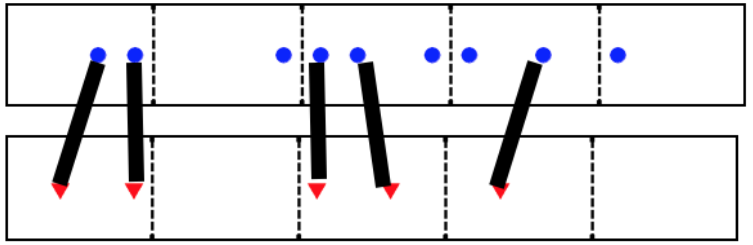


$$K_\Delta = \sum_{i=0}^{L} w_i N_i$$

$$= 1(2) + \tfrac{1}{2}(2) + \tfrac{1}{4}(1) = 3.25$$

  - Similarity S(.,.) for match at finest level = 1, consistent with PMK

optimal match



$$K = \max_{\pi:\mathbf{X}\to\mathbf{Y}} \sum_{\mathbf{x}_i\in\mathbf{X}} \mathcal{S}(\mathbf{x}_i, \pi(\mathbf{x}_i))$$

$$= 1(2) + \tfrac{1}{2}(3) = 3.5$$

Image Classification

- To alleviate quantization effects arising from discrete histogram bins
  - Combine kernel values resulting from multiple (T) pyramid matches under different multi-resolution histograms with randomly shifted bins
  - Each dimension of each of the T pyramids shifted by an amount randomly drawn from U[0,D]
    - D = maximal range along a dimension (used later)
  - Combined kernel value = $\sum_{j=1}^{T} \mathcal{P}_{\Delta}\left(\Psi_j(\mathbf{Y}), \Psi_j(\mathbf{Z})\right)$

Image Classification

- Changing “similarity” to “cost”
  - To use matching as a cost function,
    set weights as distance estimates $w_i = d\ 2^i$
    - d = dimension of space = dimension of bin

Image Classification

- Comparison with optimal partial matching
  - Partial matching = $\mathcal{M}(\mathbf{X}, \mathbf{Y}; \pi) = \{(\mathbf{x}_1, \mathbf{y}_{\pi_1}), \ldots, (\mathbf{x}_m, \mathbf{y}_{\pi_m})\}$
  - Cost of partial matching = $C(\mathcal{M}(\mathbf{X}, \mathbf{Y}; \pi)) = \sum_{\mathbf{x}_i \in \mathbf{X}} ||\mathbf{x}_i - \mathbf{y}_{\pi_i}||_1$
  - Optimal partial matching: $\pi^* = \underset{\pi}{\operatorname{argmin}}\, C(\mathcal{M}(\mathbf{X}, \mathbf{Y}; \pi))$
  - Proposition 1:

$$\begin{aligned} C(\mathcal{M}(\mathbf{X}, \mathbf{Y}; \pi^*)) &\leq \sum_{i=0}^{L-1} d2^i \Big(I(H_i(\mathbf{X}), H_i(\mathbf{Y})) - I(H_{i-1}(\mathbf{X}), H_{i-1}(\mathbf{Y})) \Big) \\ &\leq \mathcal{P}_\Delta(\mathbf{X}, \mathbf{Y}), \end{aligned}$$

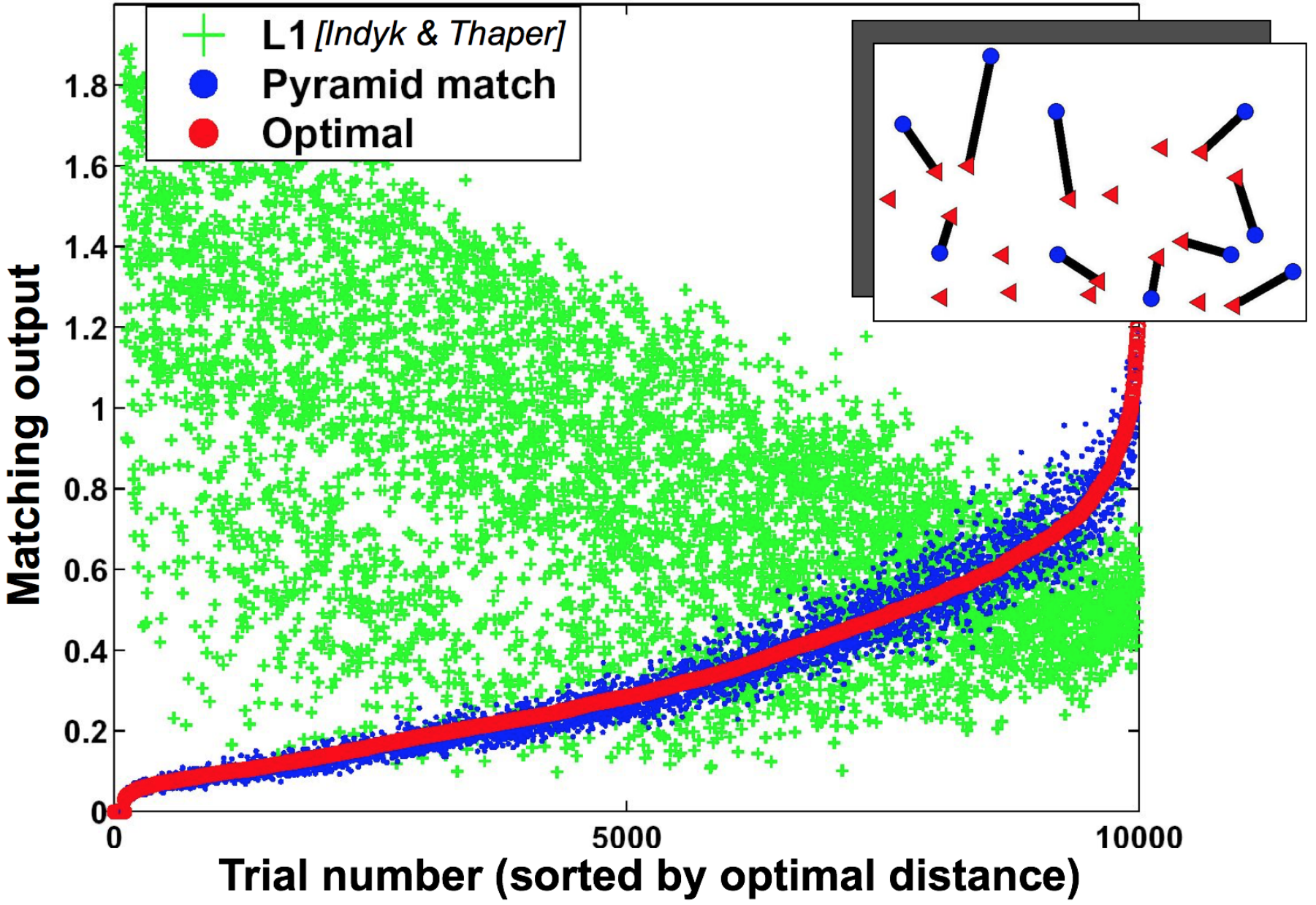
  - Proposition 2:

There is a constant C such that for any two point sets $\mathbf{X}$ and $\mathbf{Y}$, where $|\mathbf{X}| \leq |\mathbf{Y}|$, if the histogram bin boundaries are shifted randomly in the same way when computing $\Psi(\mathbf{X})$ and $\Psi(\mathbf{Y})$, then the expected value of the pyramid match cost is bounded:

$$E[\mathcal{P}_\Delta(\mathbf{X}, \mathbf{Y})] \leq \Big(C \cdot d \log D + d \Big)\, C(\mathcal{M}(\mathbf{X}, \mathbf{Y}; \pi^*)).$$

Image Classification

- Pyramid match kernel (PMK)



100 sets with 2D points, cardinalities vary between 5 and 100

Image Classification

- L1 distance over histograms
  - Directly related to histogram intersection **if** histograms have equal masses [Swain 1991 IJCV]
    - Doesn’t adapt to partial matching

$$I(H(\mathbf{Y}), H(\mathbf{Z})) = m - \frac{1}{2}||H(\mathbf{Y}) - H(\mathbf{Z})||_{L_1}, \quad \text{if } m = |\mathbf{Y}| = |\mathbf{Z}|$$

Image Classification

- Pyramid match kernel (PMK)
  - ETH-80 database, 8 object classes
    - Features
      - Harris detector
      - **PCA**-SIFT descriptor, d=10
    - 84% recognition rate
  - Caltech database, 101 object classes
    - Features:
      - SIFT detector
      - **PCA**-SIFT descriptor, d=10
    - 30 training images / class
    - 43% recognition rate
    - 0.002 seconds per match

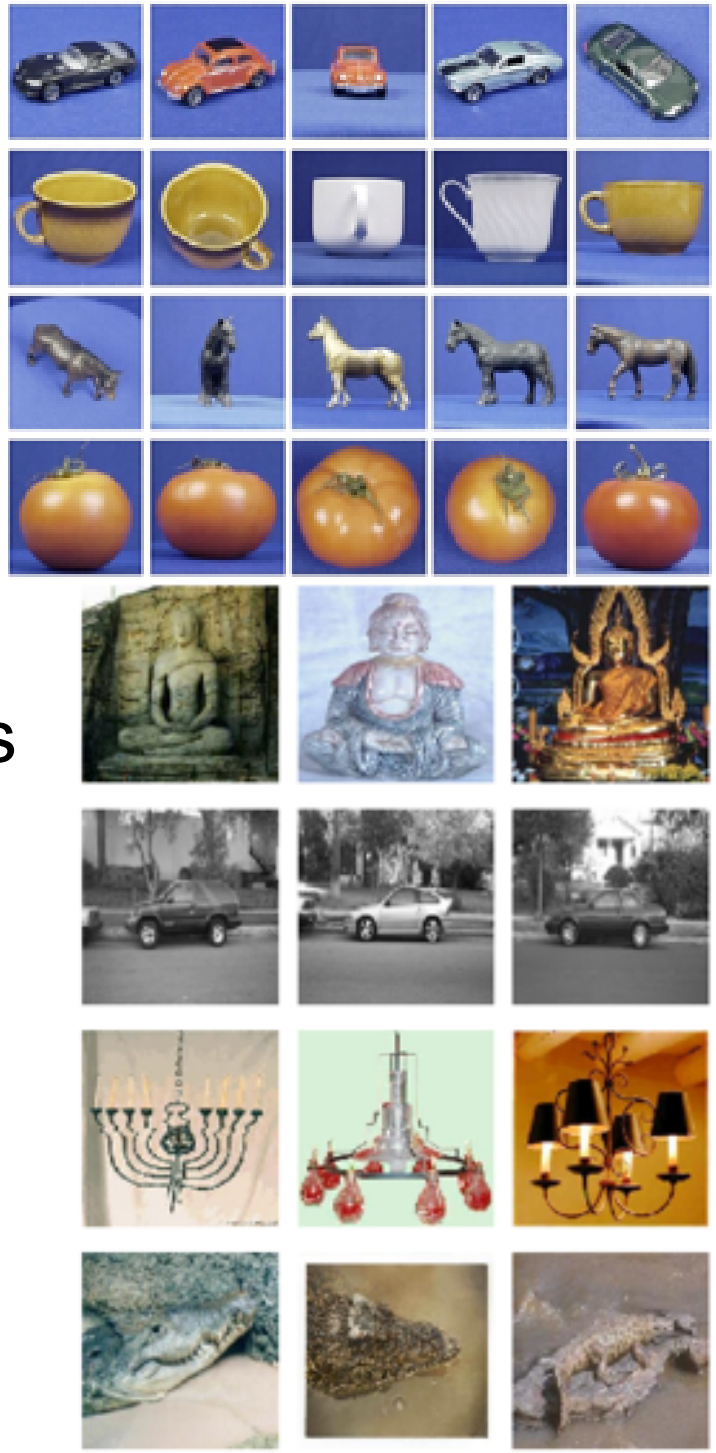


Image Classification

- More variations
  - Using a single histogram feature of the entire image results in loss of spatial information

  - So, divide image into regions and take the set of histogram features of each region

Image Classification

- Multiple modeling possibilities for image representation

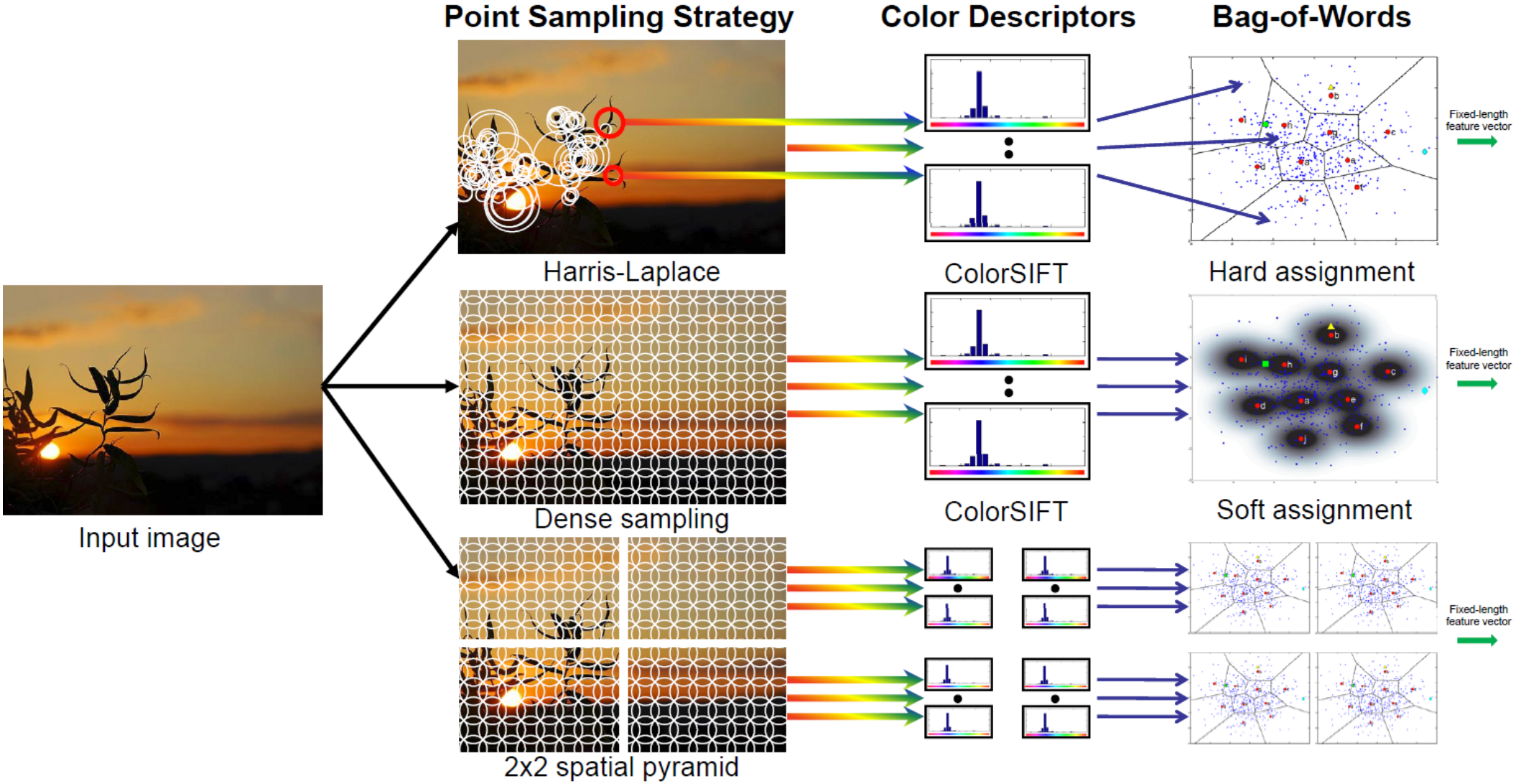


Image Classification

- Use SIFT descriptors of 16x16 patches computed over a grid with spacing 8 pixels
  - Dense features (vs. keypoint) better for scene recognition; better capture uniform regions sky, water bodies, road

Image Classification

- Successively finer spatial tessellation of the image
  - **Spatial pyramid** [Lazebnik 2006 CVPR]
    - Partition image, hierarchically, into regions
    - Within each region, compute dense / point descriptors
    - Code descriptors (kmeans, soft-memberships, sparse coding)
    - Build histogram of codes, one for each region

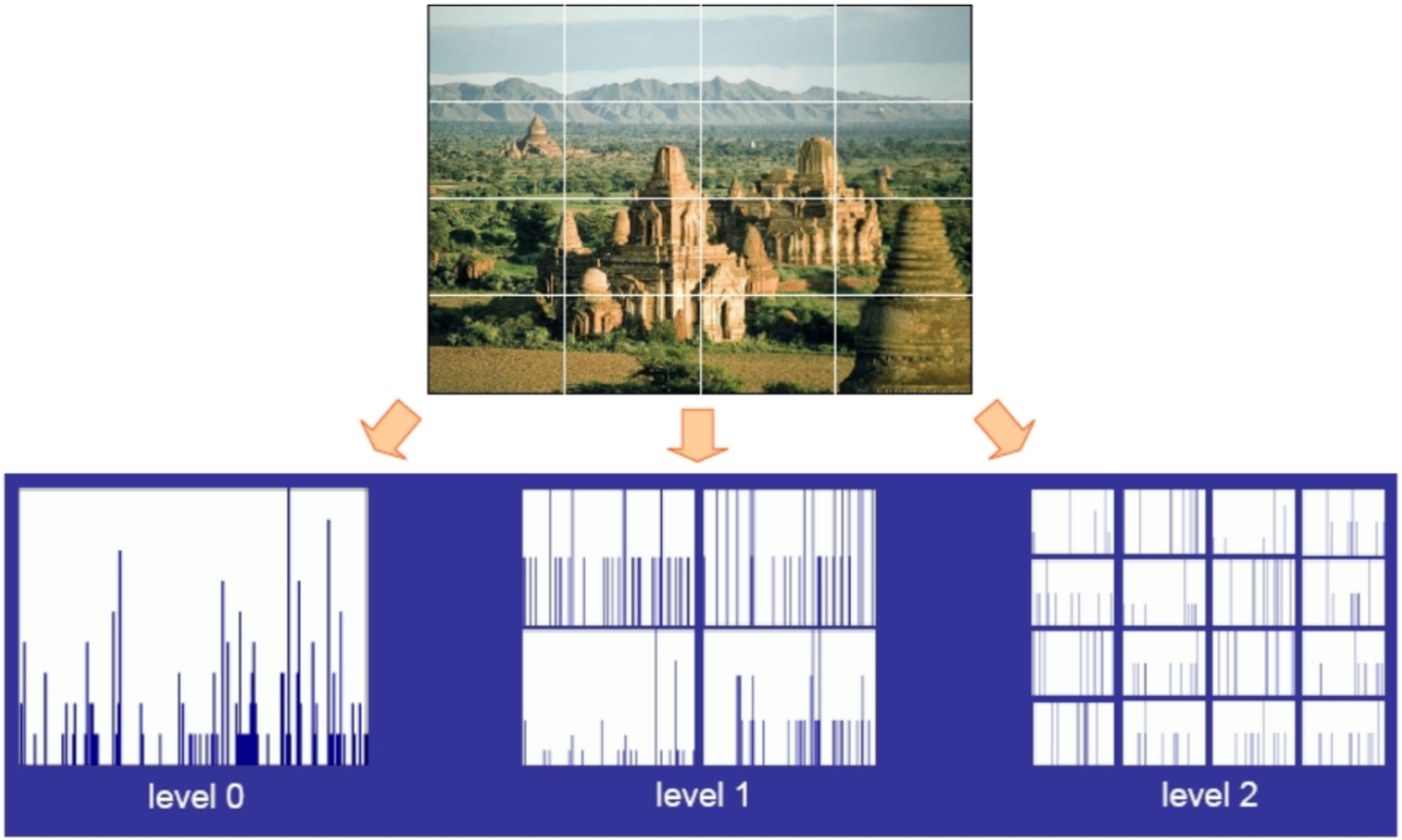


Image Classification

- Successively finer spatial tessellation of the image
  - **Spatial pyramid** [Lazebnik 2006 IEEE CVPR]

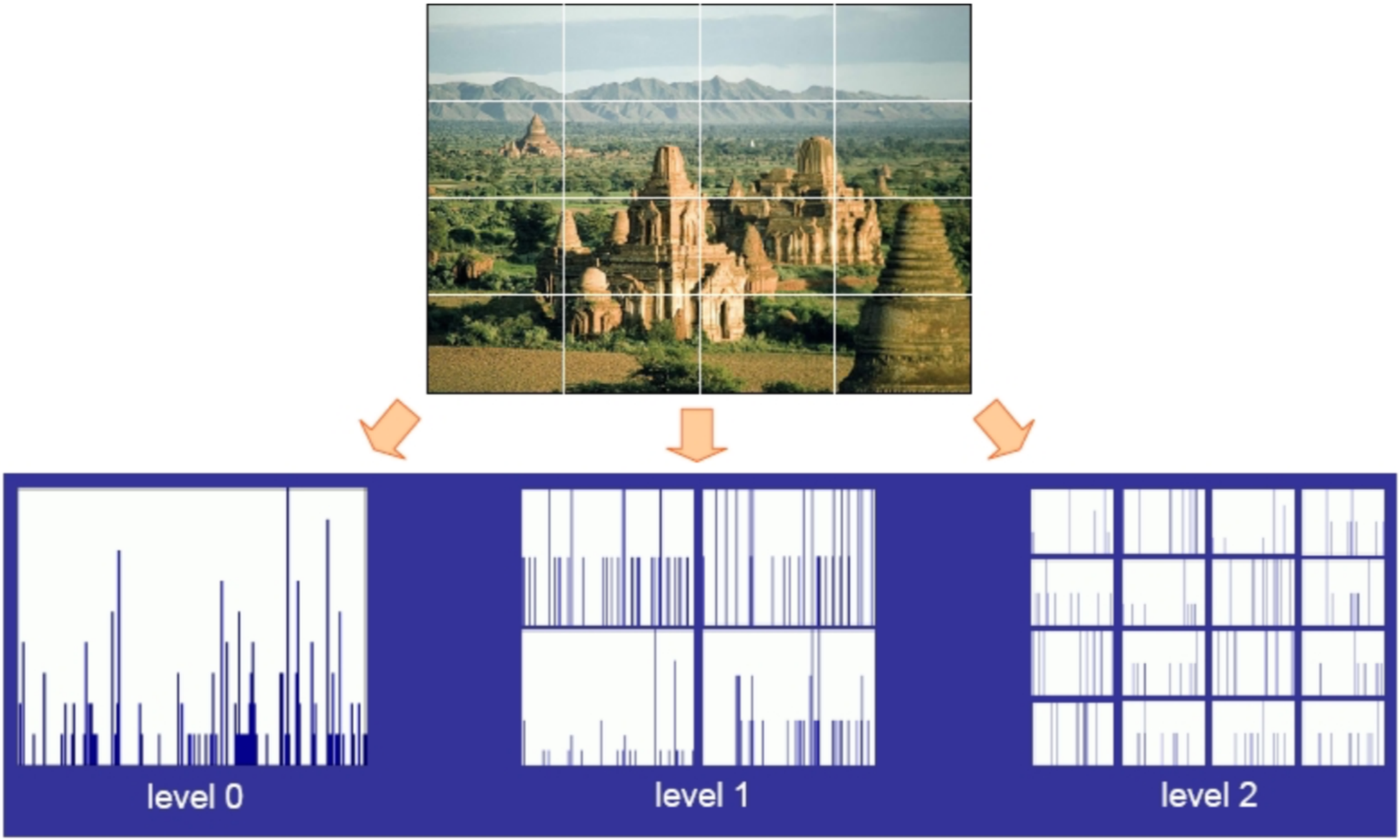
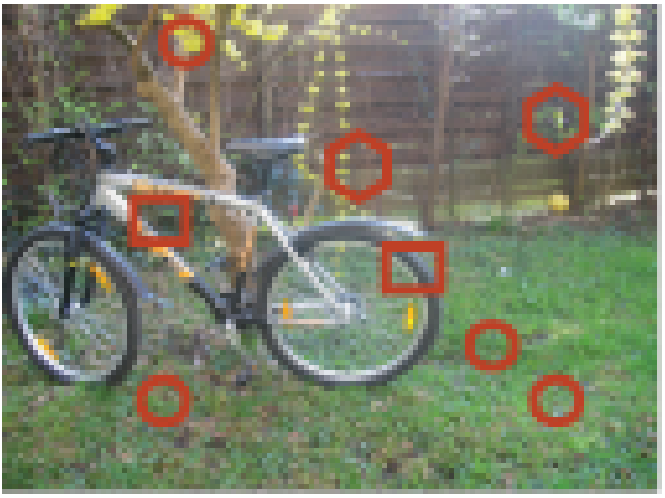
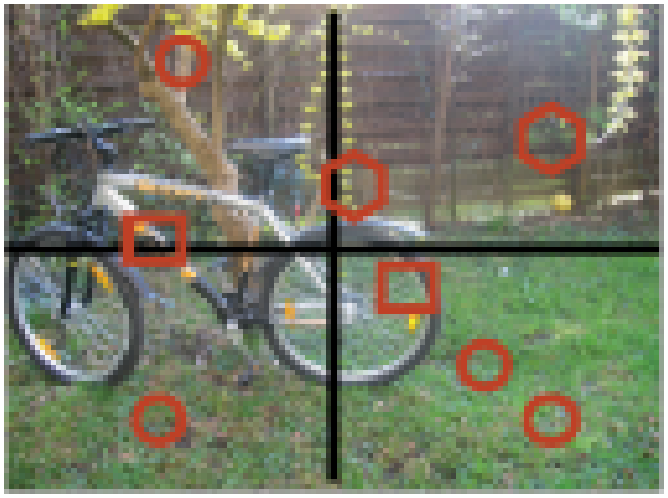
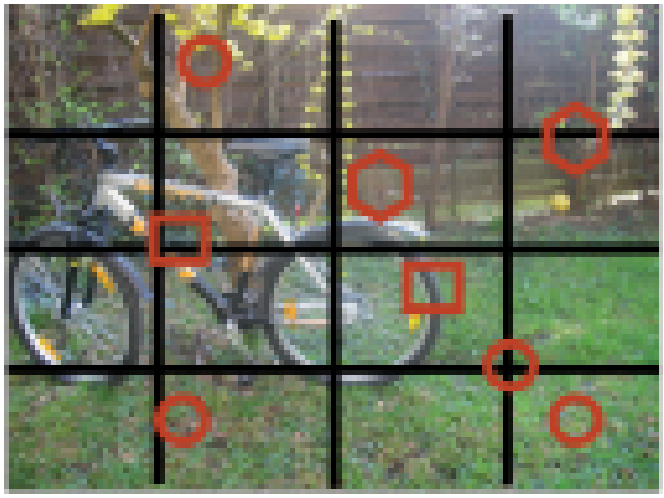


Image Classification

- Spatial pyramid with “point” (non-dense) features



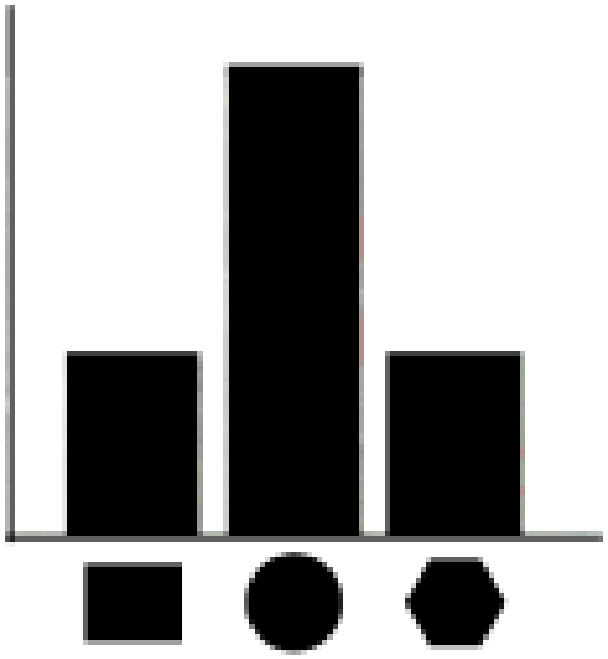


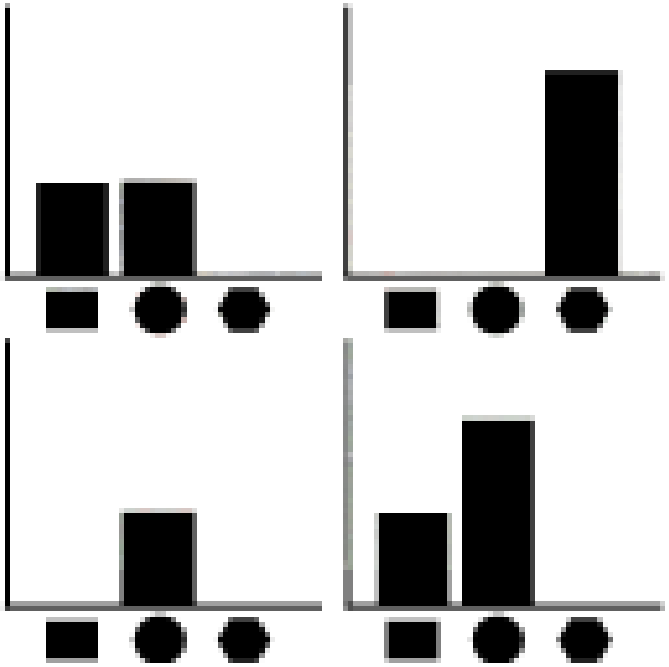


Level = 0

Level = 1

Level = 2





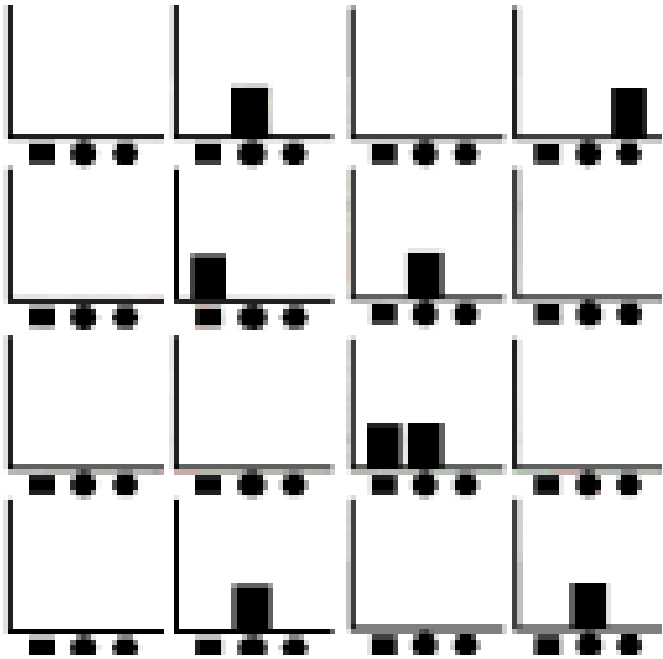


Image Classification

- Weighting histogram (matches) at different scales
  - Matches at finer scales carry more weight

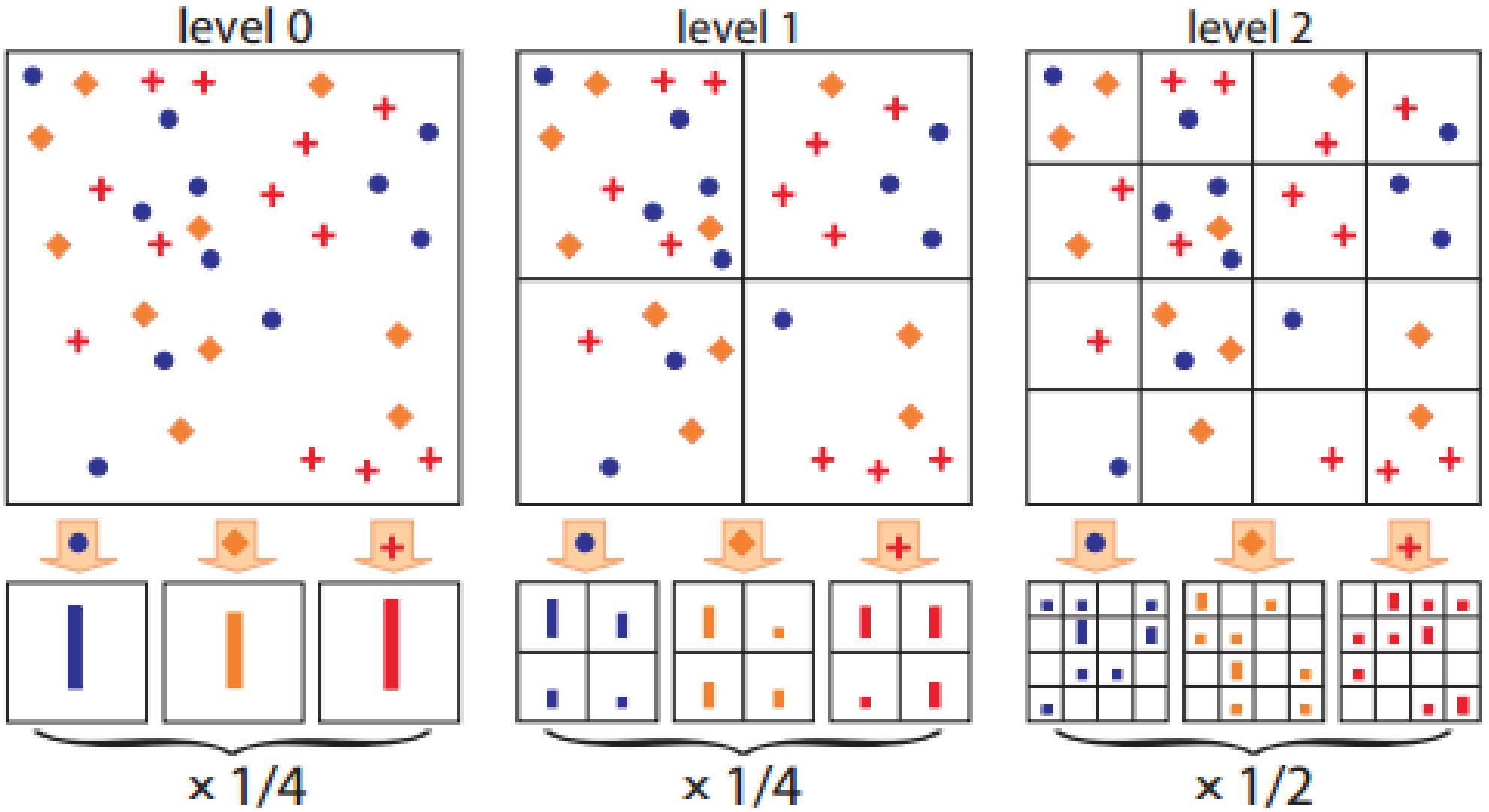


Image Classification

- How to compute similarity ?
  - Quantize all feature vectors into M codewords
  - Compute M-bin histogram for each region
  - Assume that only codewords of the same type can be matched to each other
  - Given: M codewords, L+1 levels, spatial pyramids X & Y
  - Similarity between X and Y is:

$$K^L(X,Y) = \sum_{m=1}^{M} \kappa^L(X_m, Y_m)$$

    - where

$$\kappa^L(X,Y) = \mathcal{I}^L + \sum_{\ell=0}^{L-1} \frac{1}{2^{L-\ell}} \left(\mathcal{I}^\ell - \mathcal{I}^{\ell+1}\right)$$

$$= \frac{1}{2^L}\mathcal{I}^0 + \sum_{\ell=1}^{L} \frac{1}{2^{L-\ell+1}} \mathcal{I}^\ell .$$

    - where

$$\mathcal{I}^\ell = \mathcal{I}(H_X^\ell, H_Y^\ell) = \sum_{i=1}^{D} \min\left(H_X^\ell(i), H_Y^\ell(i)\right)$$

Image Class

- Histogram counts for **1 codeword** at level ‘l’ spatial bin ‘i’ of pyramid X

$$H_X^\ell(i)$$

  - Example codeword occurs often in sky region

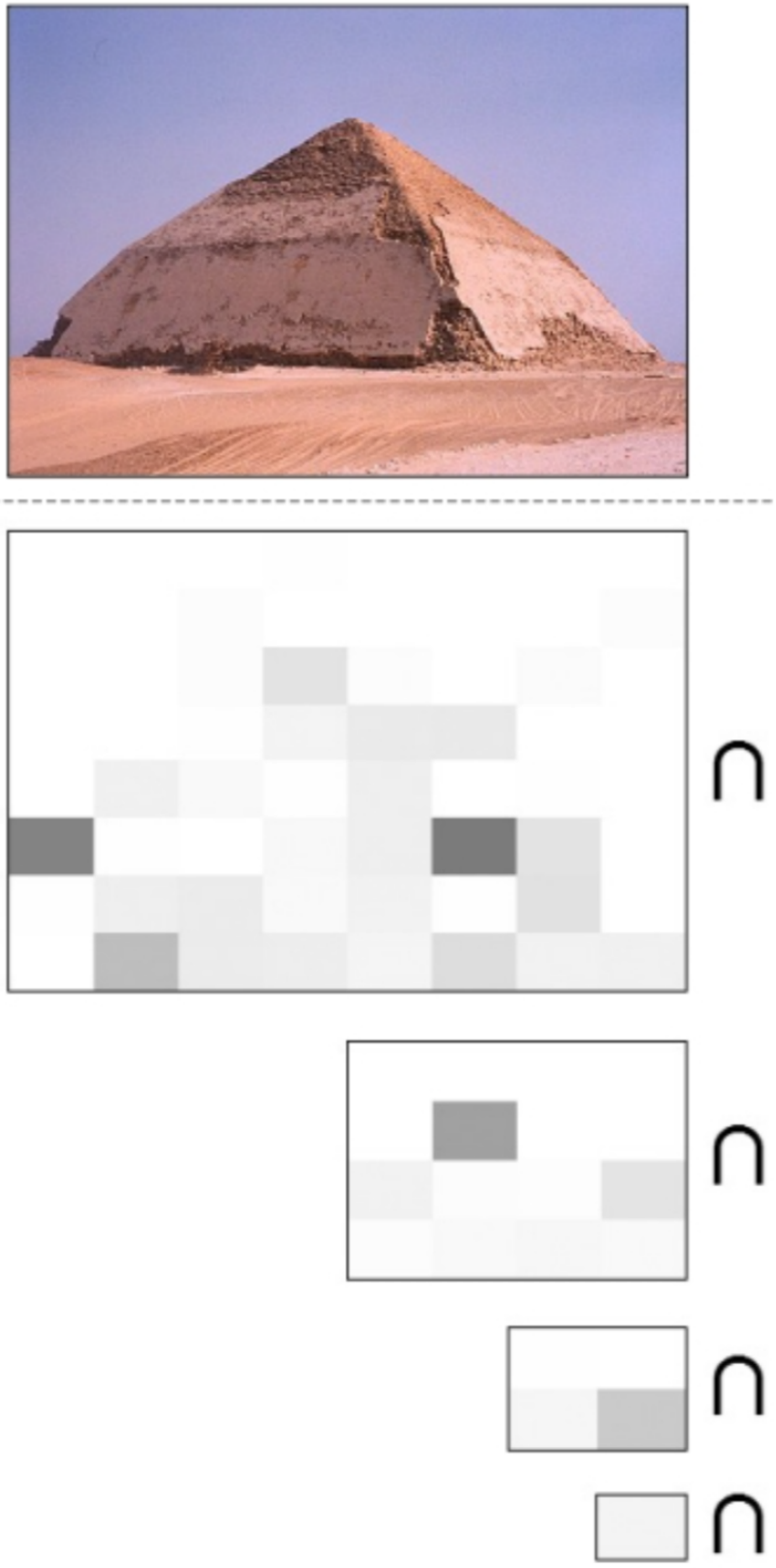
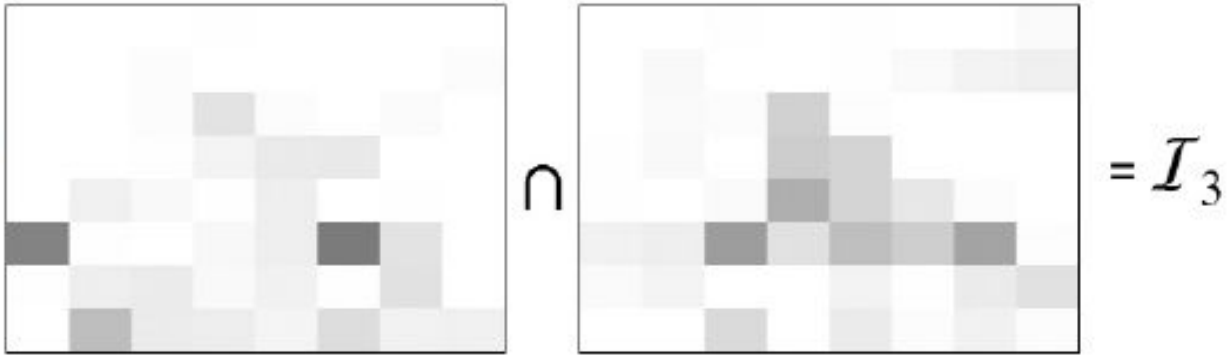


Image Classification

Original images

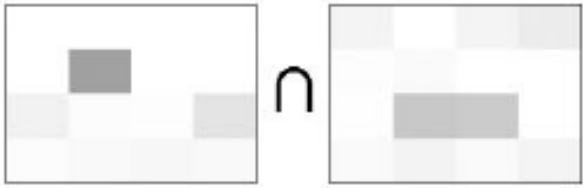
Feature histograms:

Level 3



$= \mathcal{I}_3$

Level 2



$= \mathcal{I}_2$

Level 1

$= \mathcal{I}_1$

Level 0

$= \mathcal{I}_0$

Total weight (value of *pyramid match kernel*): $\mathcal{I}_3 + \frac{1}{2}(\mathcal{I}_2 - \mathcal{I}_3) + \frac{1}{4}(\mathcal{I}_1 - \mathcal{I}_2) + \frac{1}{8}(\mathcal{I}_0 - \mathcal{I}_1)$

Image Classification

- Normalize all histograms by the total weight of all features in the image
  - Enforces the total # features in all images to be same
- SPM is a Mercer kernel:
  - Pyramid matching kernel (PMK) is a Mercer kernel
    - Histogram intersection is a Mercer kernel
  - Addition+scaling of Mercer kernels is a Mercer kernel
- Typically, M=200 (codewords), L=2 (levels)
- When L=0, SPM reduces to BoW model with histogram-intersection similarity
- Histograms are very sparse → fast computation

Image Class

- Scene recognition performance

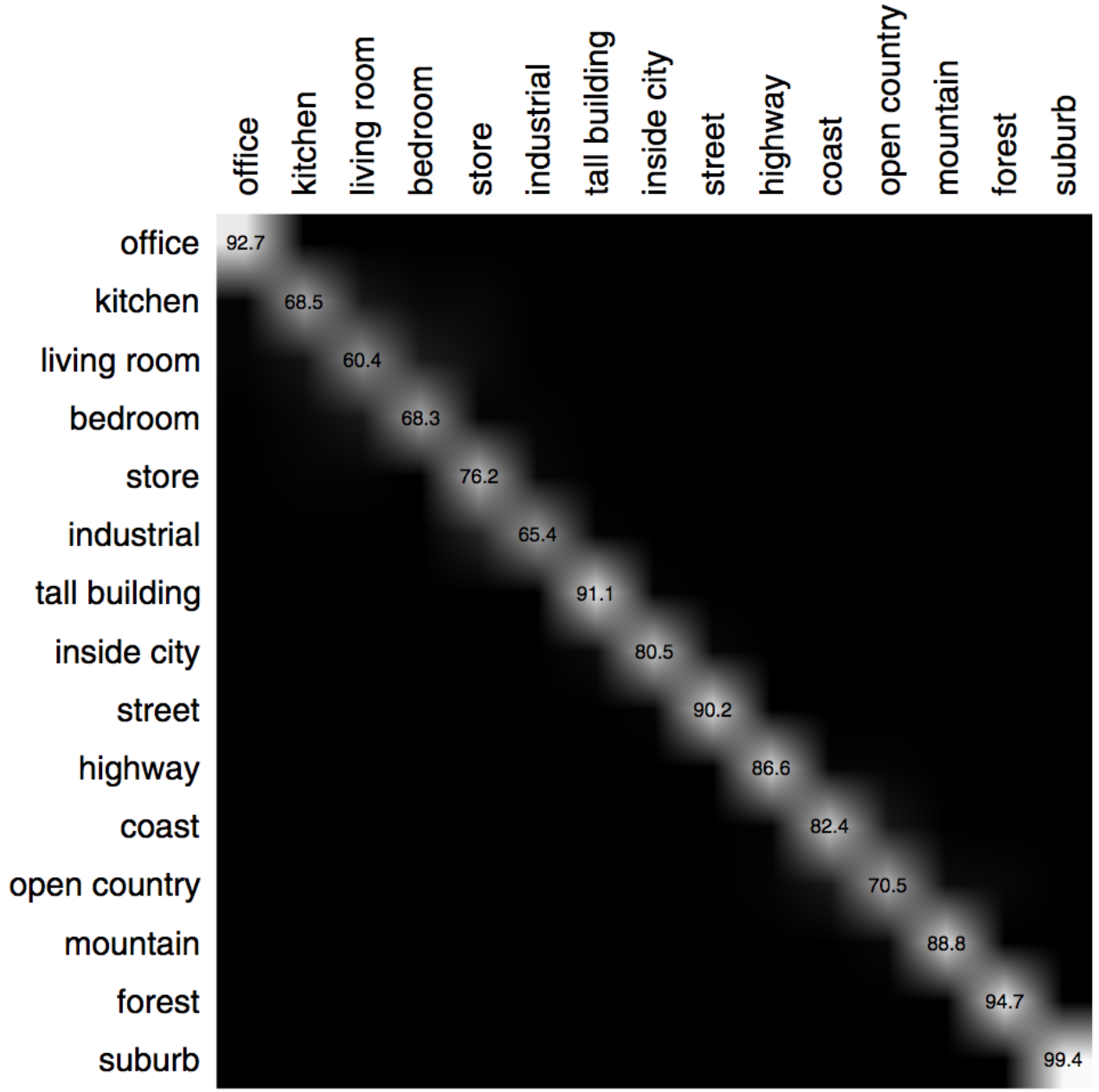


Figure 3. Confusion table for the scene category dataset. Average classification rates for individual classes are listed along the diagonal. The entry in the ith row and jth column is the percentage of images from class i that were misidentified as class j.

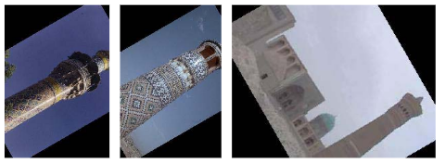
Image Classification

- Performance on scene retrieval queries

Figure 4. Retrieval from the scene category database. The query images are on the left, and the eight images giving the highest values of the spatial pyramid kernel (for $L = 2, M = 200$) are on the right. The actual class of incorrectly retrieved images is listed below them.

Image Classification

- Scene recognition performance on another dataset


minaret (97.6%)


windsor chair (94.6%)


joshua tree (87.9%)


okapi (87.8%)


cougar body (27.6%)


beaver (27.5%)


crocodile (25.0%)


ant (25.0%)

Figure 5. Caltech-101 results. Top: some classes on which our method ($L = 2, M = 200$) achieved high performance. Bottom: some classes on which our method performed poorly.

Image Classification

- Generalized **deformable** spatial pyramid [Hur 2015 CVPR]
  - Based on [Kim-...-Grauman 2013 CVPR]

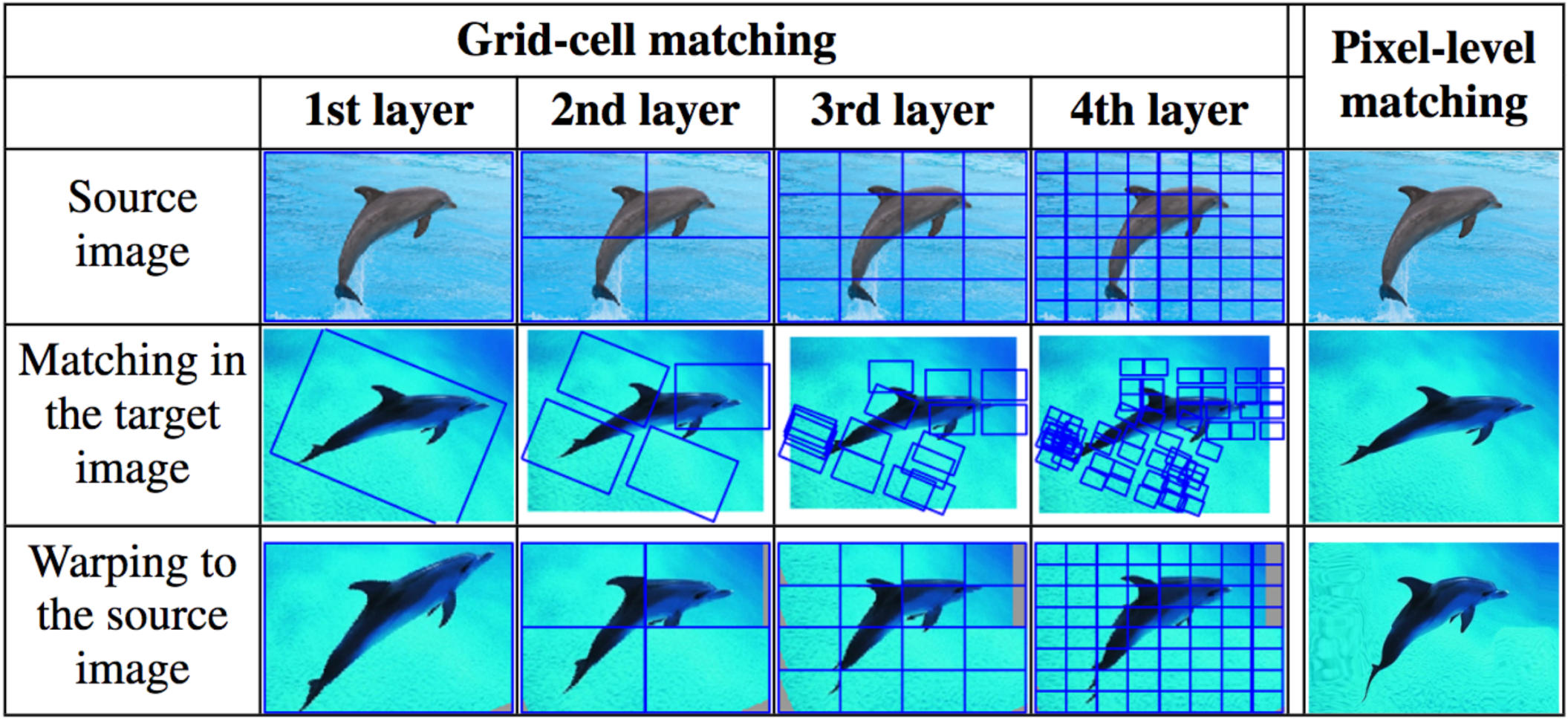
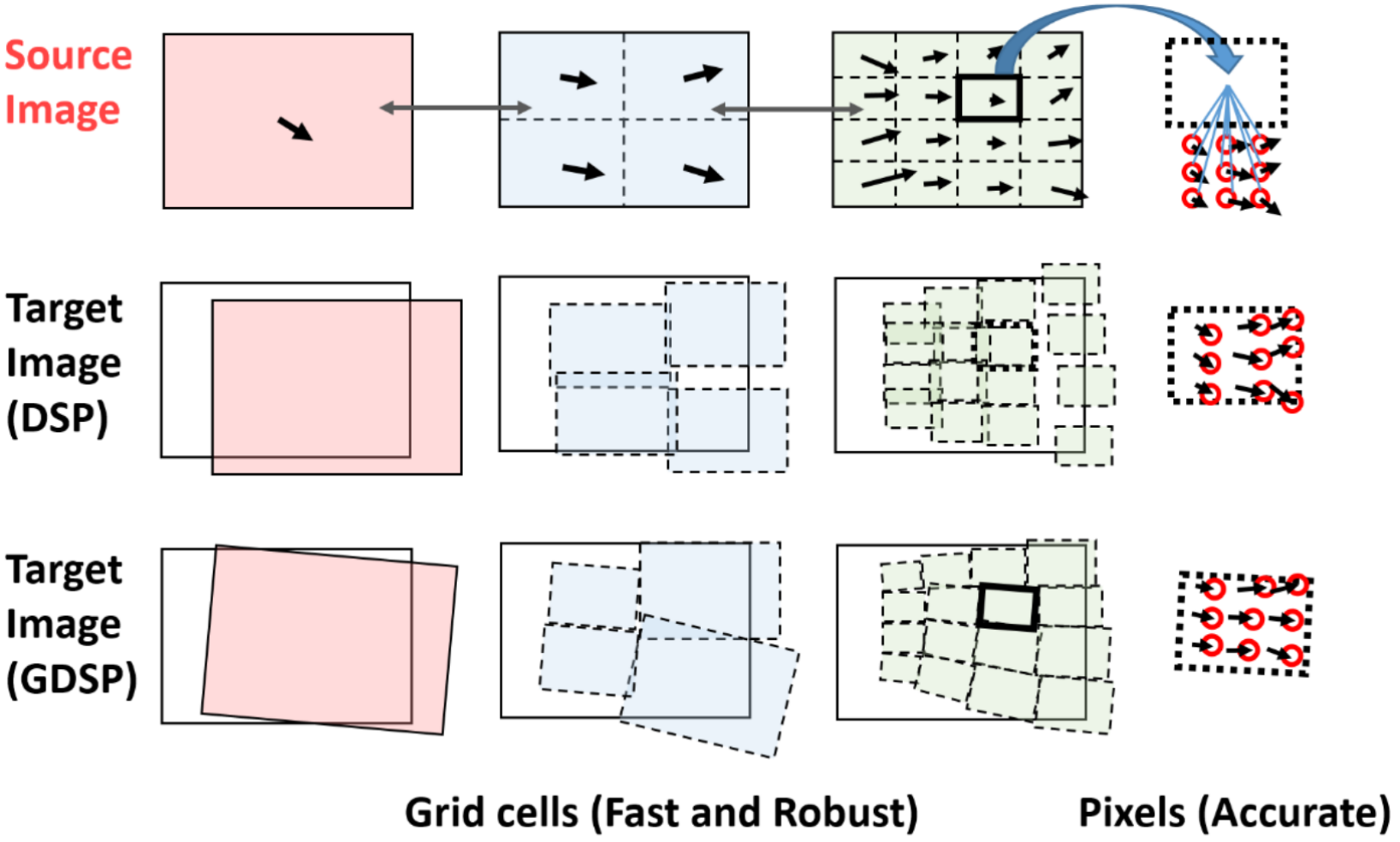


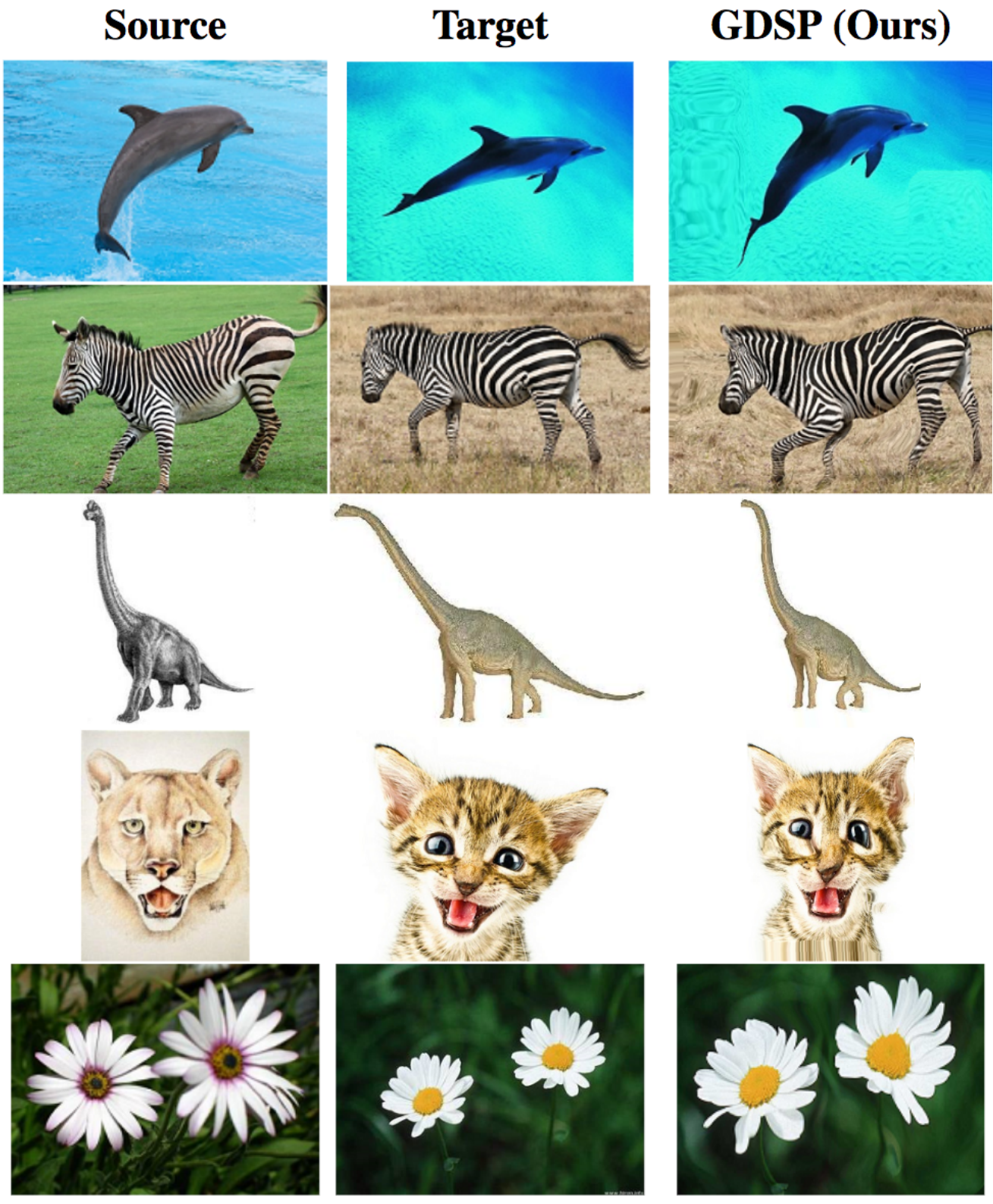
Image Classification

- Generalized **deformable** spatial pyramid [Hur 2015 CVPR]
  - Based on [Kim-...-Grauman 2013 CVPR] that had only axis-aligned shifts



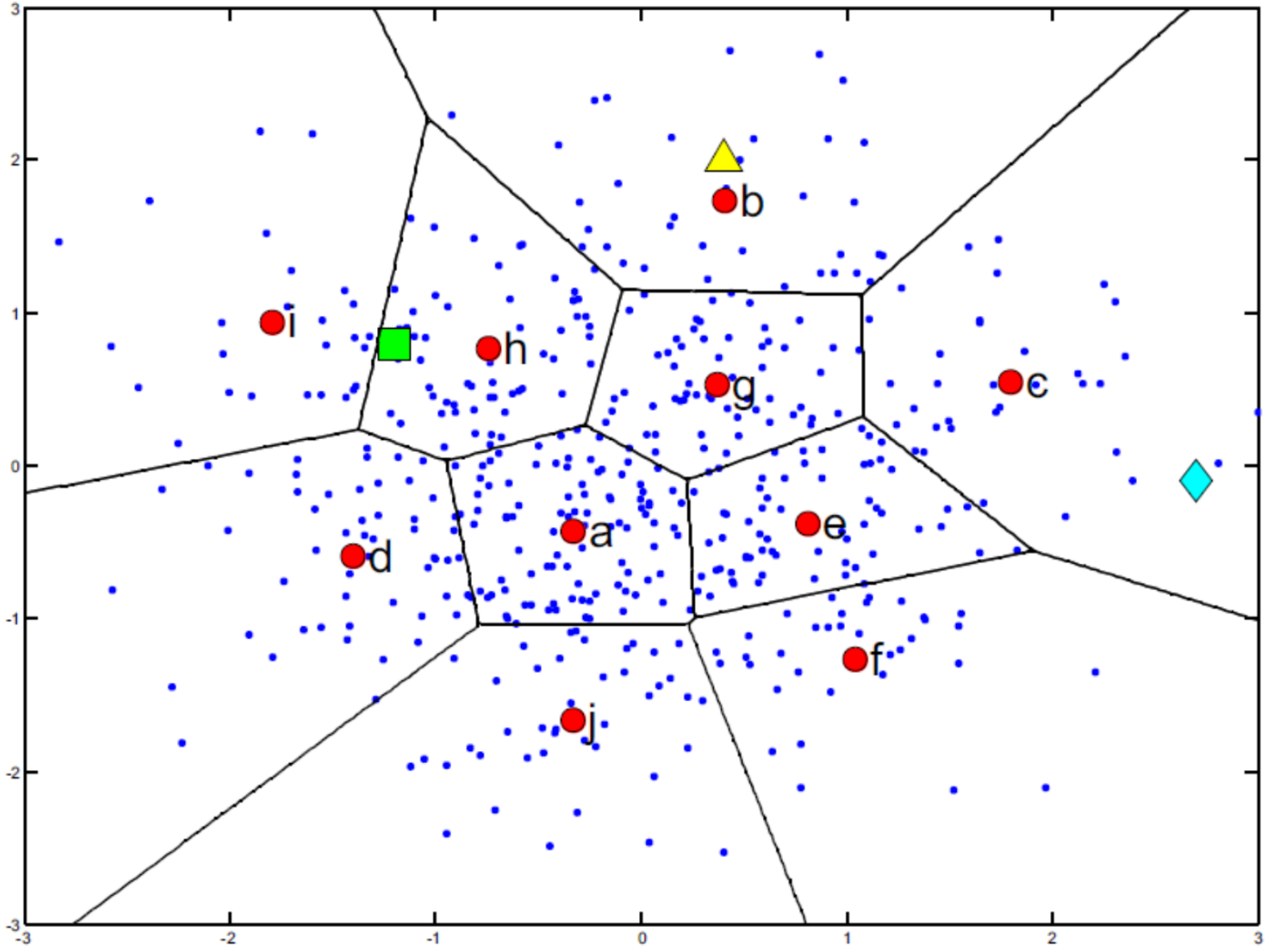
Image

- Generalized deforr
  [Hur 2015 CVPR]
  - Based on [Kim-...-(

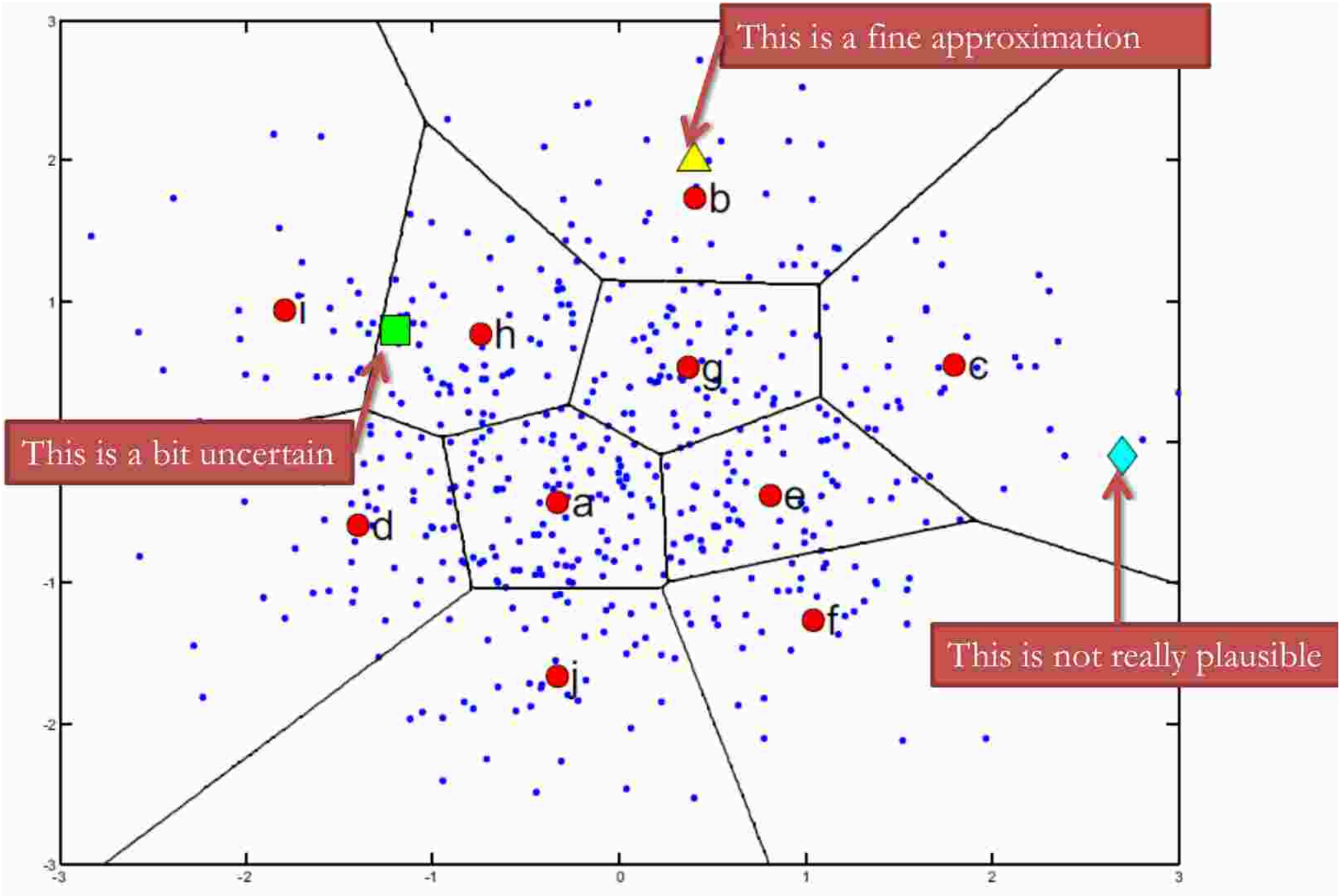


Coding

- Problems with quantization (kmeans) in Bag of Visual Words model



Coding


This is a fine approximation
This is a bit uncertain
This is not really plausible
a
b
c
d
e
f
g
h
i
j
3
2
1
-1
-2
-3
-3
-2
-1
0
1
2
3

Kernel Coding

- A strategy
  - Place Gaussian kernel at each cluster representative
  - For each descriptor,
    find probability of descriptor under Gaussian
  - Store a vector of such probabilities → descriptor
- How to define the clusters ?
  - Replace kmeans by soft clustering

Kernel Coding

- Conventional coding (quantization)
  - Code vector “y” contains component y_i, where only one y_i is non-zero (= 1)

$$C = \left\{ \boldsymbol{y} \mid \boldsymbol{y} \in \{0,1\}^N, \sum_i \boldsymbol{y}_i = 1 \right\}$$

- Soft coding (with kernels)
  - Code vector “y” contains component y_i, where all components sum to 1

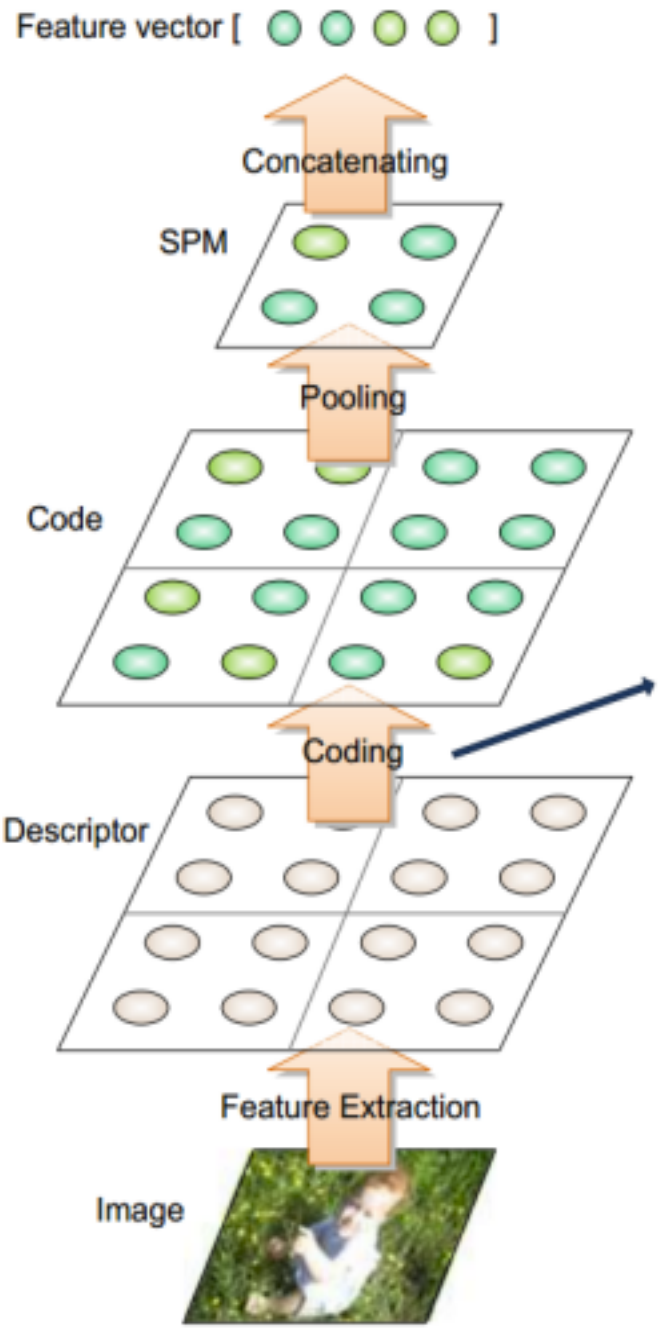
$$\boldsymbol{y}_i = \frac{\exp\left(-\sigma \|\boldsymbol{x} - \boldsymbol{d}_i\|_2^2\right)}{\sum_j \exp\left(-\sigma \|\boldsymbol{x} - \boldsymbol{d}_j\|_2^2\right)}$$

Local Coding

- Gaussian-kernel based coding doesn't give "sparse" code vector
  - Many components are *small*, but *still non-zero*
- For the sake of *robustness*,
  we need only few non-zero components in the code
- **Local coding**
- **Locality-constrained Linear Coding (LcLC)**

Local Coding

- LcLC [Wang 2010 CVPR]
  - Each descriptor is well modeled as a linear combination of (few, local) centroids



LLC Coding process

Step 3:
c_i is an Mx1 vector with K non-zero elements whose values are the corresponding c^* of step 2

input: x_i → code: c_i

Step 2:
Reconstruct x_i using B_i

$$c^* = \underset{c}{\operatorname{argmin}} \| x_i - c_i^T B_i \|^2$$

$$\text{st. } \sum_j^K c_j = 1$$

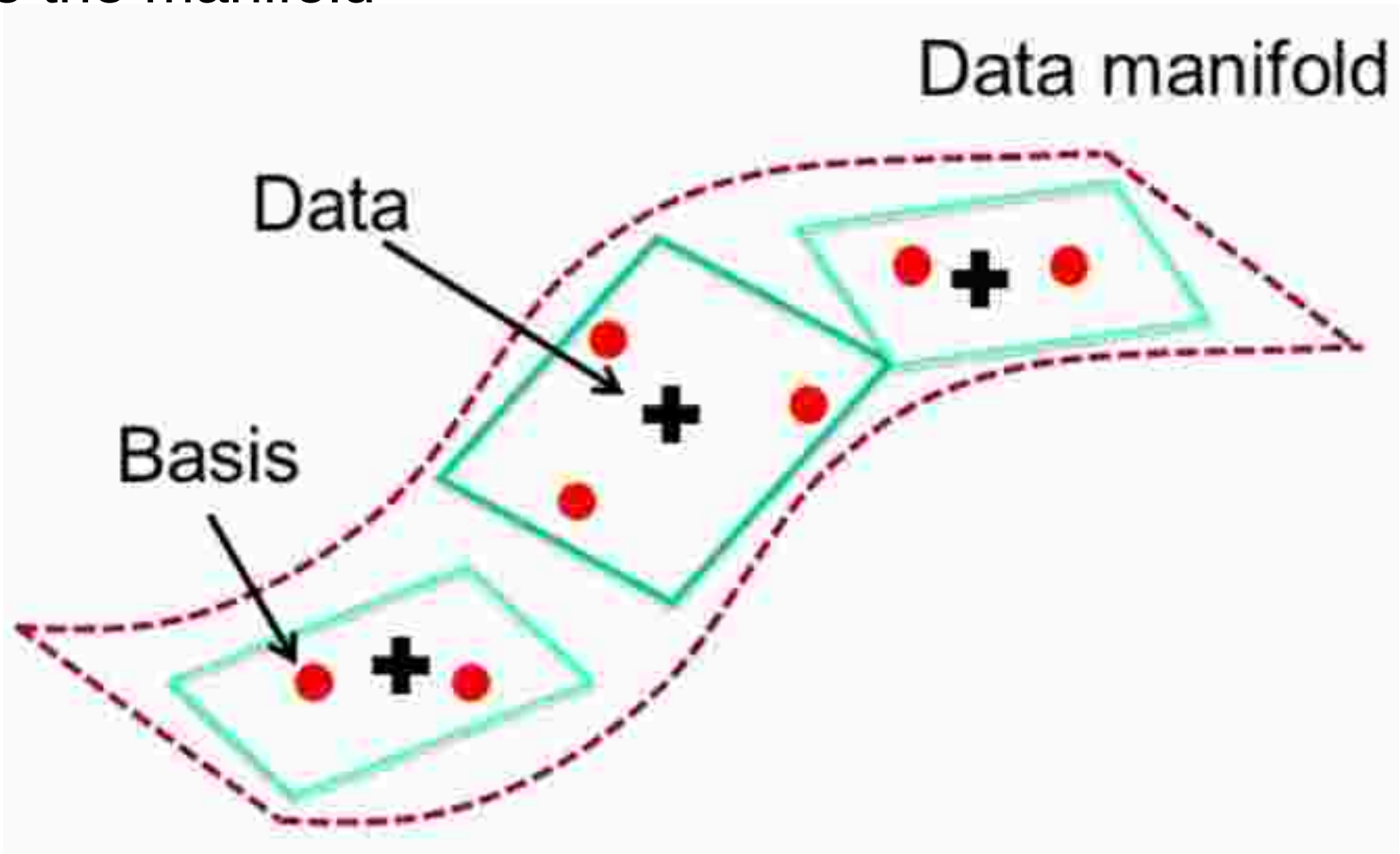
input: x_i

codebook: $B=\{b_j\}_{j=1,\ldots,M}$

Step 1:
Find K-Nearest Neighbors of x_i, denoted as B_i

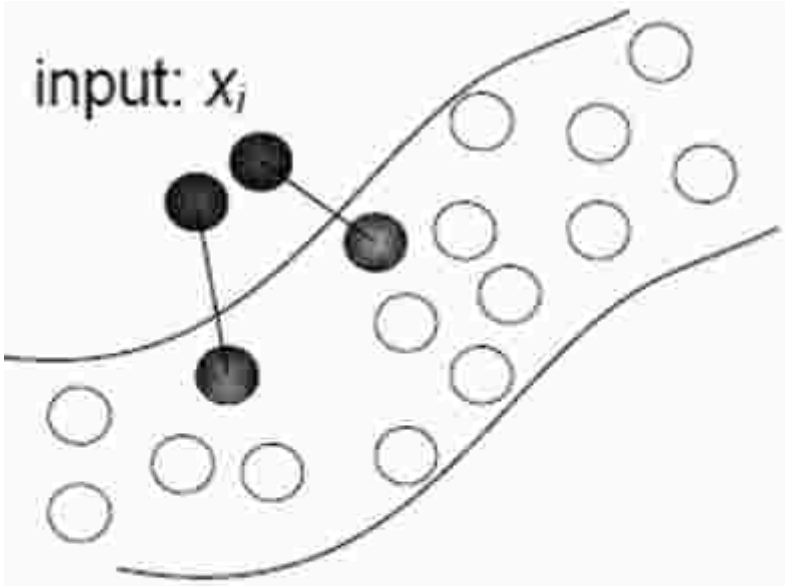
Local Coding

- When data lies on a manifold,
  linear combination makes sense only for a locality
  - Linear combination of far-off centroids can take us **outside** the manifold



Coding

- Revisit …
- (1/3) **Vector quantization (VQ)**



VQ

A set of D-dimensional local descriptors

$$X = [x_1, x_2, \ldots, x_N] \in R^{D \times N}$$

Codebook with M entries

$$B = [b_1, b_2, \ldots, b_M] \in R^{D \times M}$$

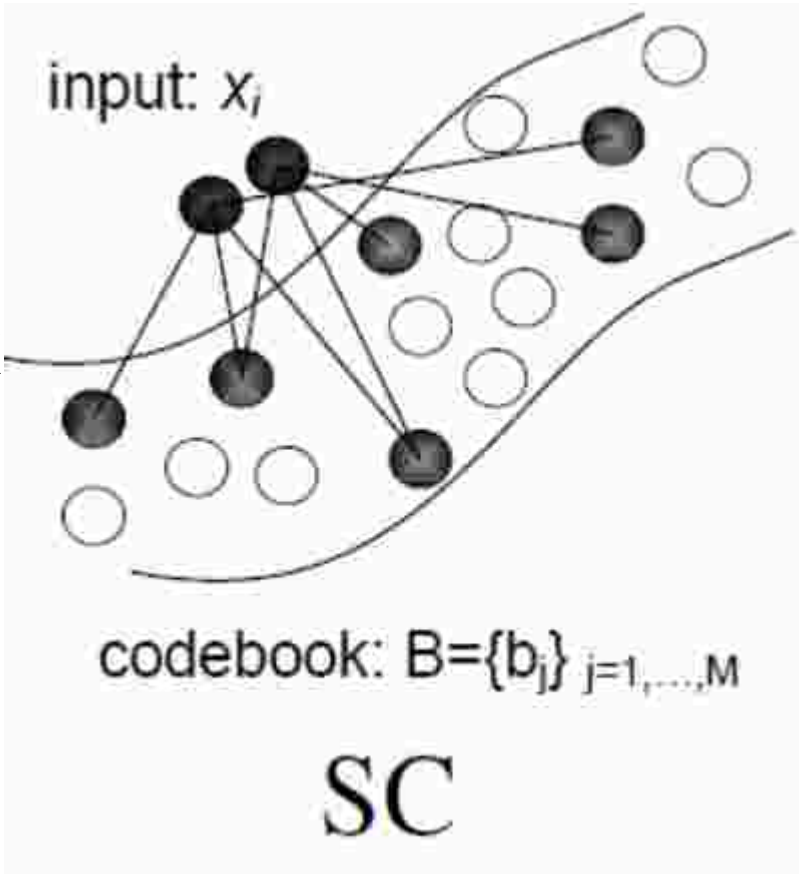
$$\min_C \sum_{i=1}^{N} \|x_i - Bc_i\|^2$$

$$s.t. \|c_i\|_{l^0} = 1, \|c_i\|_{l^1} = 1, c_i \geqslant 0, \forall i$$

where $C = [c_1, c_2, \ldots, c_N]$ is the set of codes for X

Coding

- (2/3) **Sparse coding (SC)**
  - Allow multiple entities in codebook to represent one descriptor
    - Reduce quantization errors
  - Relax the L0 norm to L1 norm
    - May impose positivity
    - May impose sum-to-1 constraint

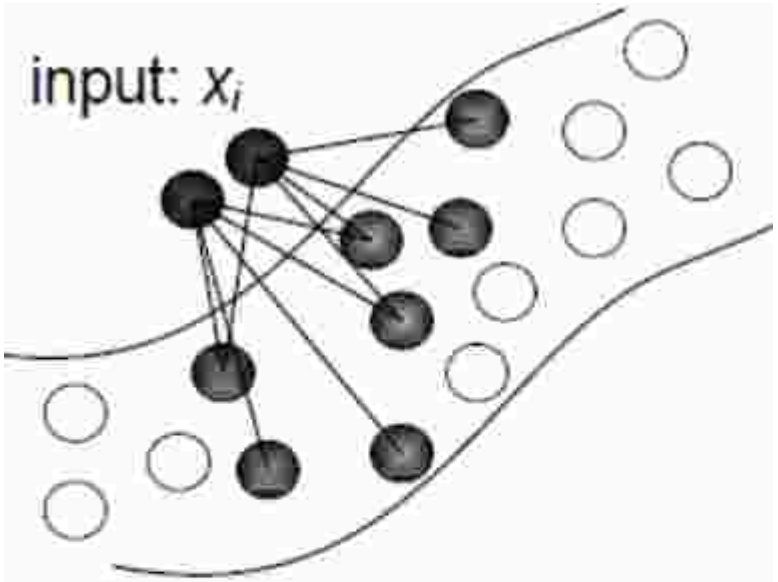


$$\min_{C} \sum_{i=1}^{N} \|x_i - B c_i\|^2 + \lambda \|c_i\|_{l^1}$$

- Can lead to non-local coding

Coding

- (3/3) **LcLC**
  - To code a descriptor, its local codebook vectors more important
  - Penalize far-off codebook vectors exp. more (instead of fixing K)



LLC

$$\min_C \sum_{i=1}^{N} \|x_i - Bc_i\|^2 + \lambda \|d_i \odot c_i\|^2$$

$$s.t. 1^T c_i = 1, \forall i$$

$\odot$: the element-wise multiplication

$d_i \in R^M$

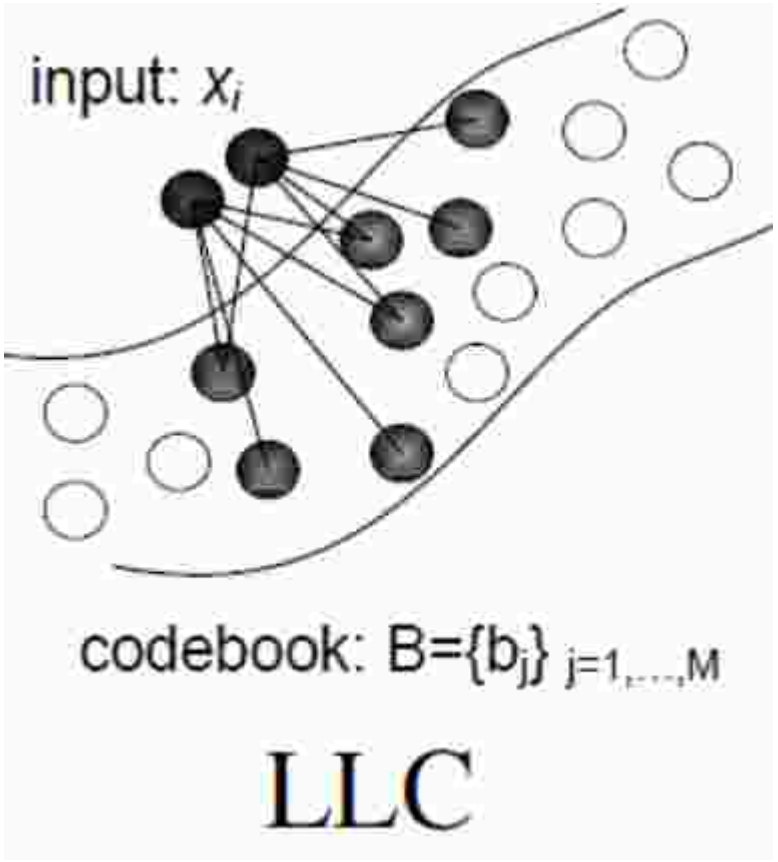
$$d_i = \exp\left(\frac{dist(x_i, B)}{\sigma}\right)$$

where $dist(x_i, B) = [dist(x_i, b_1), dist(x_i, b_2), \ldots, dist(x_i, b_M)]^T$

Coding

- (3/3) **Approximated LcLC**
  - To code a descriptor, its local codebook vectors more important
  - If you fix K (instead of using exponential penalty), then you have a small system to solve for each coding process
    - Typical K 5–10

$$\min_{\tilde{C}} \sum_{i=1}^{N} \|x_i - \tilde{c}_i B_i\|^2$$
$$s.t.\ 1^T \tilde{c}_i = 1, \forall i$$



Coding

- Approx-LcLC performance with varying K (Caltech-101)

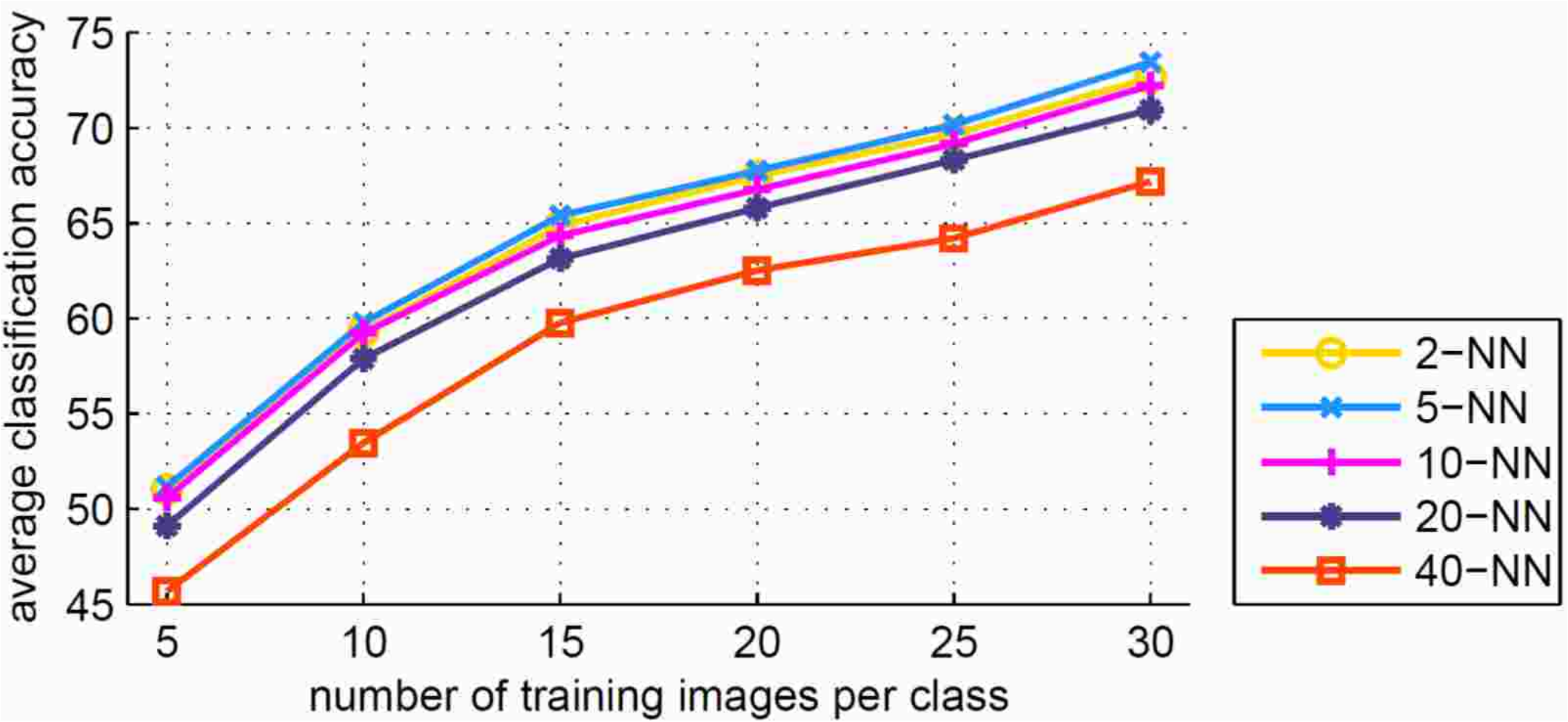
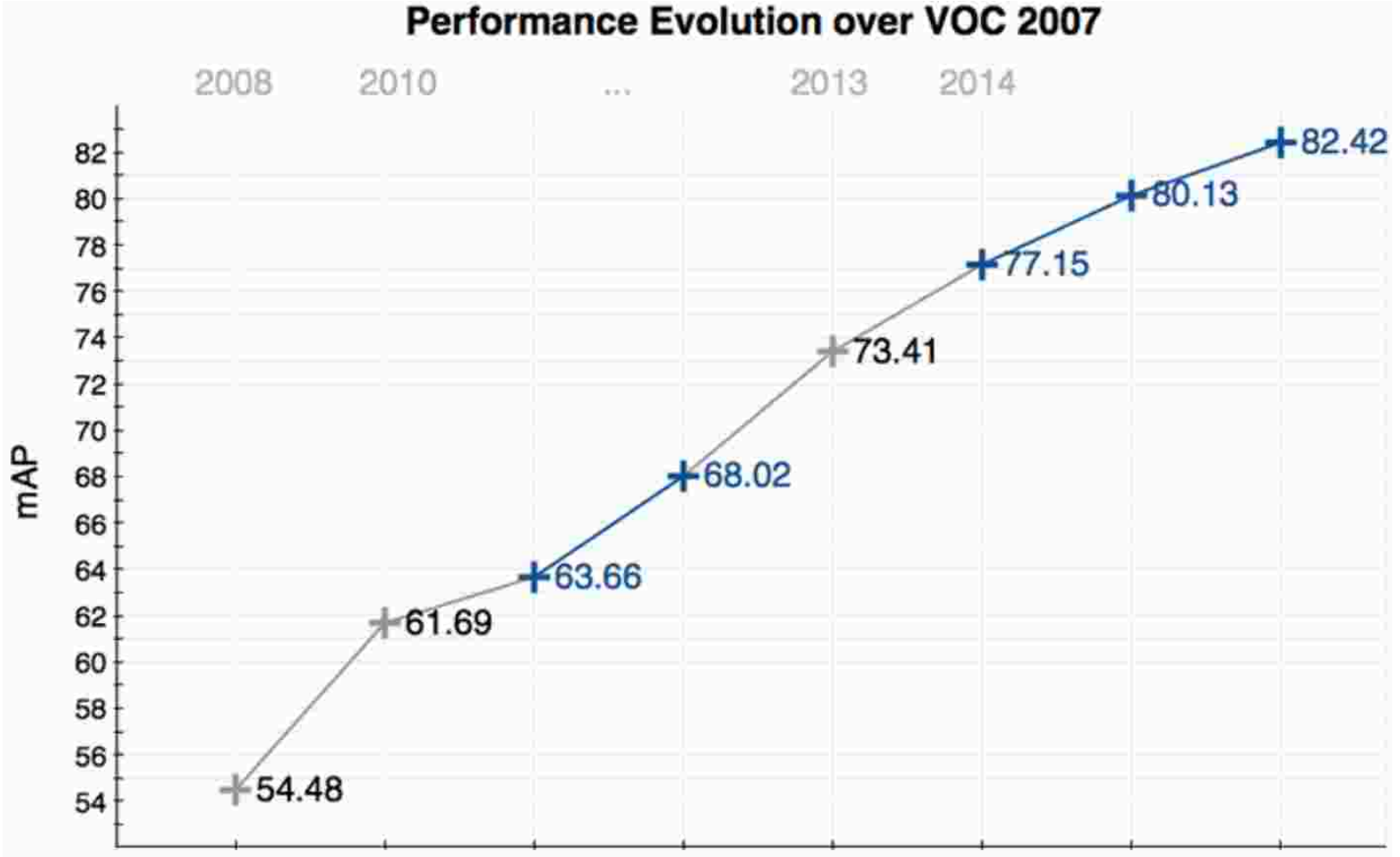


Image Classification

- VOC 2007 data [host.robots.ox.ac.uk/pascal/VOC/voc2007]
  - mAP= mean (over classes) average (over queries) precision



Method	BOW	FK-BL	FK	FK-IN	DeCAF	CNN-F	CNN-M 2K	CNN-S
Dim.	32K	327K	327K	84K	4K	4K	2K	4K (TN)
Aug.	–	–	–	f s	t t	f s	f s	f s
Ref.		[I]	[a]	[i]	[II]	[m]	[y]	[γ]