C280, Computer Vision

Prof. Trevor Darrell
trevor@eecs.berkeley.edu

Lecture 14: Discriminative Kernels for Recognition

Two Lectures ago…

- Scanning window paradigm
- GIST
- HOG
- Boosted Face Detection
- Local-feature Alignment; from Roberts to Lowe...
- BOW Indexing

Last Lecture: Topic Models for Recognition

Guest lecture by Kate Saenko:

- Dataset issues
- Topic models for category discovery [Sivic05]
- Category discovery from web [Fergus05]
- Bootstrapping a category model [Li07]
- Using text in addition to image [Berg06]
- Learning objects from a dictionary [Saenko08]

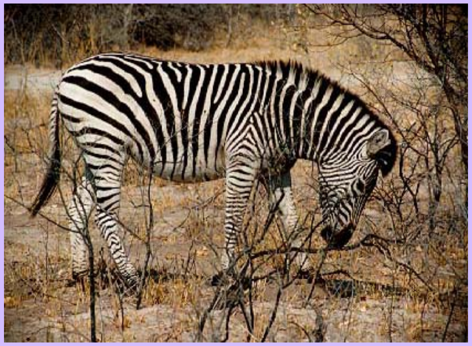
Last Lecture Summary

- The web contains unlimited, but extremely noisy object category data
- The text surrounding the image on the web page is an important recognition cue
- Topic models (pLSA, LDA, HDP, etc.) are useful for discovering objects in images and object senses in text
- Bootstrap model from small amount of labeled or weakly labeled data
- Still an open research problem!

Today: Discriminative Kernels

- SVM-BOW
- Pyramid and Spatial-Pyramid match
- Fast Intersection Kernels
- Latent-part SVM models

Object categorization: the statistical viewpoint



$$p(zebra \mid image)$$

vs.

$$p(no\ zebra \mid image)$$

- Bayes rule:

$$\underbrace{\frac{p(zebra \mid image)}{p(no\ zebra \mid image)}}_{\text{posterior ratio}} = \underbrace{\frac{p(image \mid zebra)}{p(image \mid no\ zebra)}}_{\text{likelihood ratio}} \cdot \underbrace{\frac{p(zebra)}{p(no\ zebra)}}_{\text{prior ratio}}$$

Fei-Fei, Fergus, Torralba, CVPR 2007 SC

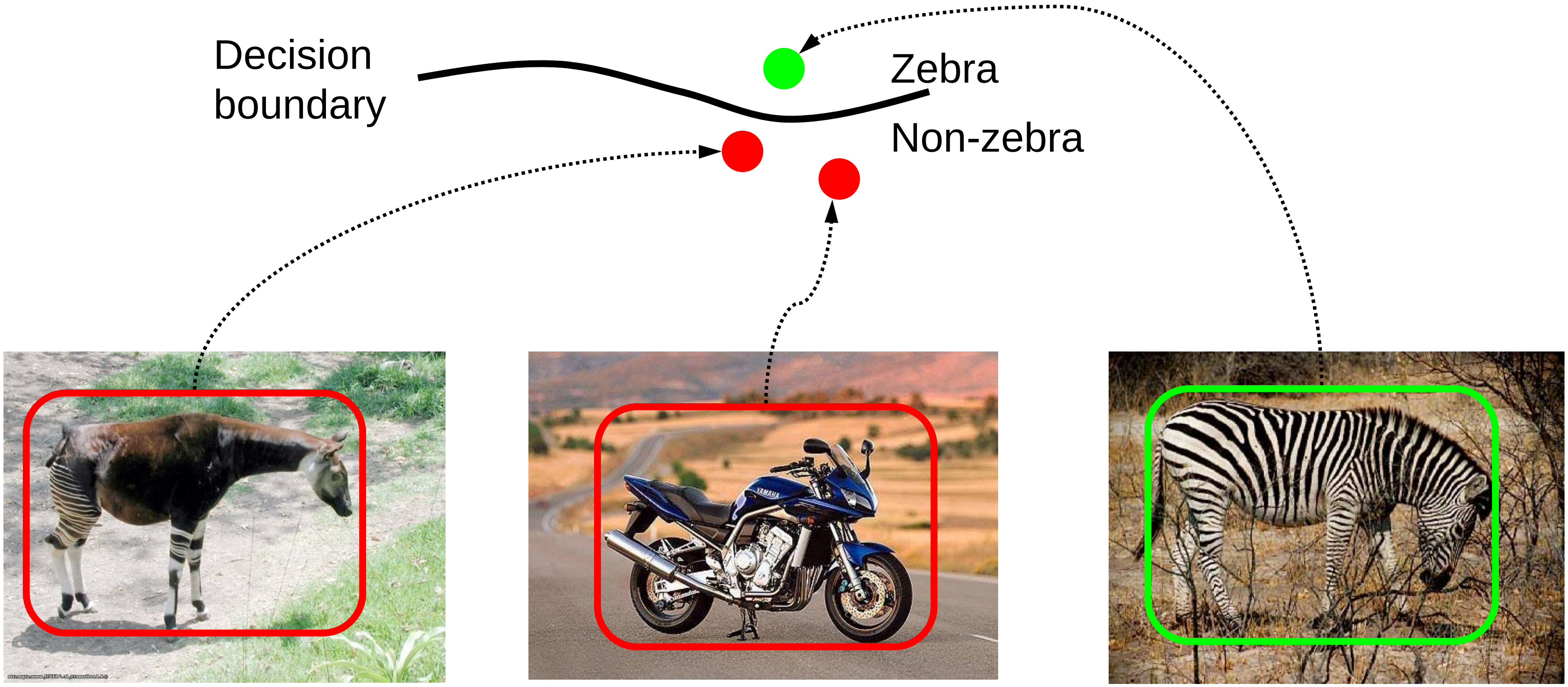
Object categorization: the statistical viewpoint

$$\underbrace{\frac{p(zebra \mid image)}{p(no\ zebra \mid image)}}_{\text{posterior ratio}} = \underbrace{\frac{p(image \mid zebra)}{p(image \mid no\ zebra)}}_{\text{likelihood ratio}} \cdot \underbrace{\frac{p(zebra)}{p(no\ zebra)}}_{\text{prior ratio}}$$

- **Discriminative methods model posterior**
- **Generative methods model likelihood and prior**

Discriminative

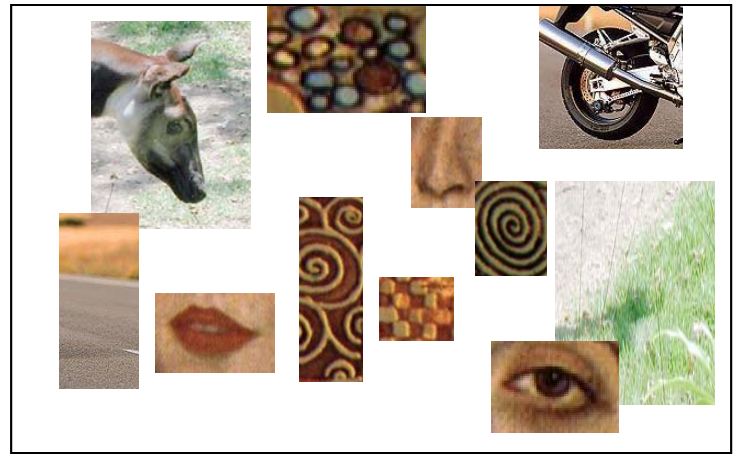
- Direct modeling of $$\frac{p(zebra \mid image)}{p(no\ zebra \mid image)}$$



Fei-Fei, Fergus, Torralba, CVPR 2007 SC

Generative

- Model $p(image \mid zebra)$ and $p(image \mid no\ zebra)$

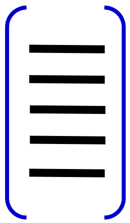


	$p(image \mid zebra)$	$p(image \mid no\ zebra)$
	Low	Middle
	High	Middle→Low

1.Feature detection and representation

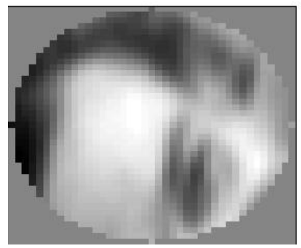
- Regular grid
  – Vogel & Schiele, 2003
  – Fei-Fei & Perona, 2005
- Interest point detector
  – Csurka, Bray, Dance & Fan, 2004
  – Fei-Fei & Perona, 2005
  – Sivic, Russell, Efros, Freeman & Zisserman, 2005
- Other methods
  – Random sampling (Vidal-Naquet & Ullman, 2002)
  – Segmentation based patches (Barnard, Duygulu, Forsyth, de Freitas, Blei, Jordan, 2003)

1.Feature detection and representation

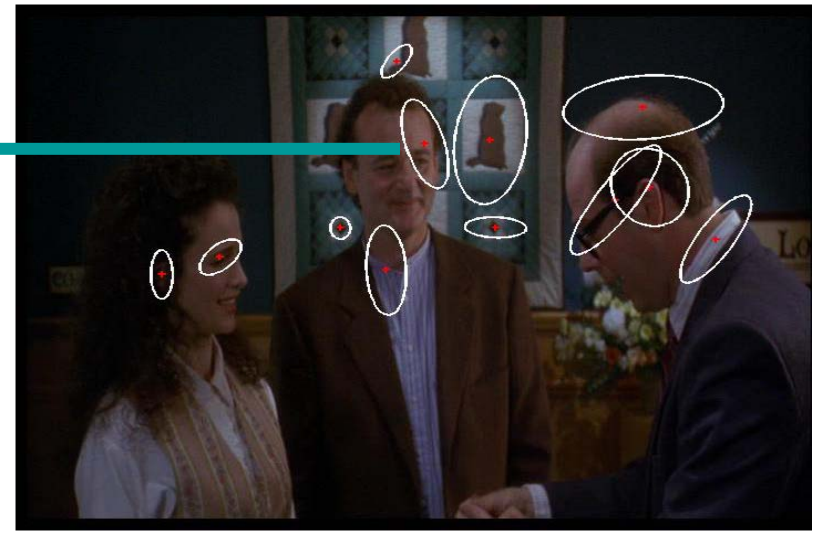


Compute SIFT descriptor

[Lowe’99]



Normalize patch



Detect patches

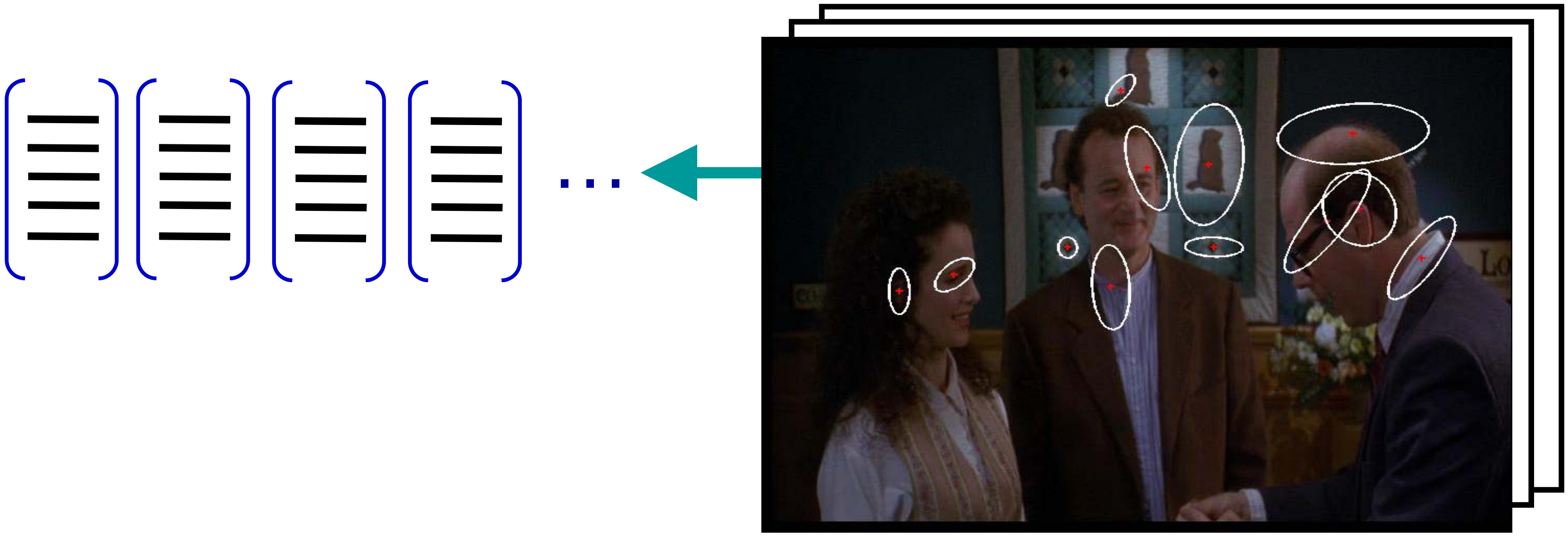
[Mikojaczyk and Schmid ’02]

[Mata, Chum, Urban & Pajdla, ’02]

[Sivic & Zisserman, ’03]

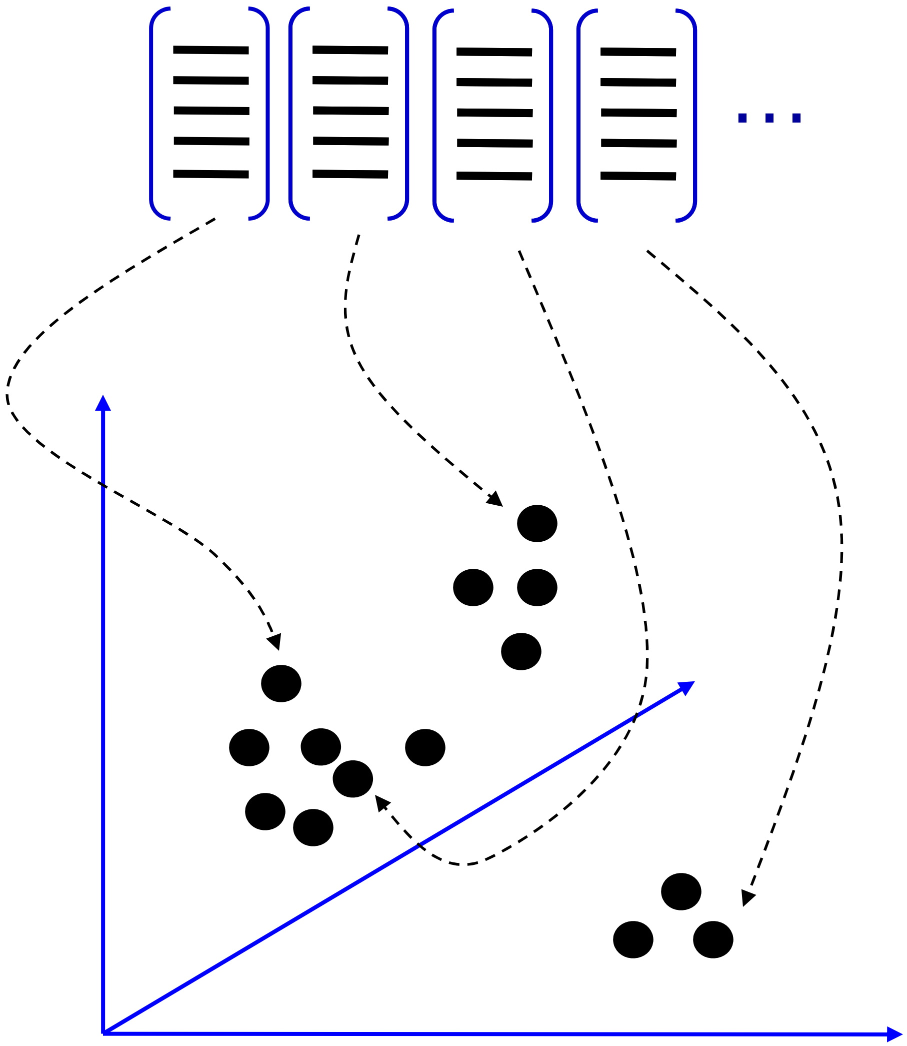
Slide credit: Josef Sivic

1.Feature detection and representation



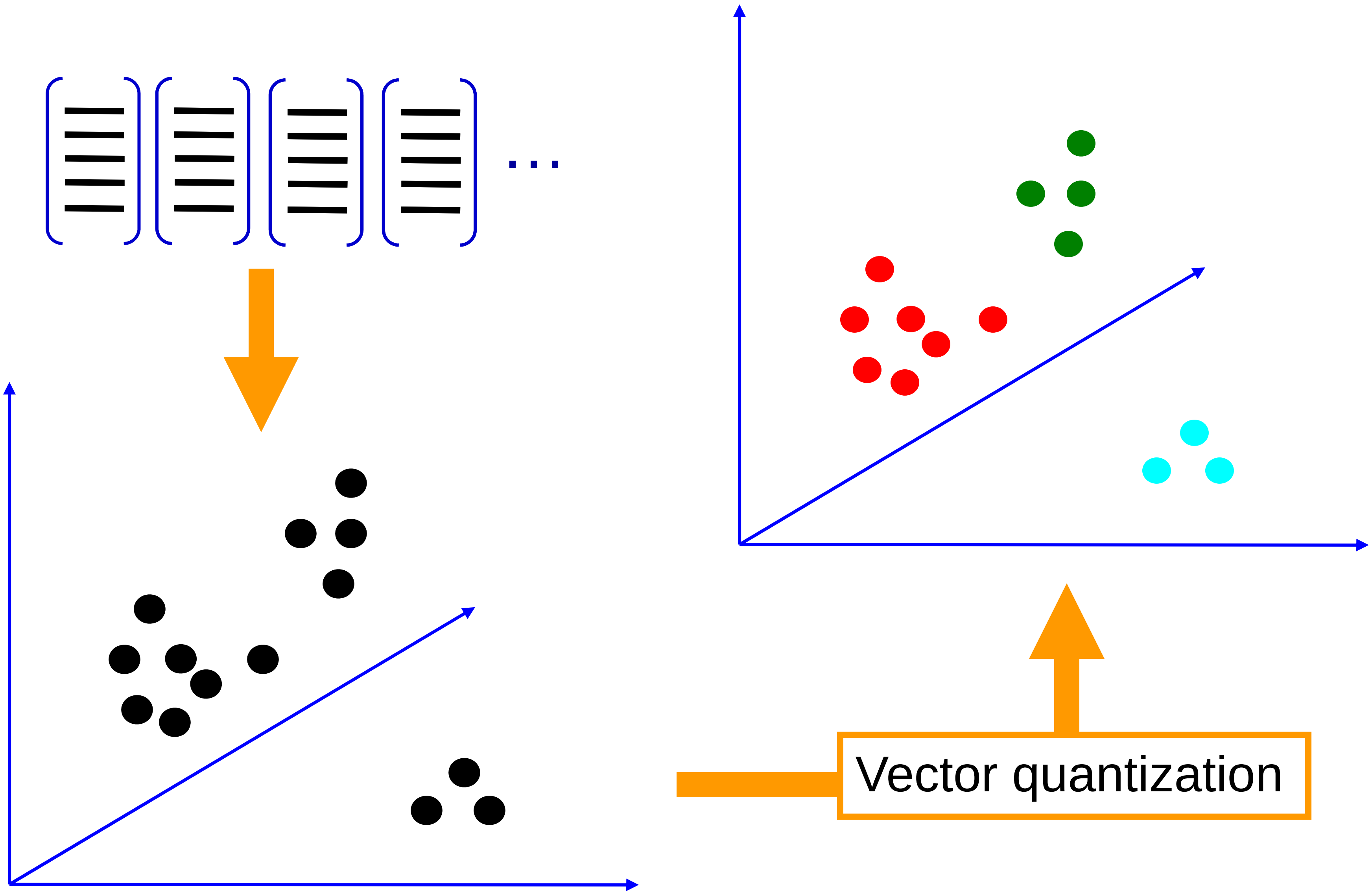
Fei-Fei, Fergus, Torralba, CVPR 2007 SC

2. Codewords dictionary formation



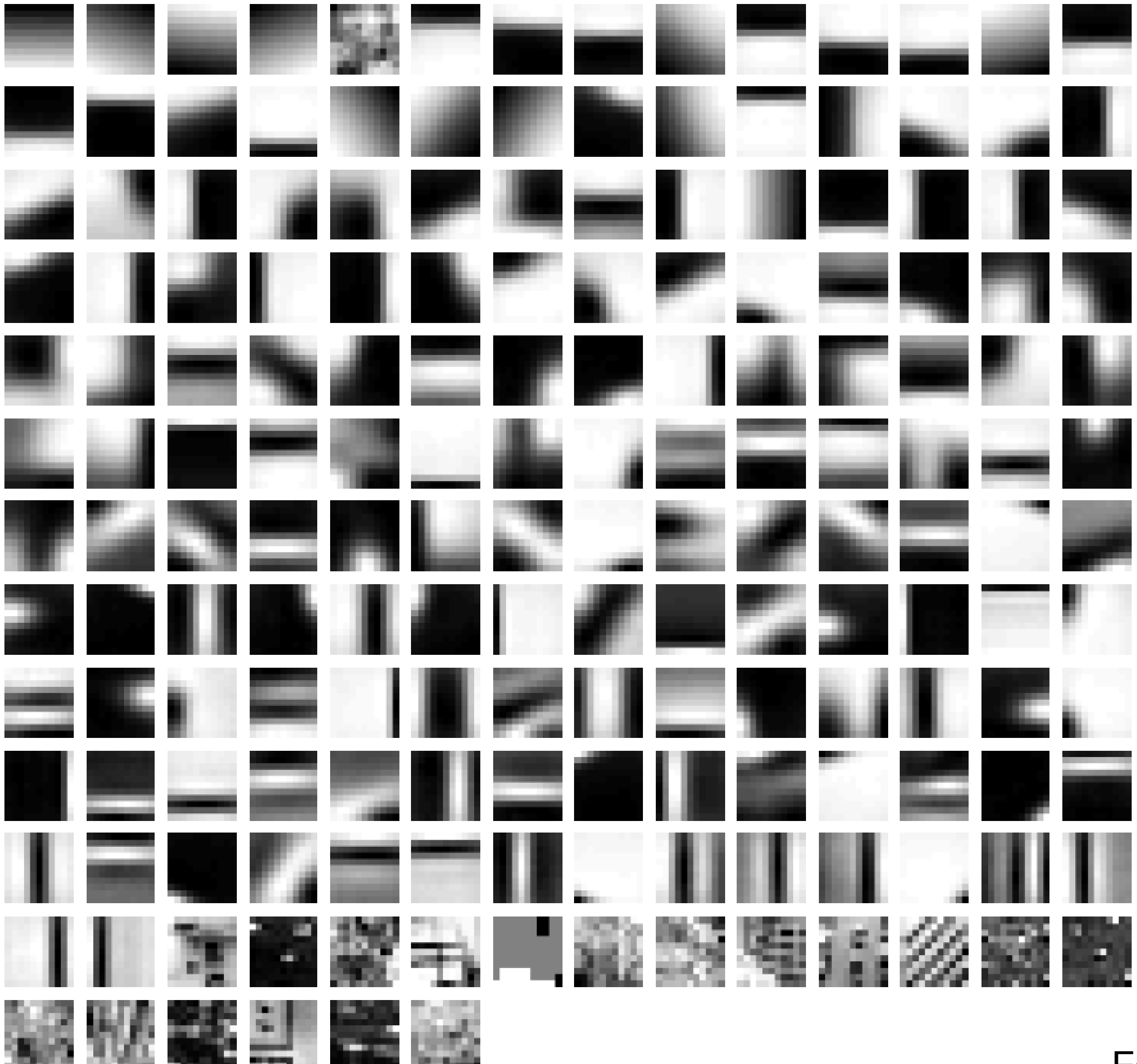
Fei-Fei, Fergus, Torralba, CVPR 2007 SC

2. Codewords dictionary formation



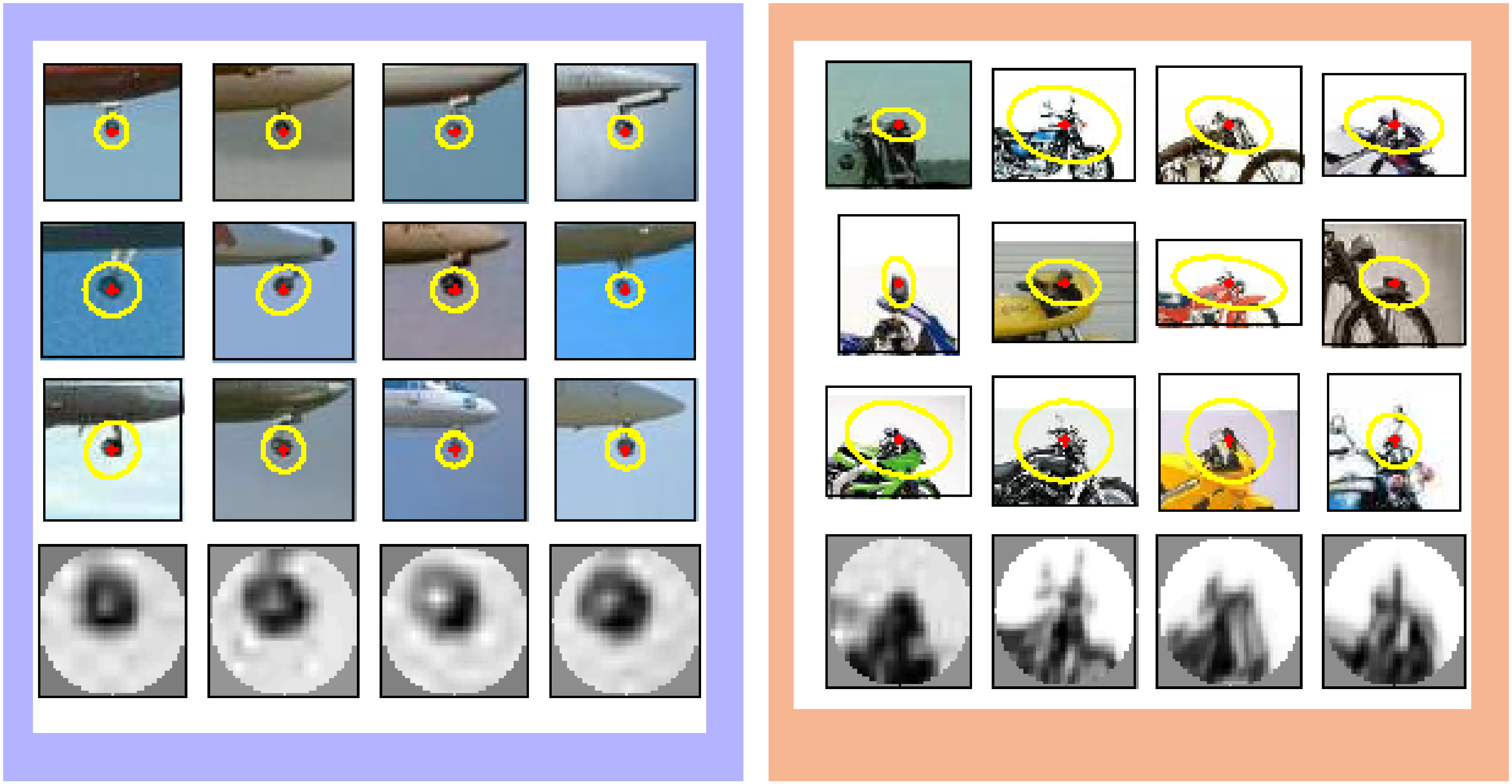
Slide credit: Josef Sivic

2. Codewords dictionary formation



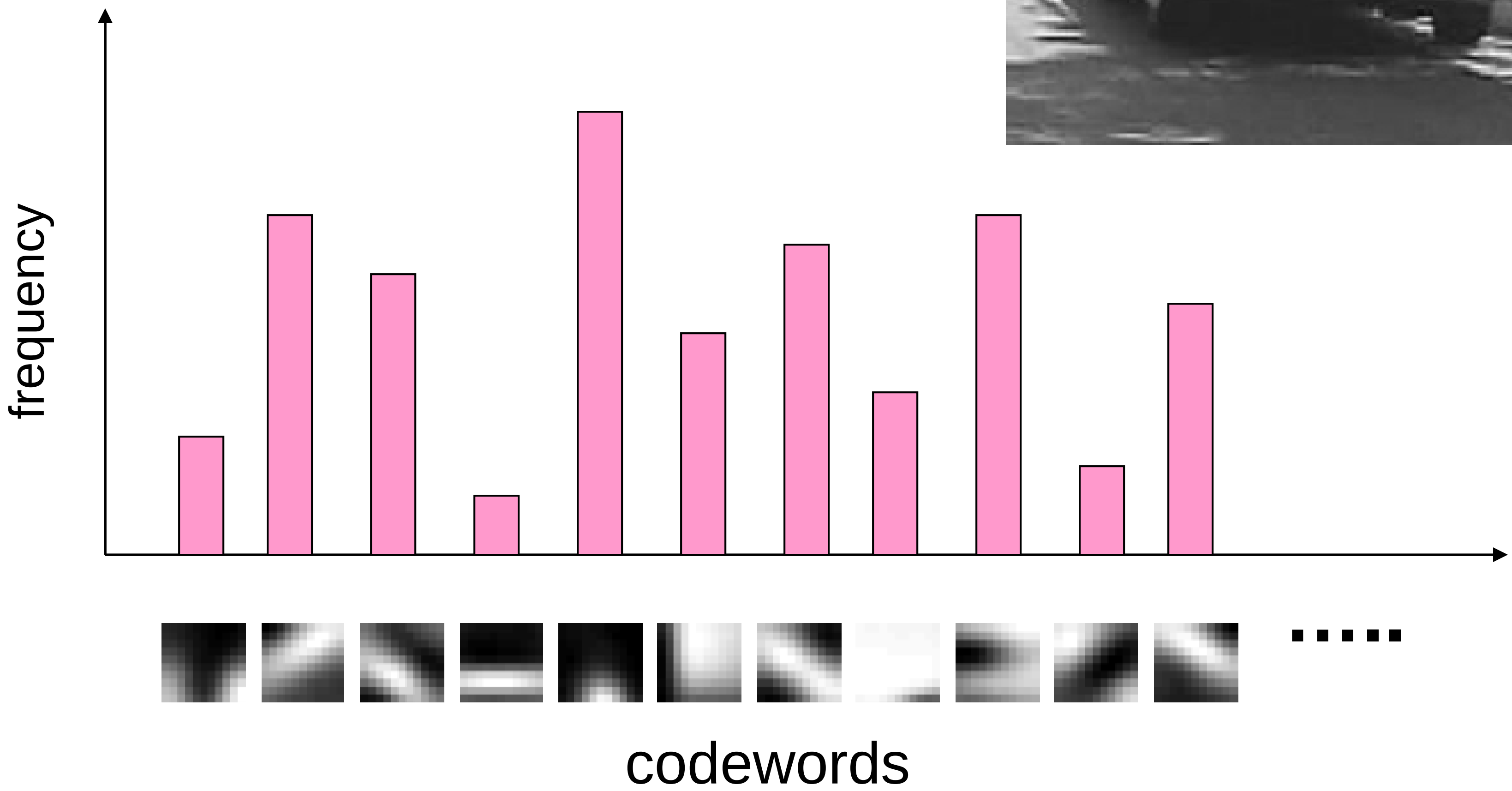
Fei-Fei et al. 2005

Image patch examples of codewords



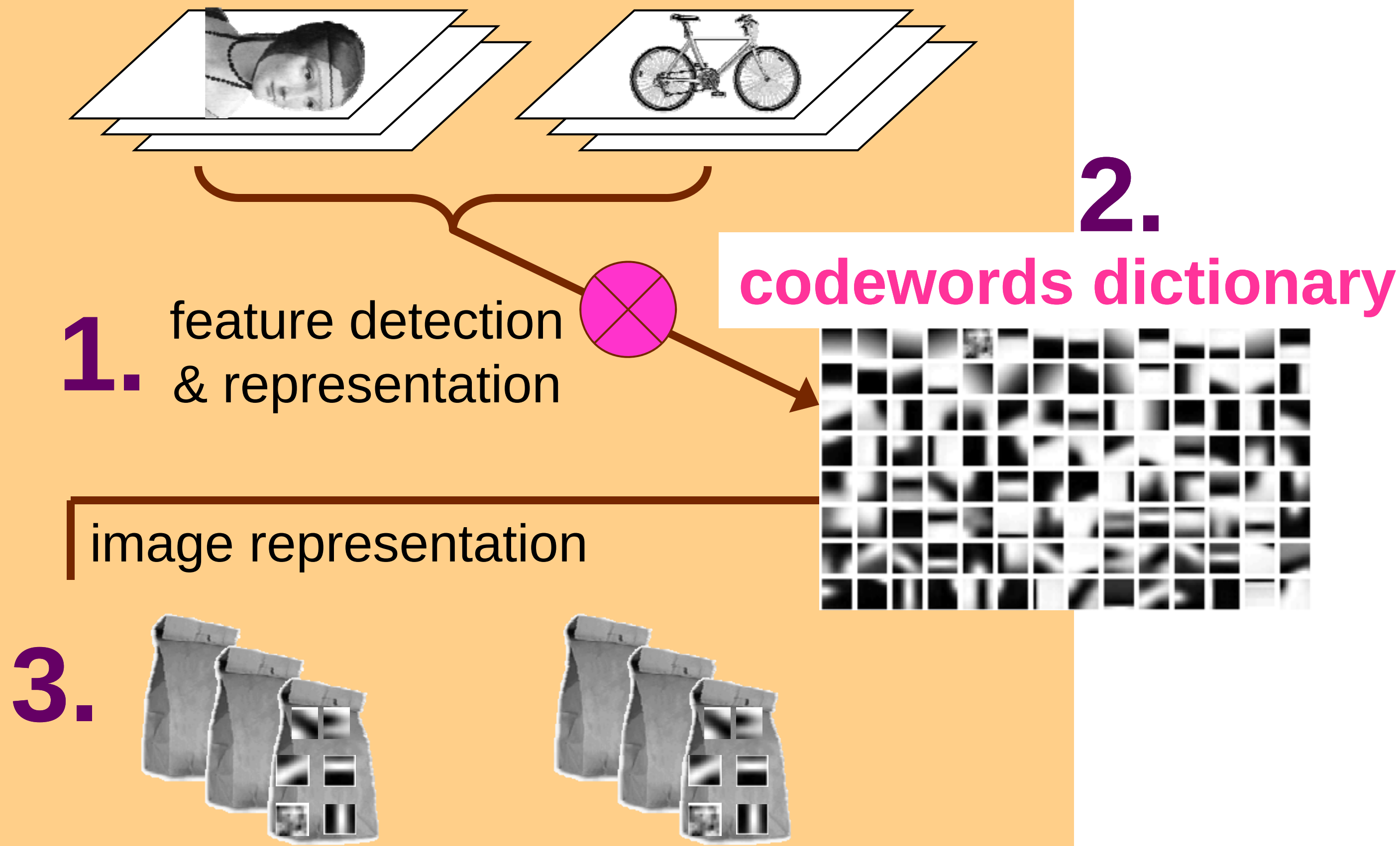
Sivic et al. 2005

3. Image representation



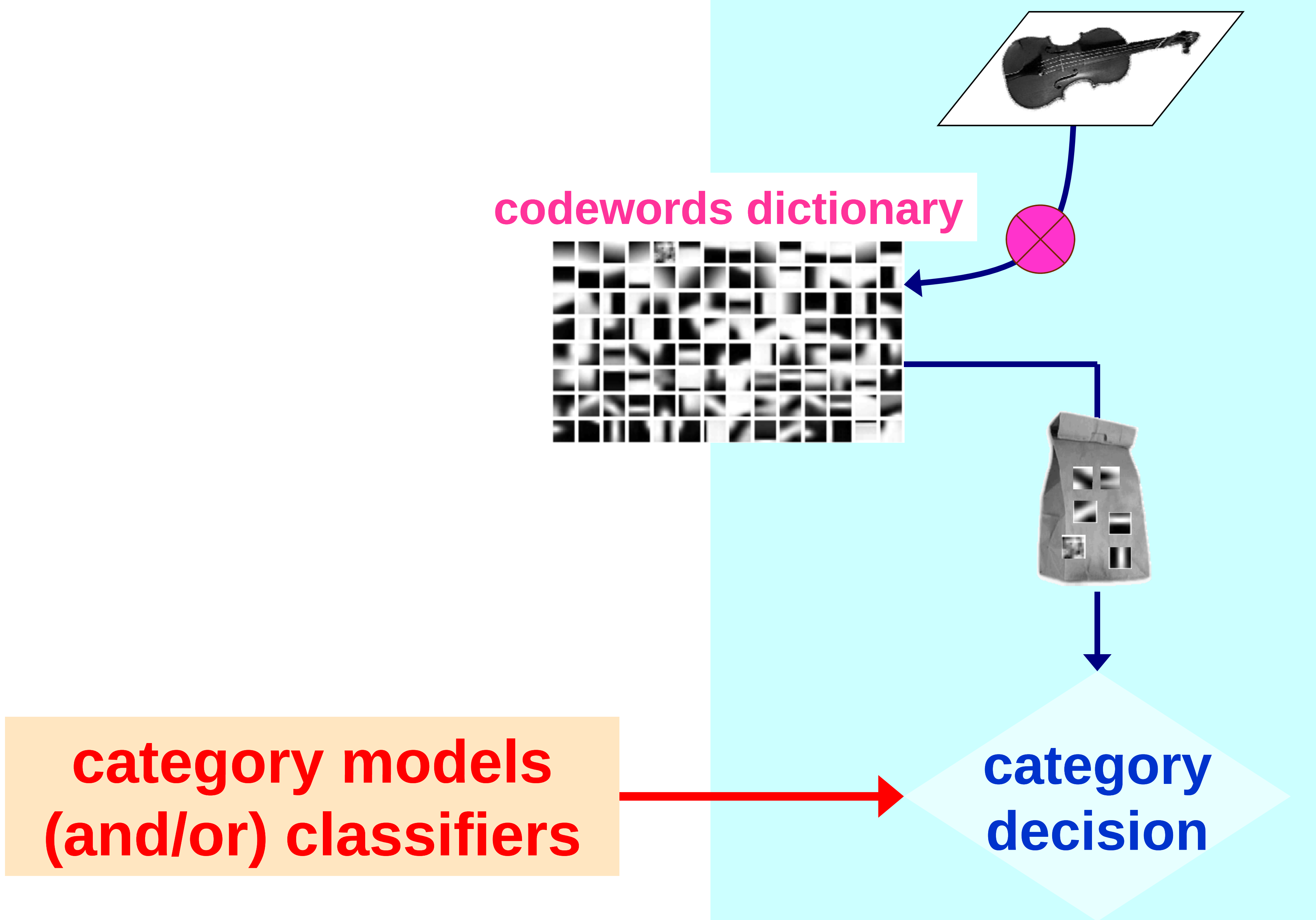
Fei-Fei, Fergus, Torralba, CVPR 2007 SC

Representation



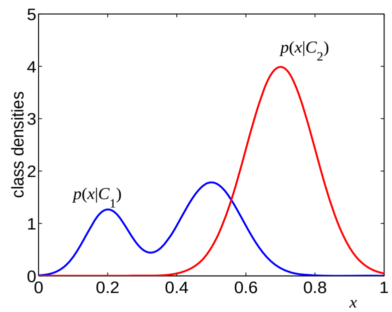
Fei-Fei, Fergus, Torralba, CVPR 2007 SC

Learning and Recognition

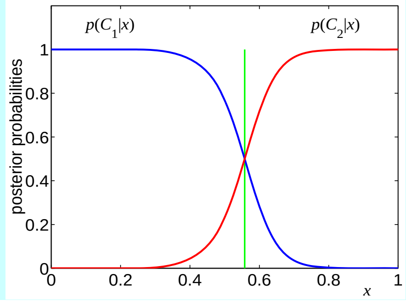


Learning and Recognition

1. Generative method:
- graphical models



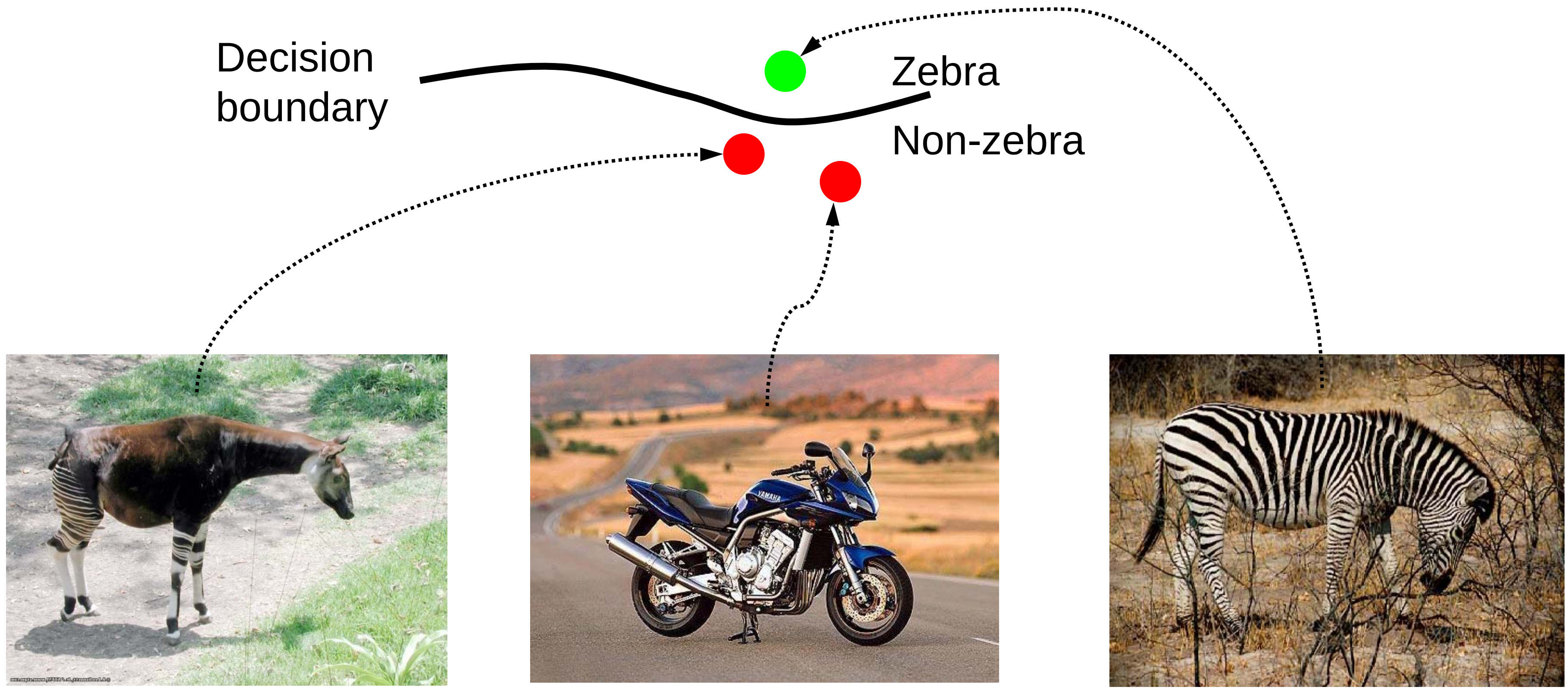
2. Discriminative method:
- SVM



category models
(and/or) classifiers

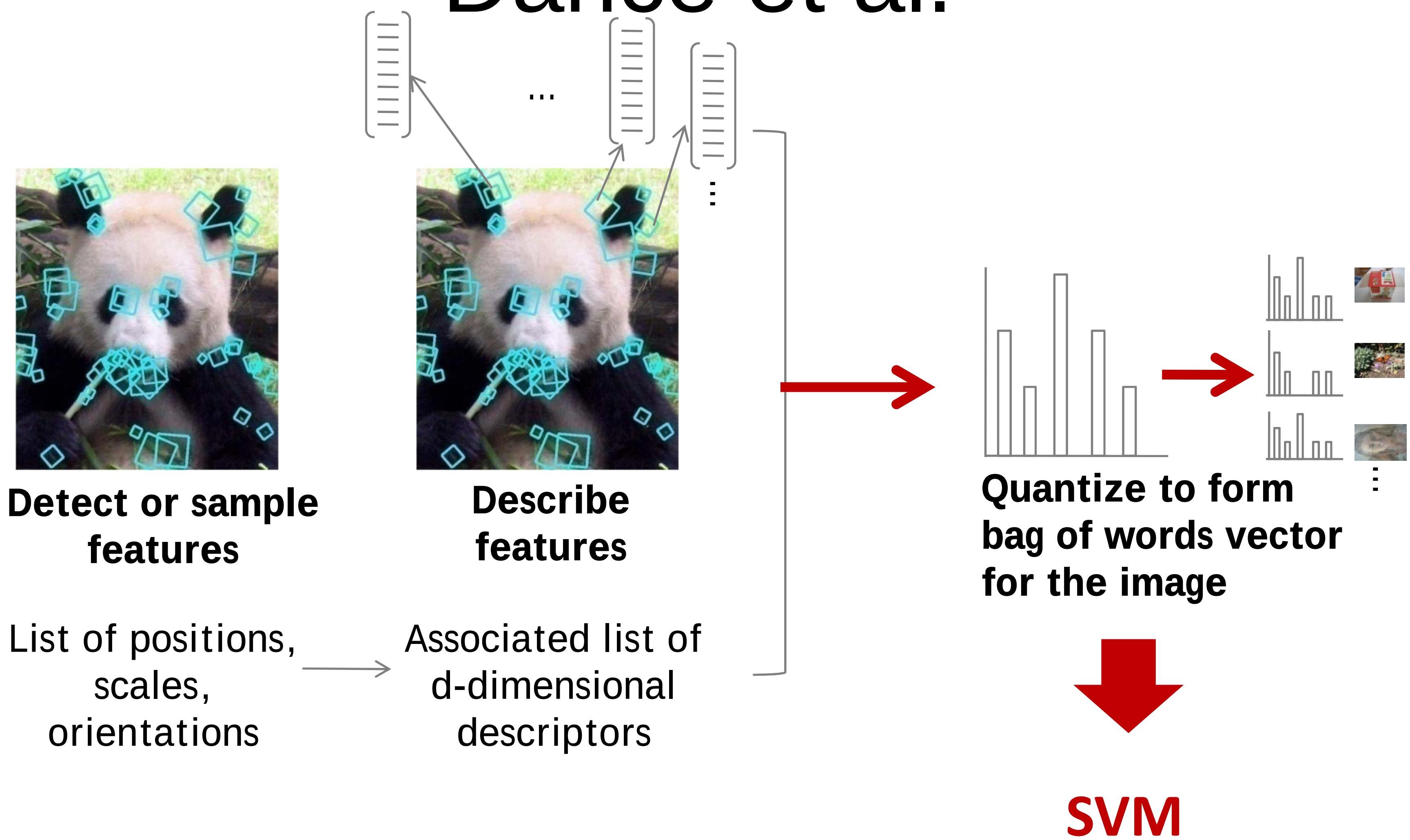
Fei-Fei, Fergus, Torralba, CVPR 2007 SC

Discriminative methods based on ‘bag of words’ representation



Fei-Fei, Fergus, Torralba, CVPR 2007 SC

Dance et al.


...
Detect or sample features
Describe features
List of positions, scales, orientations
Associated list of d-dimensional descriptors
Quantize to form bag of words vector for the image
SVM

Visual Categorization with Bags of Keypoints

Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, Cédric Bray

Xerox Research Centre Europe
6, chemin de Maupertuis
38240 Meylan, France
{gcsurka,cdance}@xrce.xerox.com

Abstract. We present a novel method for generic visual categorization: the problem of identifying the object content of natural images while generalizing across variations inherent to the object class. This *bag of keypoints* method is based on vector quantization of affine invariant descriptors of image patches. We propose and compare two alternative implementations using different classifiers: Naïve Bayes and SVM. The main advantages of the method are that it is simple, computationally efficient and intrinsically invariant. We present results for simultaneously classifying seven semantic visual categories. These results clearly demonstrate that the method is robust to background clutter and produces good categorization accuracy even without exploiting geometric information.

Chris Dance, Jutta Willamowski, Lixin Fan, Cedric Bray, and Gabriela Csurka. Visual categorization with bags of keypoints. In ECCV International Workshop on Statistical Learning in Computer Vision, 2004.

Embedding

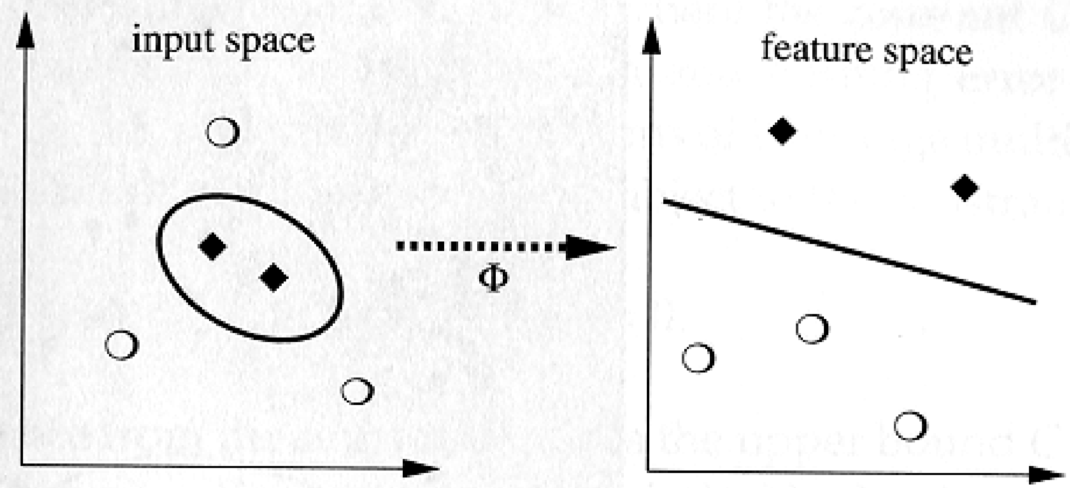


Figure 1.6 The idea of SVMs: map the training data into a higher-dimensional feature space via Φ, and construct a separating hyperplane with maximum margin there. This yields a nonlinear decision boundary in input space. By the use of a kernel function (1.2), it is possible to compute the separating hyperplane without explicitly carrying out the map into the feature space.

Learning with Kernels, Scholkopf and Smola, 2002

Kernels

- linear classifier:

$$\mathbf{f}(\mathbf{x}) = \mathbf{sign}(\mathbf{w}^{\mathrm{T}}\mathbf{x} + \mathbf{b})$$

- Kernel classifier:

$$\mathbf{K}(\mathbf{u}, \mathbf{v}) = \Phi(\mathbf{u}) \cdot \Phi(\mathbf{v})$$

$$\mathbf{f}(\mathbf{x}) = \mathbf{sign}(\sum_{\mathbf{i}} \mathbf{y}_{\mathbf{i}} \alpha_{\mathbf{i}} \mathbf{K}(\mathbf{x}, \mathbf{x}_{\mathbf{i}}) + \mathbf{b}).$$

[Dance et al.]

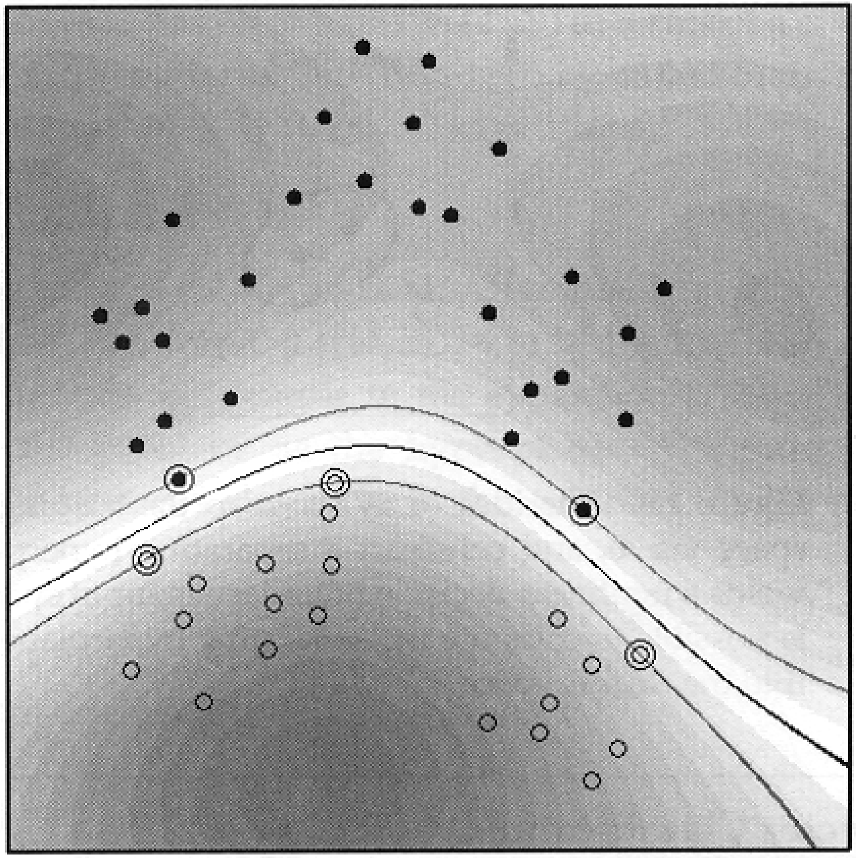


Figure 1.7 Example of an SV classifier found using a radial basis function kernel $k(x, x') = \exp(-\|x - x'\|^2)$ (here, the input space is $\mathcal{X} = [-1, 1]^2$). Circles and disks are two classes of training examples; the middle line is the decision surface; the outer lines precisely meet the constraint (1.25). Note that the SVs found by the algorithm (marked by extra circles) are not centers of clusters, but examples which are critical for the given classification task. Gray values code $|\sum_{i=1}^{m} y_i \alpha_i k(x, x_i) + b|$, the modulus of the argument of the decision function (1.35). The top and the bottom lines indicate places where it takes the value 1 (from [471]).

Learning with Kernels, Scholkopf and Smola, 2002

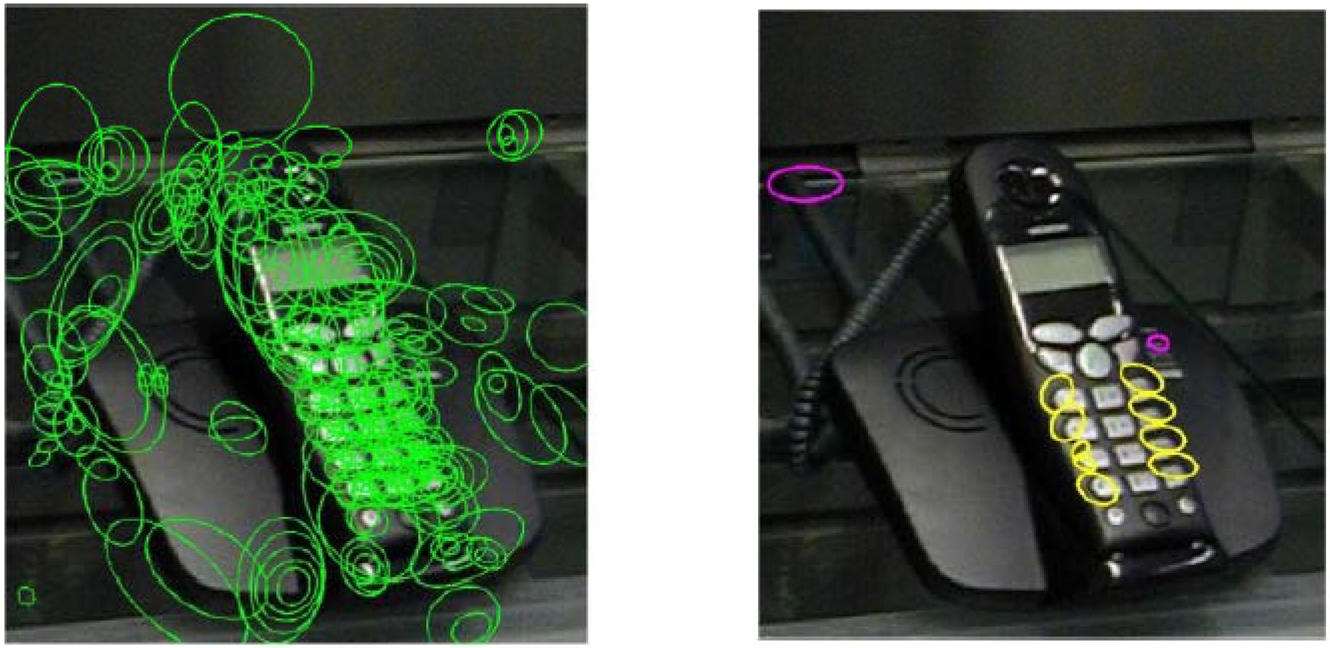
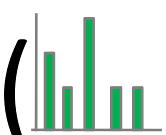
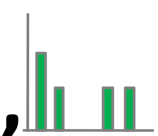
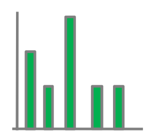
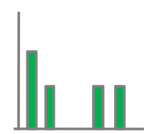


Fig. 4. *Left* all patches detected for this image. *Right* patches from two selected clusters occurring in this image (yellow and magenta ellipses).

Tried linear, quadratic, cubic; linear had best performance....

K( , ) = K( , ) = <  ,  >

[Dance et al.]

Fig. 5. Images correctly classified containing multiple objects of the same category.

Table 2. Confusion matrix and mean rank for SVM (*k=1000, linear kernel).*

SVM:

True classes →	*faces*	*buildings*	*trees*	*cars*	*phones*	*bikes*	*books*
faces	**98**	14	10	10	34	0	13
buildings	1	**63**	3	0	3	1	6
trees	1	10	**81**	1	0	6	0
cars	0	1	1	**85**	5	0	5
phones	0	5	4	3	**55**	2	3
bikes	0	4	1	0	1	**91**	0
books	0	3	0	1	2	0	**73**
Mean ranks	1.04	1.77	1.28	1.30	1.83	1.09	1.39

Table 1. Confusion matrix and the mean rank for the best vocabulary (*k=1000).*

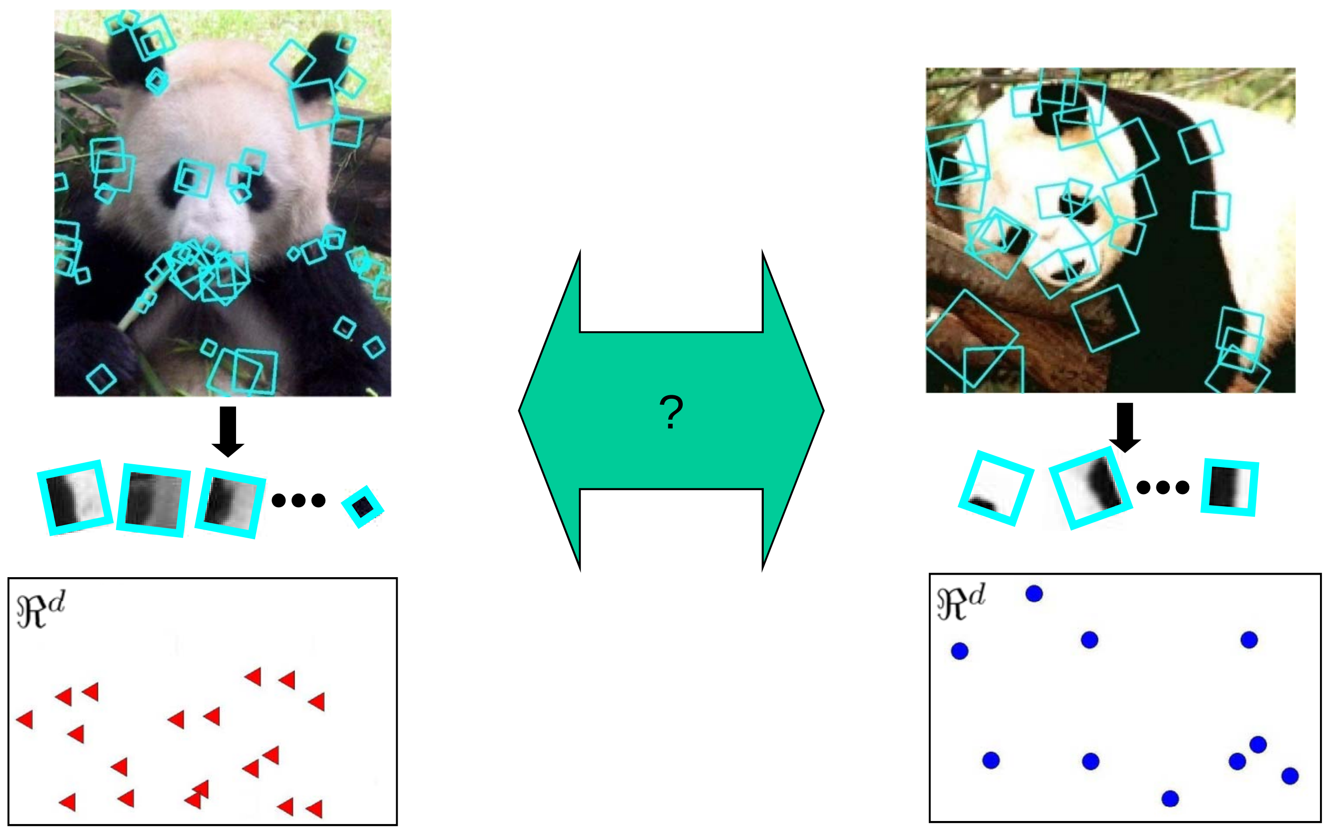
Naïve Bayes:

True classes →	*faces*	*buildings*	*trees*	*cars*	*phones*	*bikes*	*books*
faces	**76**	4	2	3	4	4	13
buildings	2	**44**	5	0	5	1	3
trees	3	2	**80**	0	0	5	0
cars	4	1	0	**75**	3	1	4
phones	9	15	1	16	**70**	14	11
bikes	2	15	12	0	8	**73**	0
books	4	19	0	6	7	2	**69**
Mean ranks	1.49	1.88	1.33	1.33	1.63	1.57	1.57

[Dance et al.]

How to Compare Sets of Features?

- Each instance is unordered set of vectors
- Varying number of vectors per instance

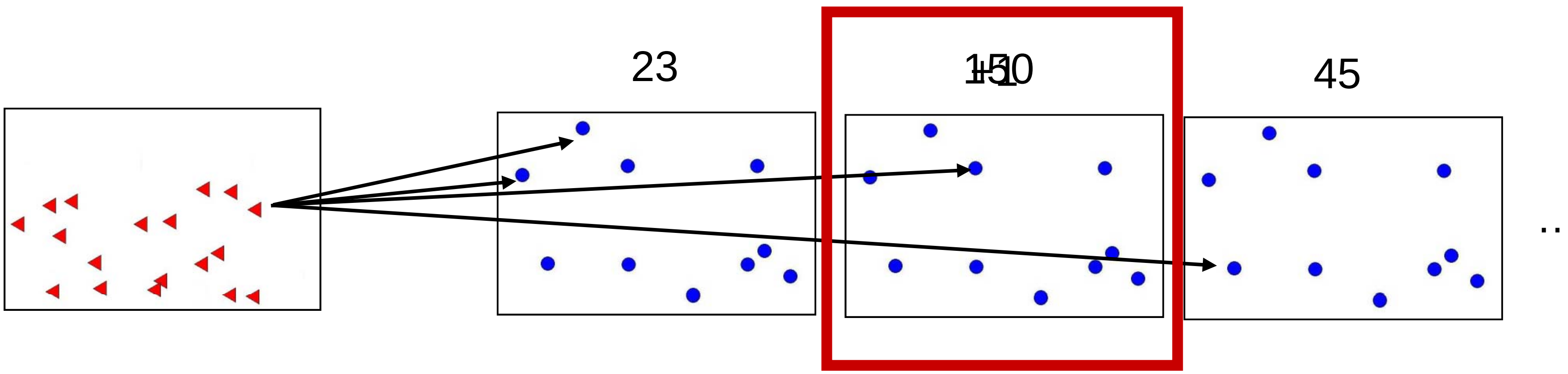


$$\mathbf{X} = \{\vec{\mathbf{x}}_1, \ldots, \vec{\mathbf{x}}_m\}$$

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

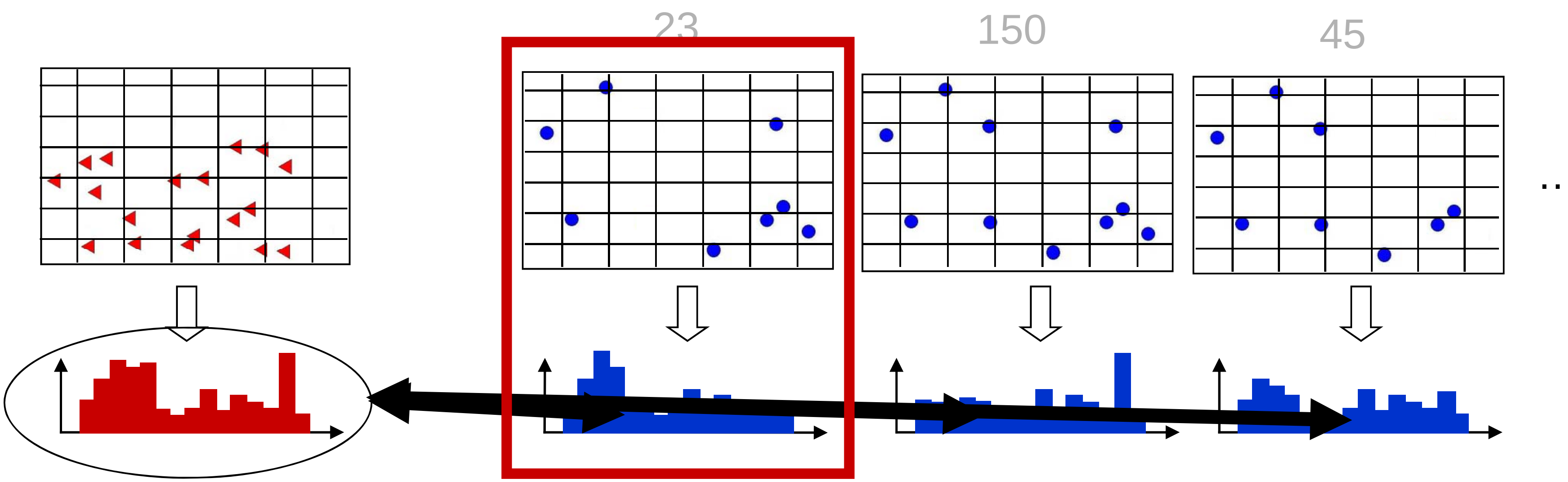
Recap: Conventional Approaches

“Voting” – for each patch, find the most similar patch in database, and vote for the image containing that patch. *[Schmid, Lowe, Tuytelaars et al.]*



Recap: Conventional Approaches

"Voting" – for each patch, find the most similar patch in database, and vote for the image containing that patch. *[Schmid, Lowe, Tuytelaars et al.]*



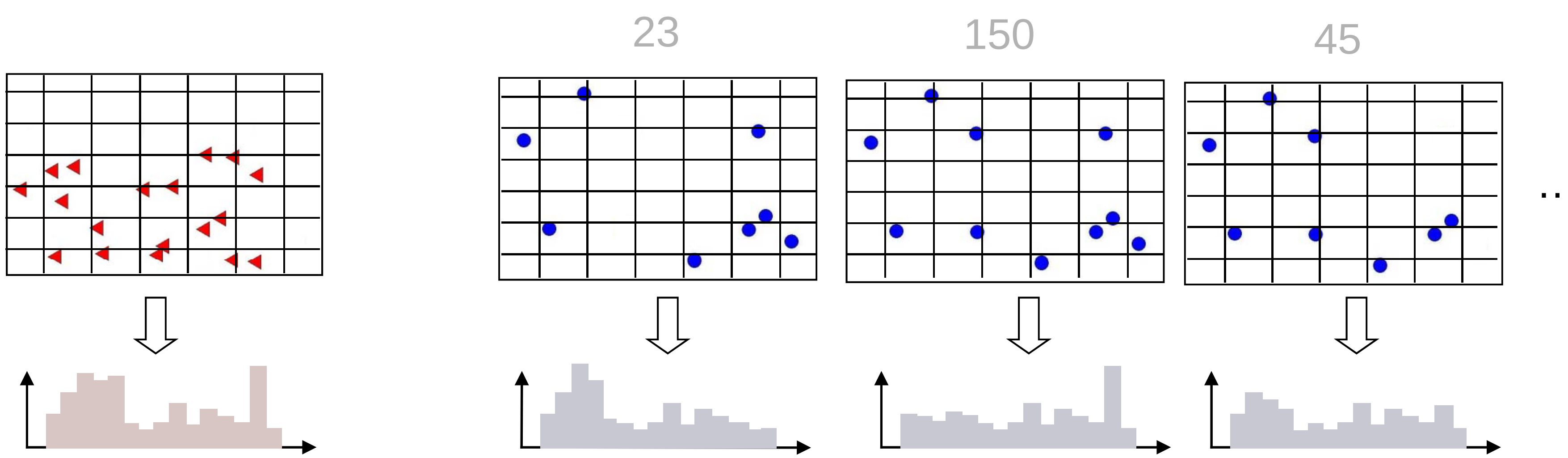
"Bag of Words" – quantize descriptor space, represent each image as a histogram over prototypes; use L1 indexing or SVM recognition. *[Csurka et al., Sivic & Zisserman, Lazebnik & Ponce, Agarwal & Triggs]*

Recap: Conventional Approaches

"Voting" – for each pa
database, and vote
patch.

Ignores co-occurance; can be costly; works well for instance matching

[Schmid, Lowe, Tuytelaars et al.]

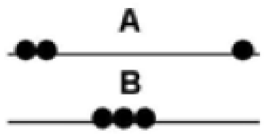


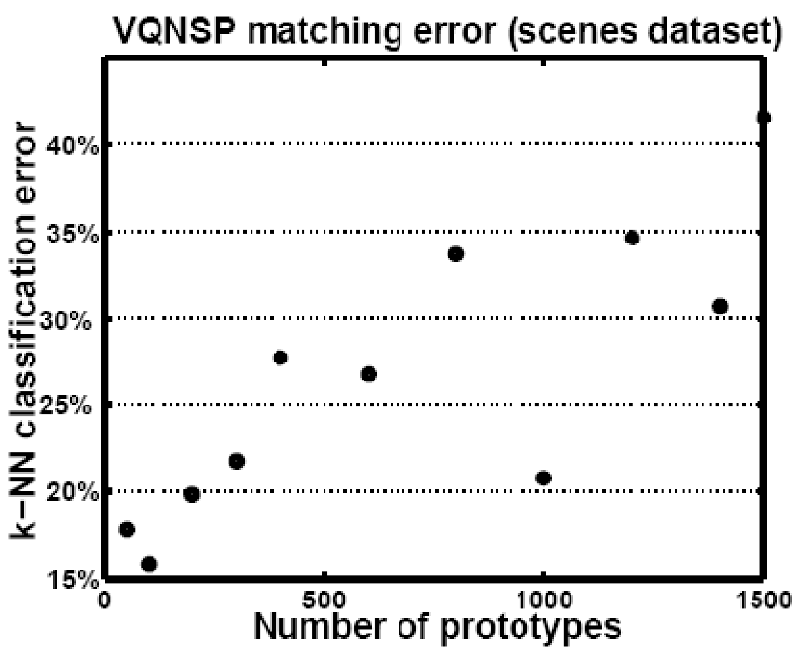
"Bag of Words" – qua
each image as a his
indexing or SVM recognition.

Sensitive to choice of quantization: how many "visual words"?

[Csurka et al., Sivic & Zisserman, Lazebnik & Ponce, Agarwal & Triggs]

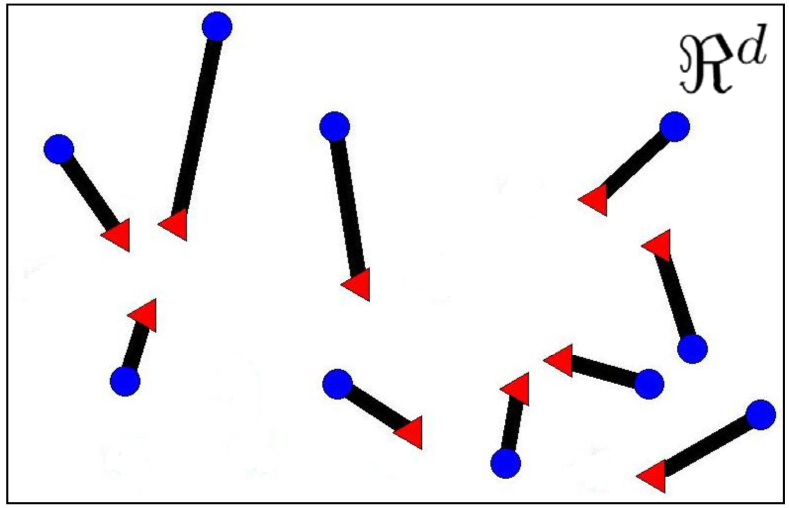
How Many Visual Words?





[Grauman and Darrell, CVPR 2005]

Correspondence-Based Match



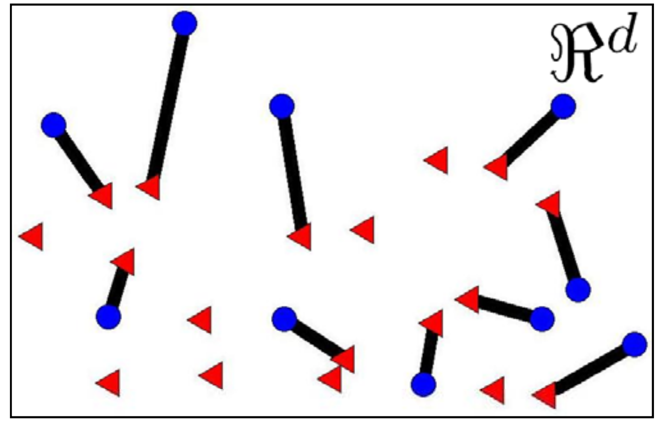
Explicit search for correspondences…

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

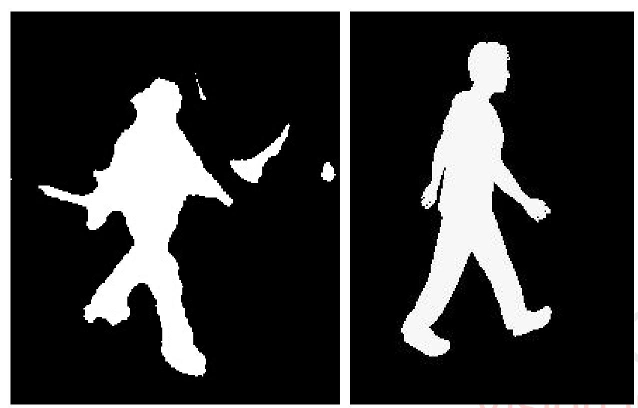
[Wallraven et al., Lyu, Boughhorbel et al., Belongie et al., Rubner et al., Berg et al., Gold & Rangarajan, Shashua & Hazan,…]

Partial Matching

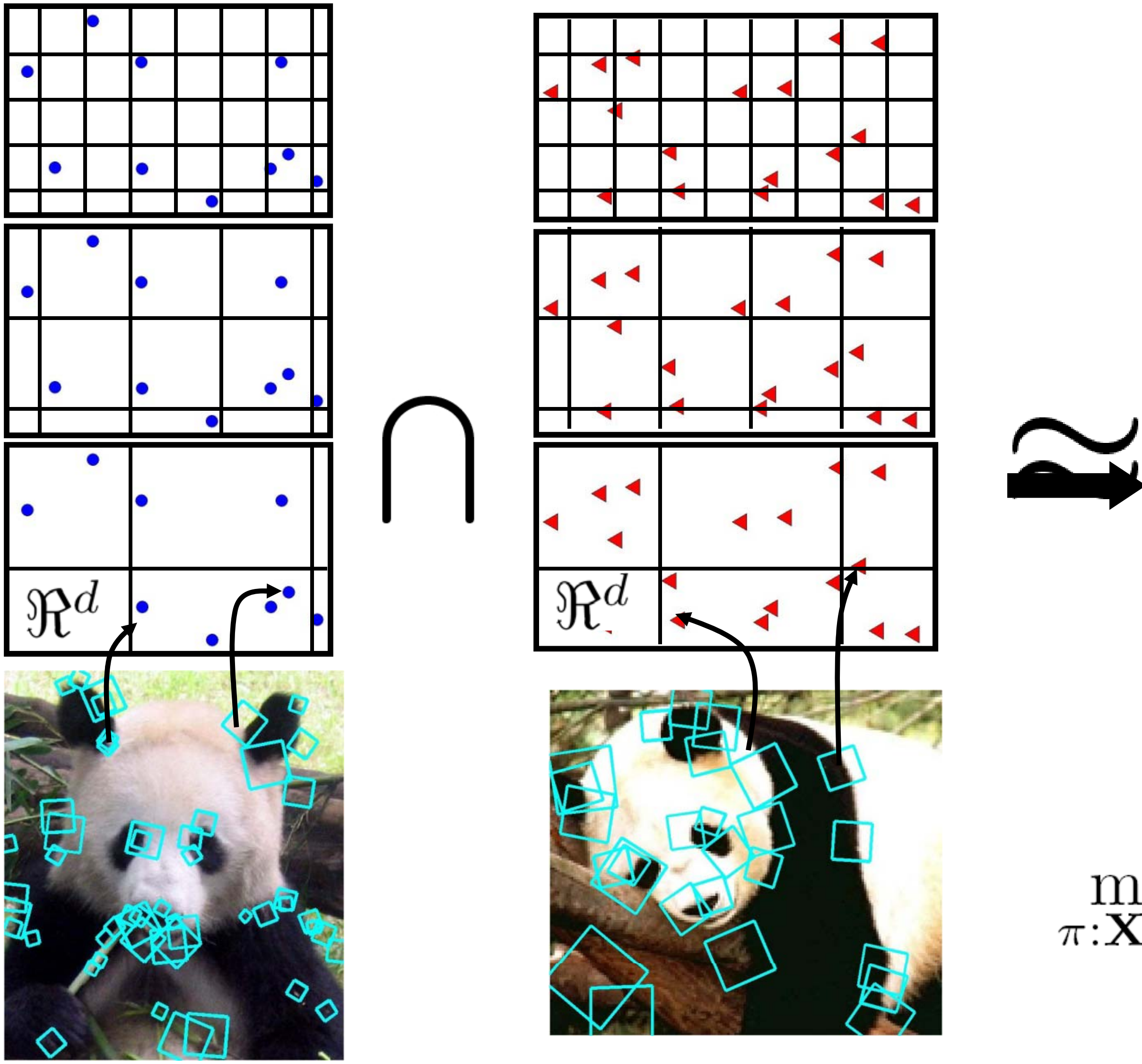
Compare sets by computing a *partial matching* between their features.

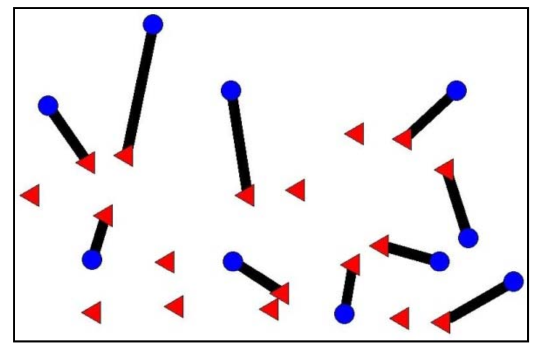


$$\max_{\pi:\mathbf{X}\rightarrow\mathbf{Y}} \sum_{\mathbf{x}_i\in\mathbf{X}} \mathcal{S}(\mathbf{x}_i, \pi(\mathbf{x}_i))$$



Pyramid Match





optimal partial matching

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

$$\mathbf{X} = \{\vec{\mathbf{x}}_1, \ldots, \vec{\mathbf{x}}_m\} \qquad \mathbf{Y} = \{\vec{\mathbf{y}}_1, \ldots, \vec{\mathbf{y}}_n\}$$

Computing the Partial Matching

- Optimal matching $O(dm^3)$
- Greedy matching $O(dm^2 \log m)$
- Pyramid match $O(dmL)$

for sets with $O(m)$ features of dimension d

[Grauman and Darrell, ICCV 2005, JMLR 2007]

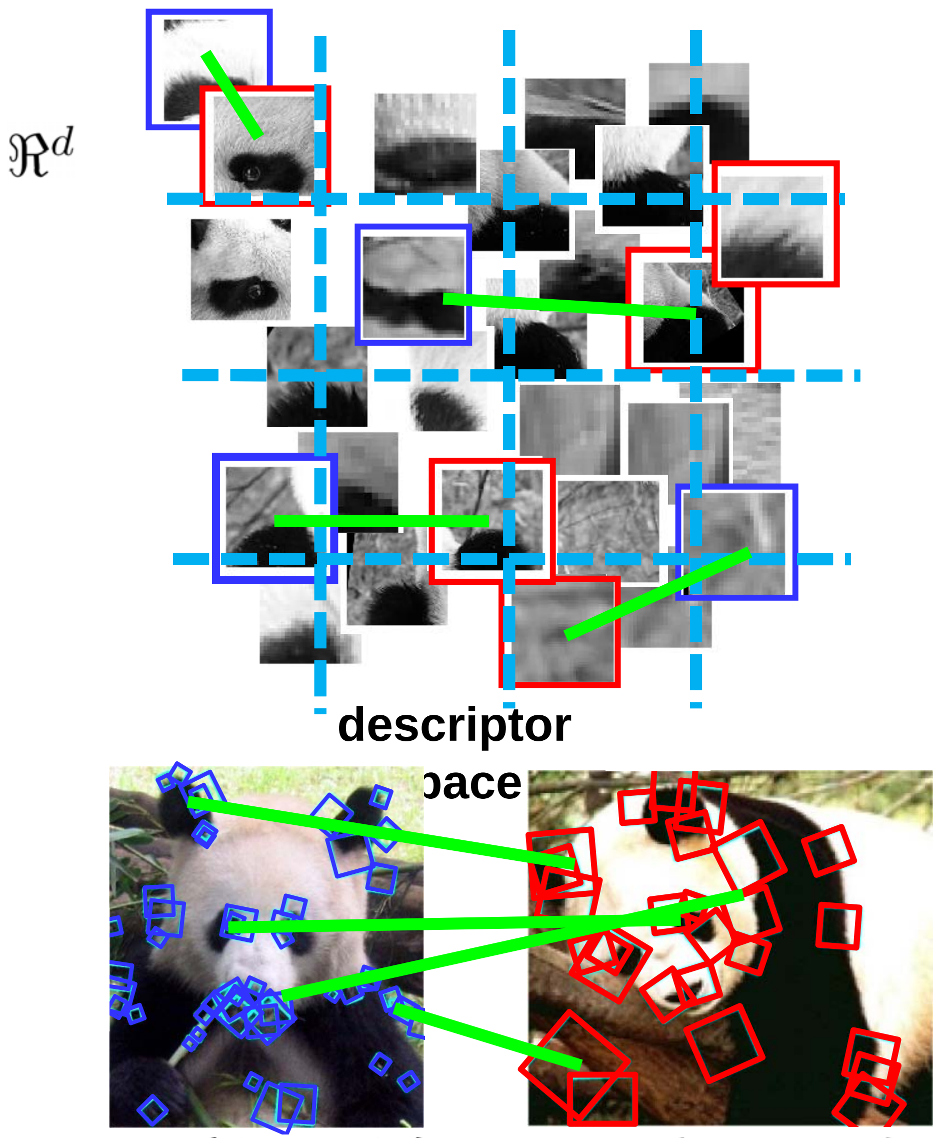
Pyramid Match Overview

Pyramid match measures similarity of a partial matching between two sets:

- Place multi-dimensional, multi-resolution grid over point sets
- Consider points matched at finest resolution where they fall into same grid cell
- Approximate optimal similarity with worst case similarity within pyramid cell

No explicit search for matches!

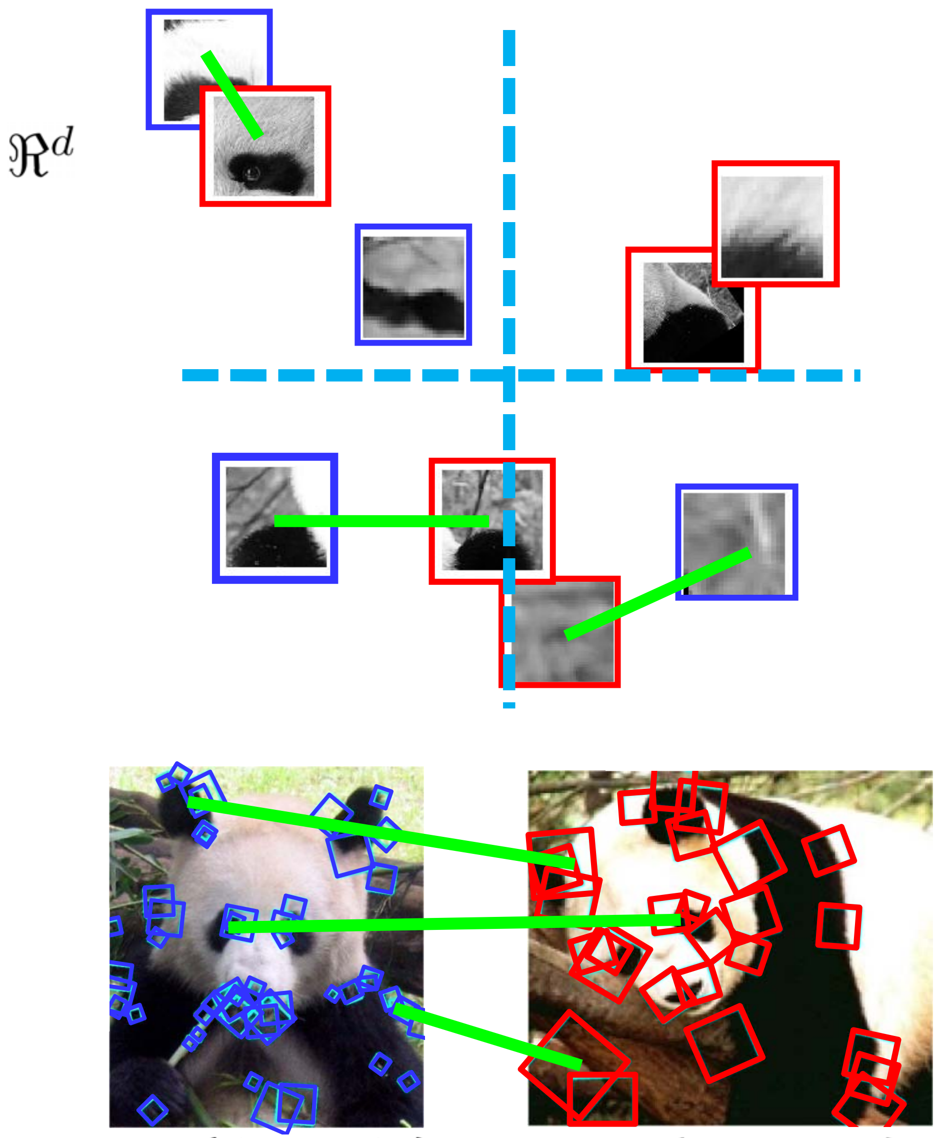
Pyramid match: main idea



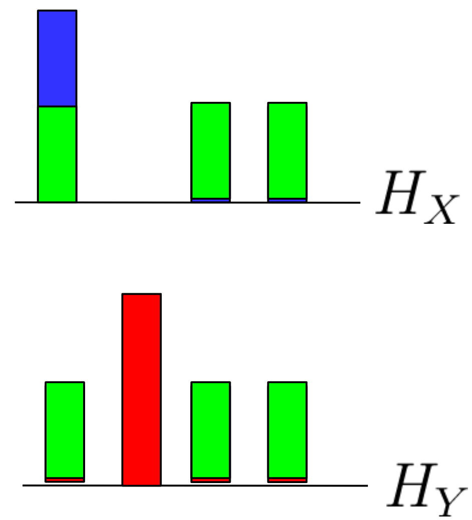
$\mathbf{X} = \{\vec{\mathbf{x}}_1, \ldots, \vec{\mathbf{x}}_m\}$ $\mathbf{Y} = \{\vec{\mathbf{y}}_1, \ldots, \vec{\mathbf{y}}_n\}$

Feature space partitions serve to “match” the local descriptors within successively wider regions.

Pyramid match: main idea



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



$$\mathcal{I}(H_X, H_Y) = \sum_j \min\left(H_X(j), H_Y(j)\right)$$
$$= 3$$

Histogram intersection counts number of possible matches at a given partitioning.

Pyramid match kernel

$$K_\Delta(X,Y) = \sum_{i=0}^{L} \underbrace{2^{-i}}_{\substack{\text{measures}\\ \text{difficulty of a}\\ \text{match at level } i}} \underbrace{\mathcal{I}\left(H_X^{(i)}, H_Y^{(i)}\right) - \mathcal{I}\left(H_X^{(i-1)}, H_Y^{(i-1)}\right)}_{\substack{\text{number of newly matched}\\ \text{pairs at level } i}}$$

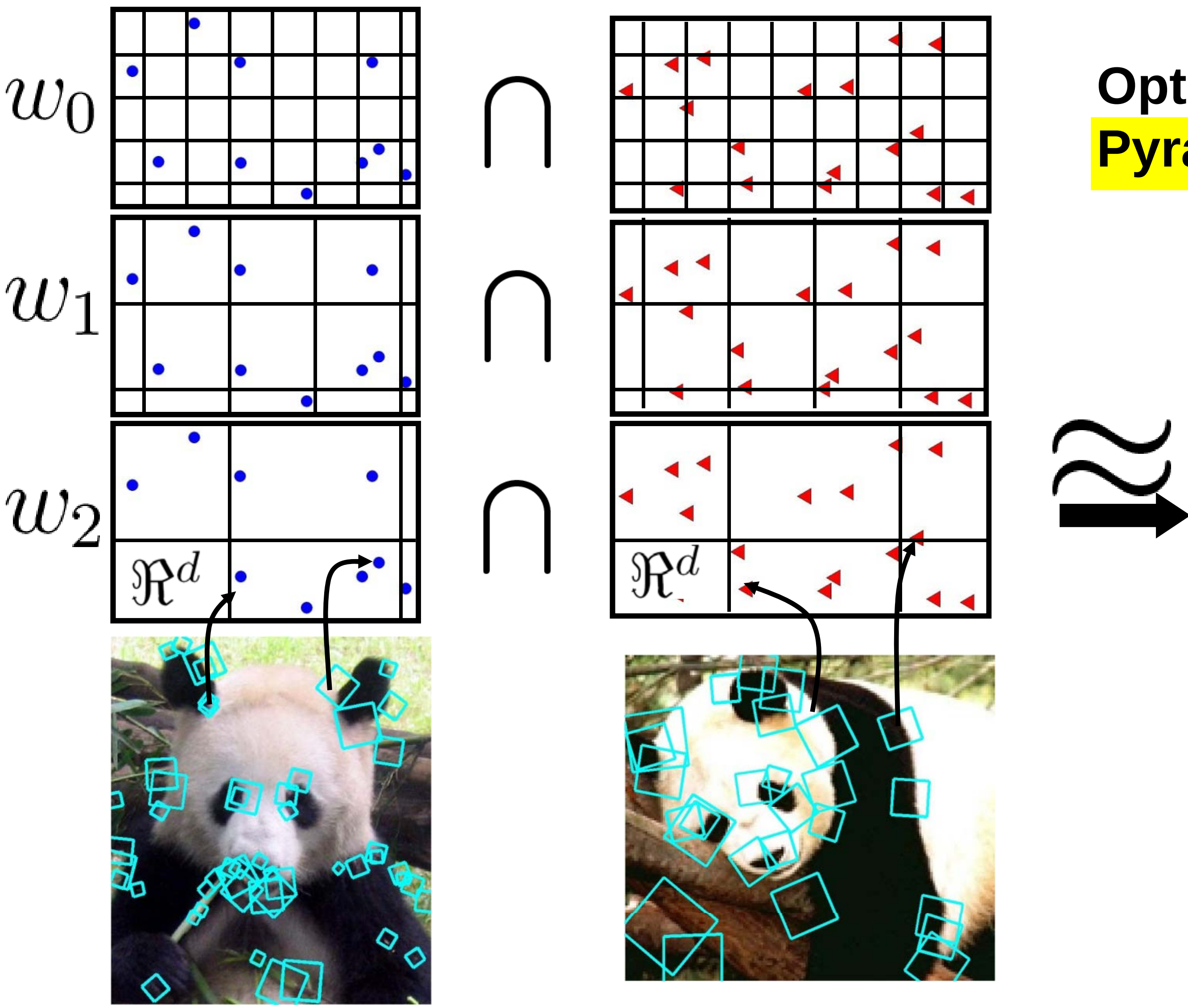
- For similarity, weights inversely proportional to bin size (or may be learned)
- Normalize these kernel values to avoid favoring large sets

[Grauman & Darrell, ICCV 2005]

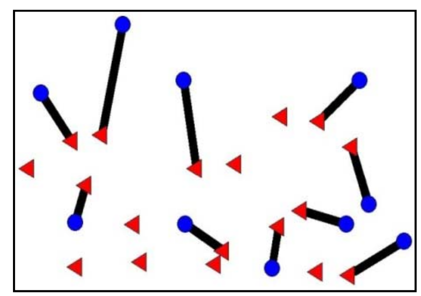
Pyramid match kernel

Optimal match: O(m³)

Pyramid match: O(mL)



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



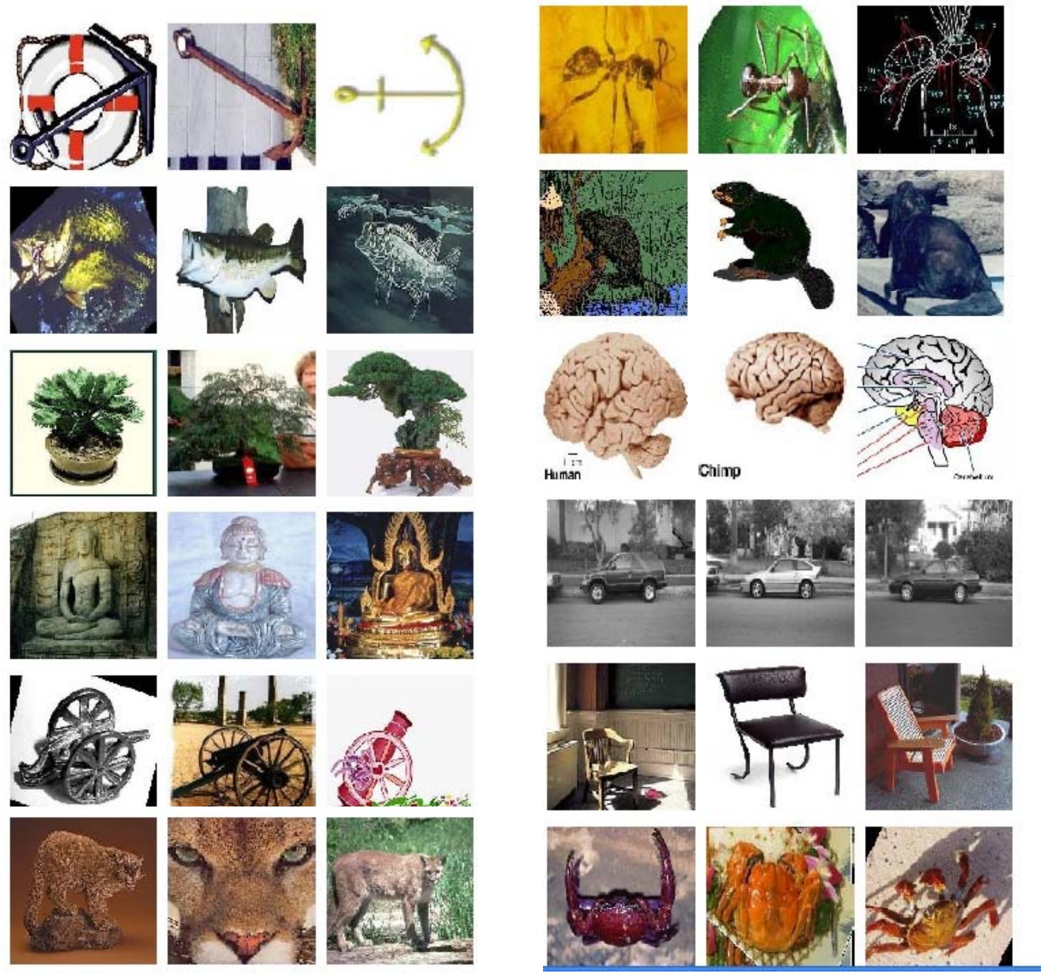
optimal partial matching

Highlights of the pyramid match

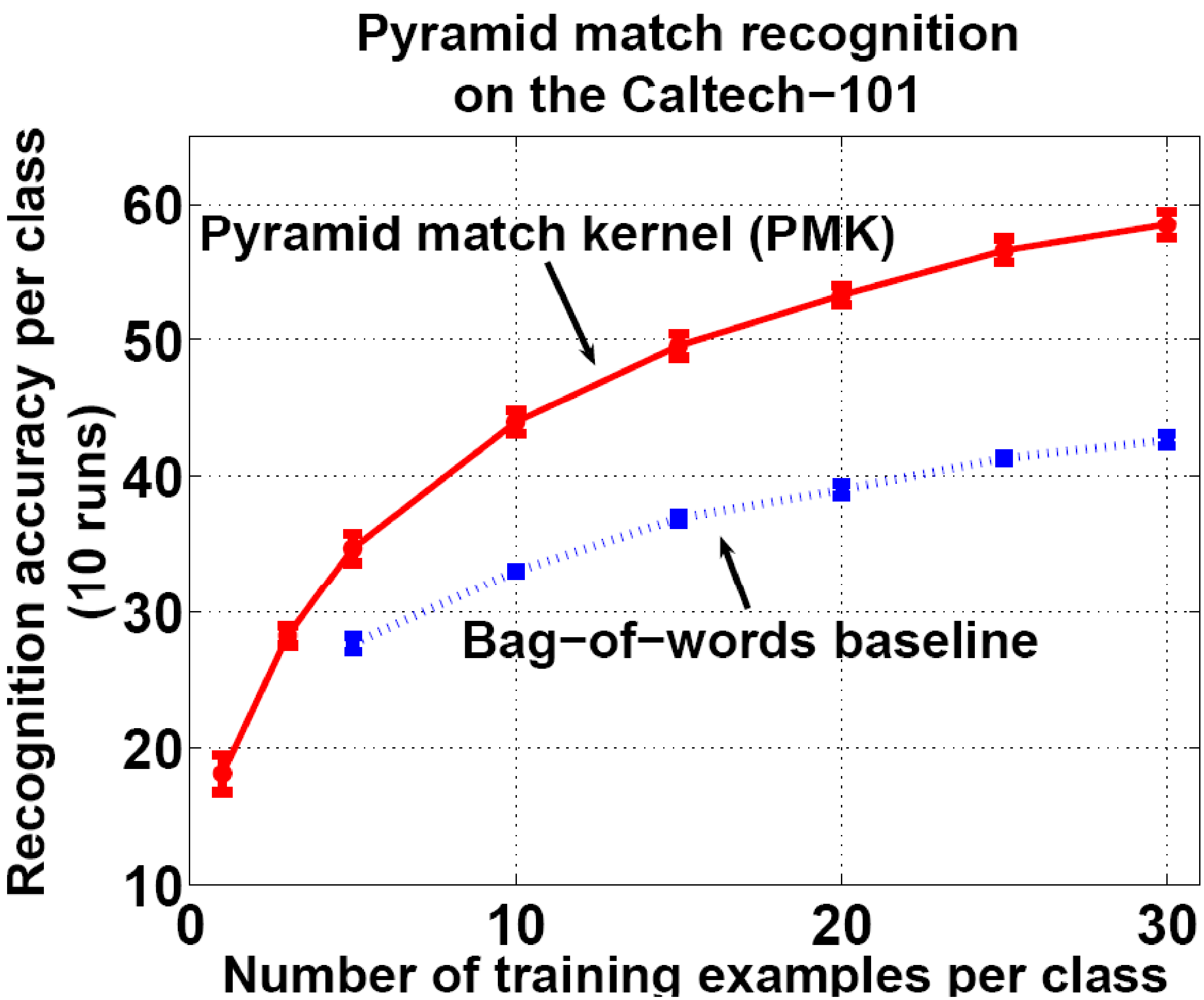
- Linear time complexity
- Formal bounds on expected error
- Mercer kernel
- Data-driven partitions allow accurate matches even in high-dim. feature spaces
- Strong performance on benchmark object recognition datasets

Recognition results:
Caltech-101 dataset

- 101 categories
  40-800 images per class

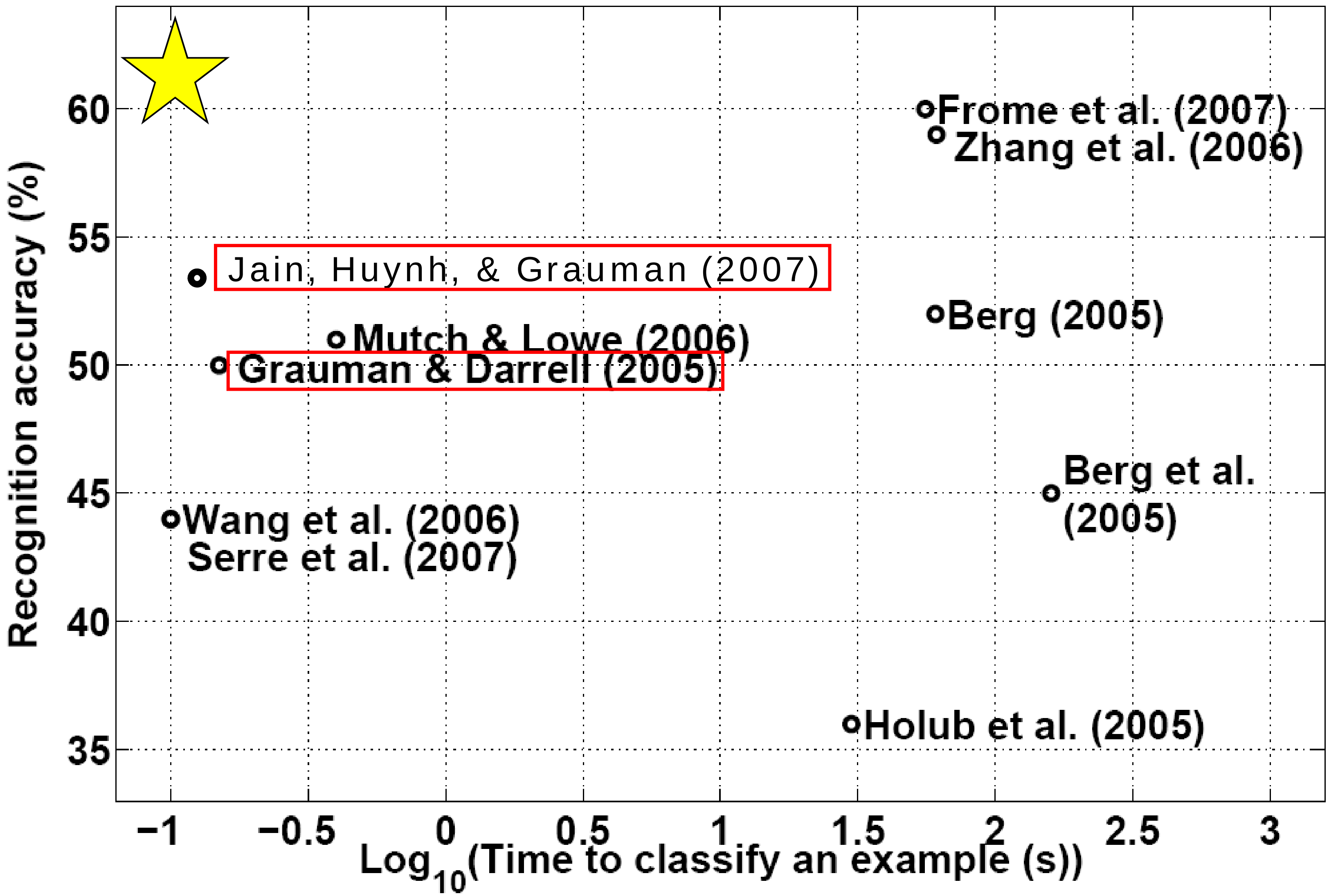


Data provided by Fei-Fei, Fergus, and Perona

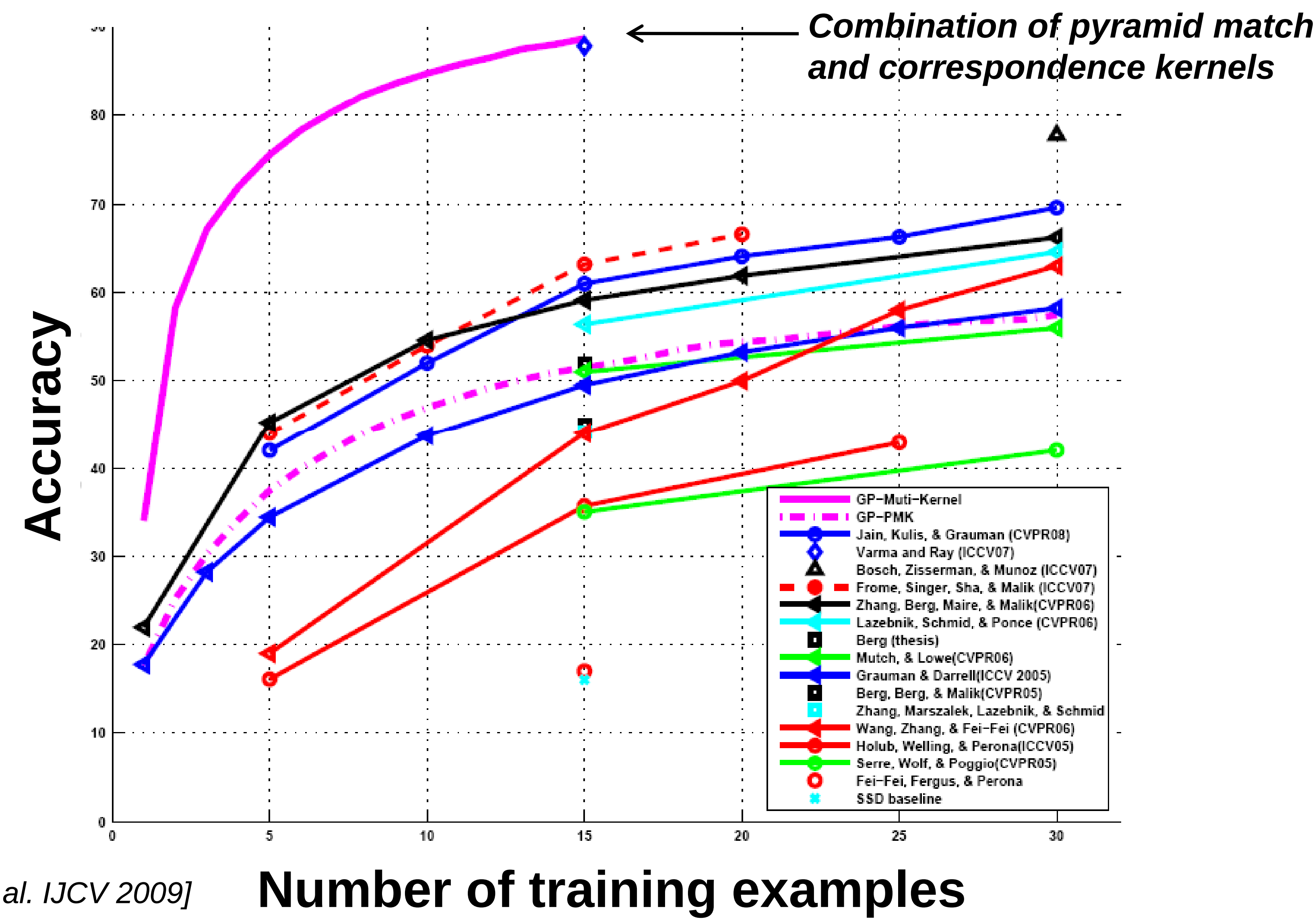

Pyramid match recognition
on the Caltech−101
Recognition accuracy per class
(10 runs)
Number of training examples per class
Pyramid match kernel (PMK)
Bag−of−words baseline
60
50
40
30
20
10
0
10
20
30

Recognition results:

Caltech-101 dataset

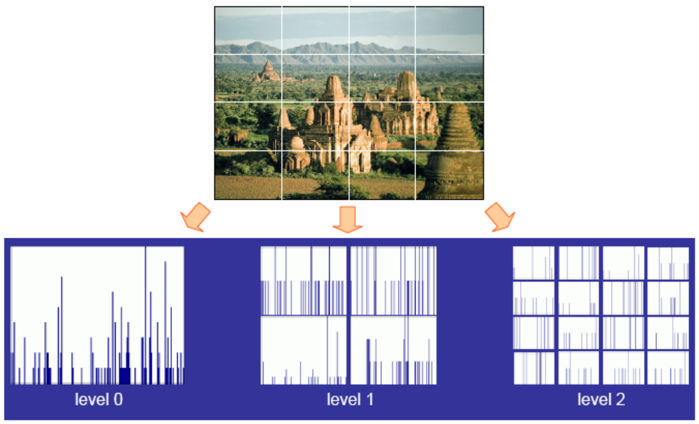


Recognition results:
Caltech-101 dataset



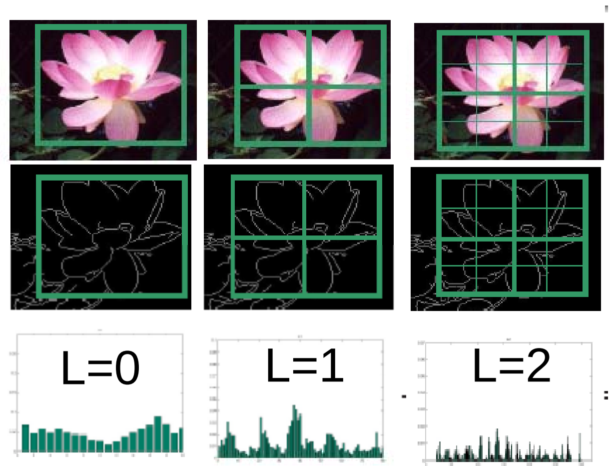
[Kapoor et al. IJCV 2009]

Pyramid match kernel: examples of extensions and applications by other groups



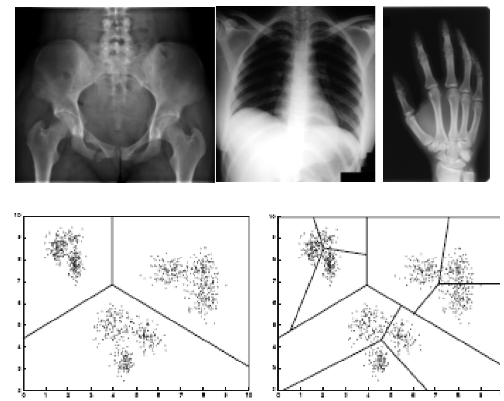
Spatial Pyramid Match Kernel
Lazebnik, Schmid, Ponce, 2006.

Scene recognition



Representing Shape with a Pyramid Kernel
Bosch & Zisserman, 2007.

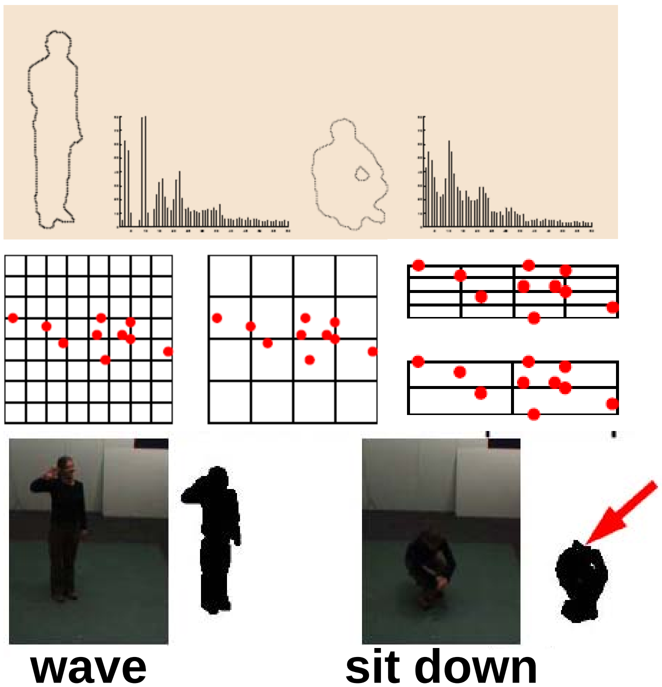
Shape representation



Dual-space Pyramid Matching
Hu et al., 2007.

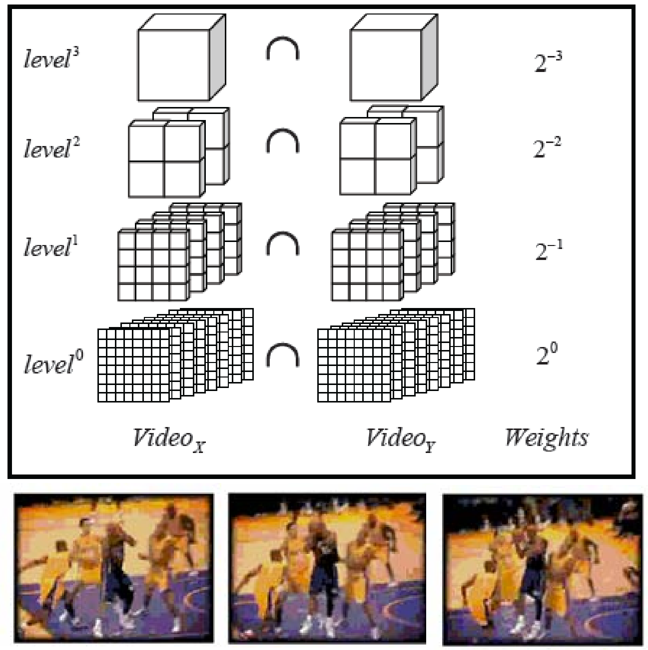
Medical image classification

Pyramid match kernel: examples of extensions and applications by other groups



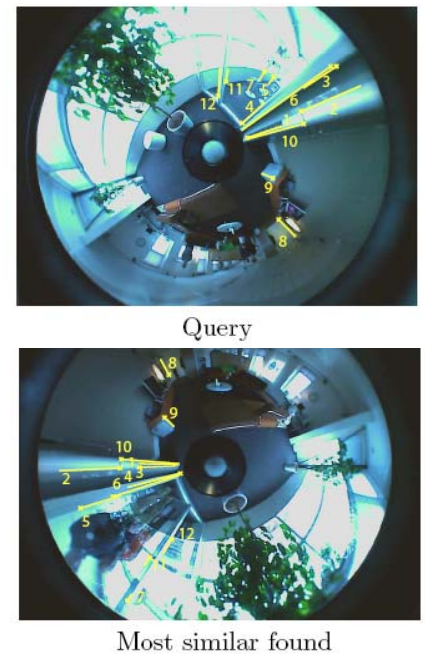
Single View Human Action Recognition using Key Pose Matching, Lv & Nevatia, 2007.

Action recognition



Spatio-temporal Pyramid Matching for Sports Videos, Choi et al., 2008.

Video indexing

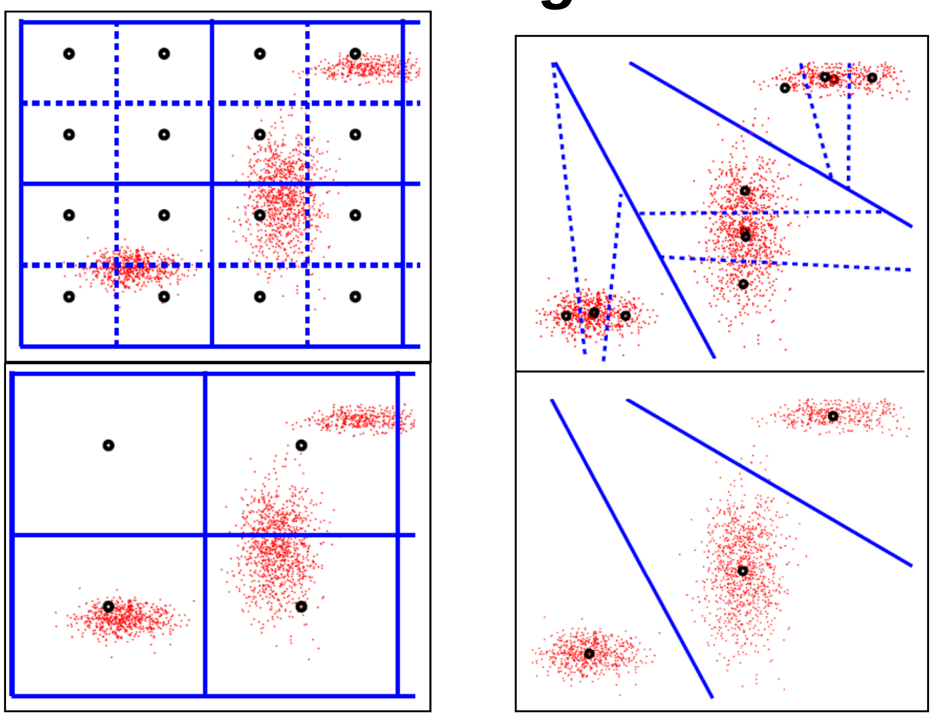


From Omnidirectional Images to Hierarchical Localization, Murillo et al. 2007.

Robot localization

Vocabulary-guided pyramid match

Uniform bins **_Vocabulary-guided_ bins**



[Grauman & Darrell, NIPS 2006]

- Tune pyramid partitions to the feature distribution
- Accurate for *d* > 100
- Requires initial corpus of features to determine pyramid structure
- Small cost increase over uniform bins: *kL* distances against bin centers to insert points

Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories

Svetlana Lazebnik, Cordelia Schmid, Jean Ponce

Slides by: Lubomir Bourdev

Slide credits: Svetlana Lazebnik

Key Idea

- Pyramid Match Kernel (Grauman & Darrell)

  *Pyramid in **feature** space, ignore location*

- Spatial Pyramid (this work)

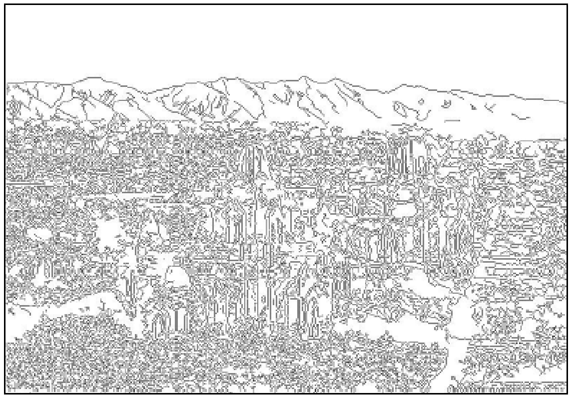
  *Pyramid in **image** space, quantize features*

Algorithm

1. Extract interest point descriptors (dense scan)
2. Construct visual word dictionary
3. Build spatial histograms
4. Create intersection kernels
5. Train an SVM

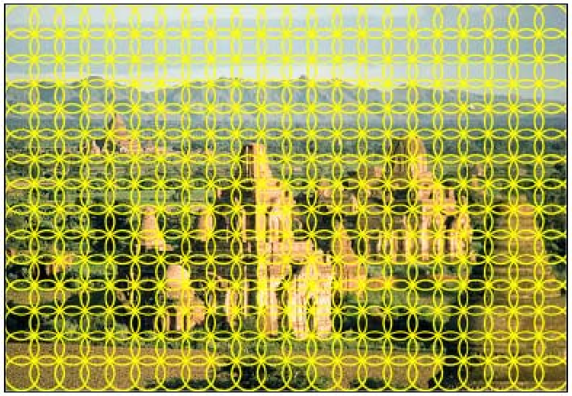
Algorithm

1. Extract interest point descriptors (dense scan)
2. Construct visual word dictionary
3. Build spatial histograms
4. Create intersection kernels
5. Train an SVM



Weak (edge orientations)

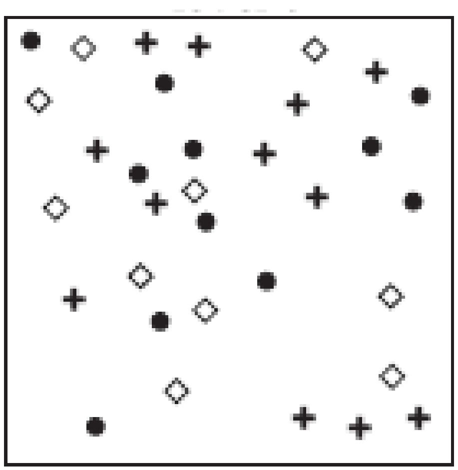
OR



Strong (SIFT)

Algorithm

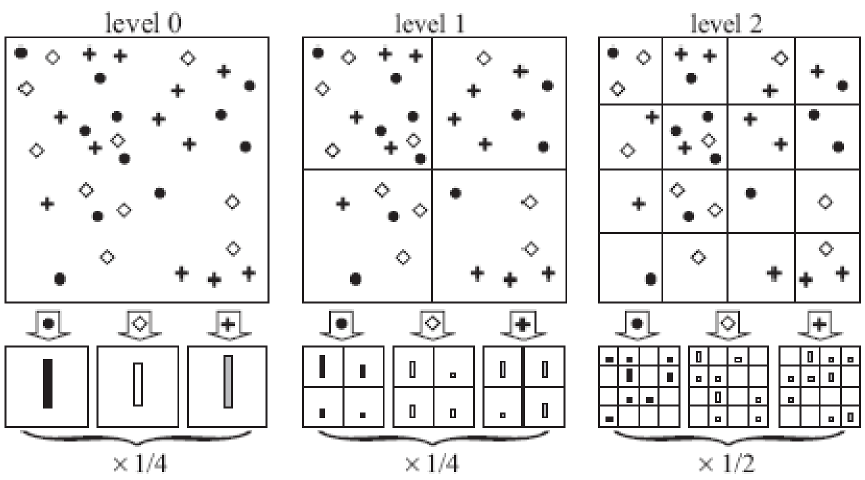
1. Extract interest point descriptors (dense scan)
2. Construct visual word dictionary
3. Build spatial histograms
4. Create intersection kernels
5. Train an SVM



- Vector quantization
- Usually K-means clustering
- Vocabulary size (16 to 400)

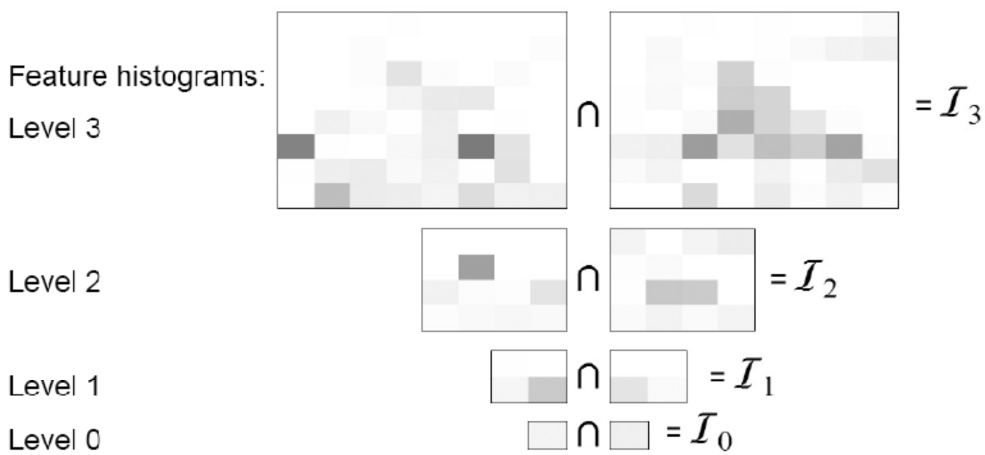
Algorithm

1. Extract interest point descriptors (dense scan)
2. Construct visual word dictionary
3. Build spatial histograms
4. Create intersection kernels
5. Train an SVM



Algorithm

1. Extract interest point descriptors (dense scan)
2. Construct visual word dictionary
3. Build spatial histograms
4. Create intersection kernels
5. Train an SVM



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

Algorithm

1. Extract interest point descriptors (dense scan)
2. Construct visual word dictionary
3. Build spatial histograms
4. Create intersection kernels
5. Train an SVM

Scene category dataset

Fei-Fei & Perona (2005), Oliva & Torralba (2001)

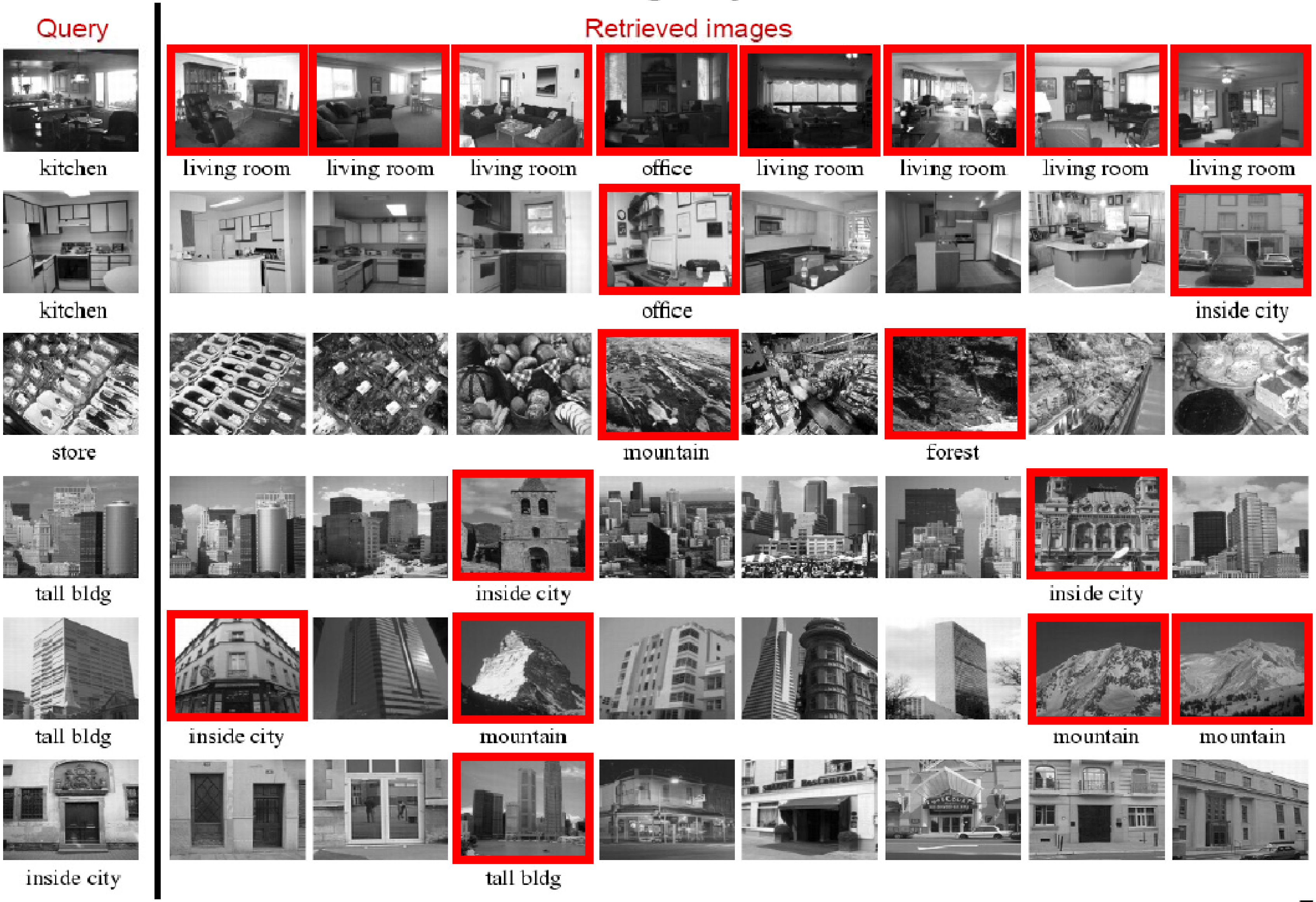
http://www-cvr.ai.uiuc.edu/ponce_grp/data

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

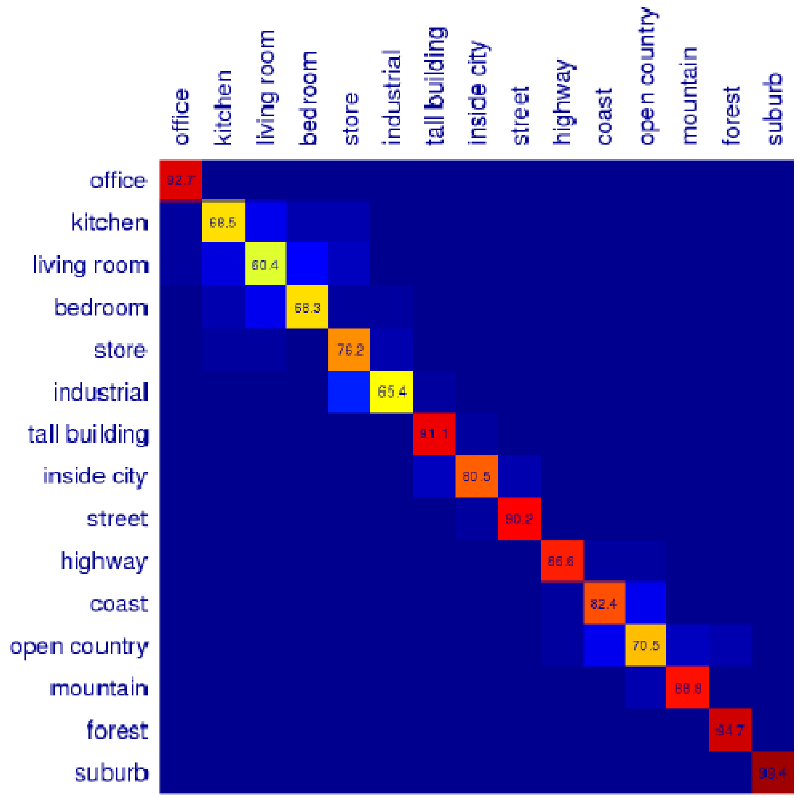
	Weak features (vocabulary size: 16)		Strong features (vocabulary size: 200)	
Level	Single-level	Pyramid	Single-level	Pyramid
0 (1×1)	45.3 $\pm$0.5		72.2 $\pm$0.6	
1 (2×2)	53.6 $\pm$0.3	56.2 $\pm$0.6	77.9 $\pm$0.6	79.0 $\pm$0.5
2 (4×4)	61.7 $\pm$0.6	64.7 $\pm$0.7	79.4 $\pm$0.3	**81.1** $\pm$0.3
3 (8×8)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

Fei-Fei & Perona: 65.2%

Scene category retrieval



Scene category confusions



Difficult indoor images

kitchen

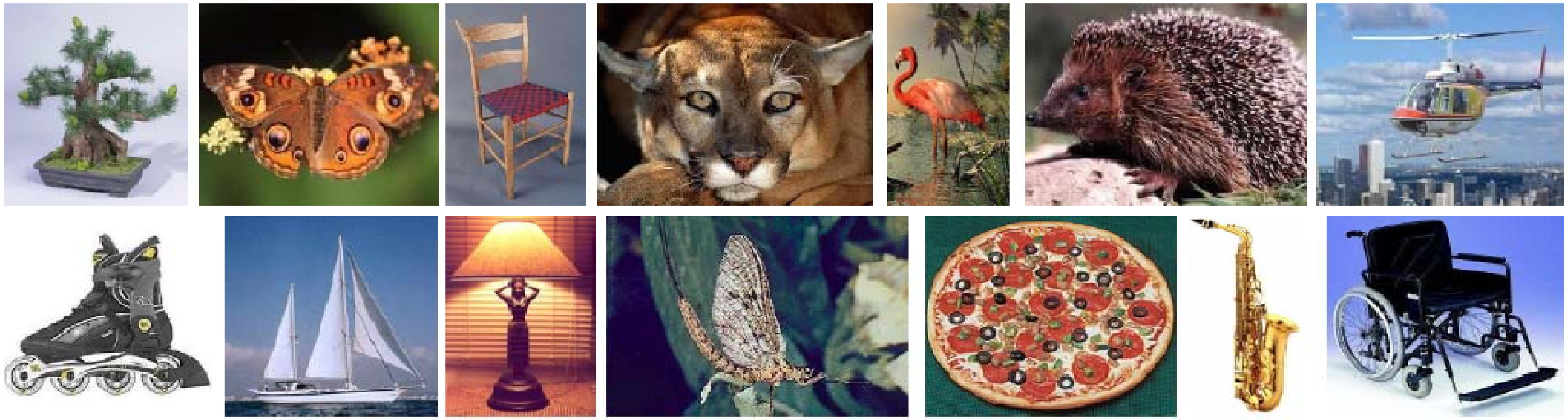
living room

bedroom

Caltech101 dataset

Fei-Fei et al. (2004)

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

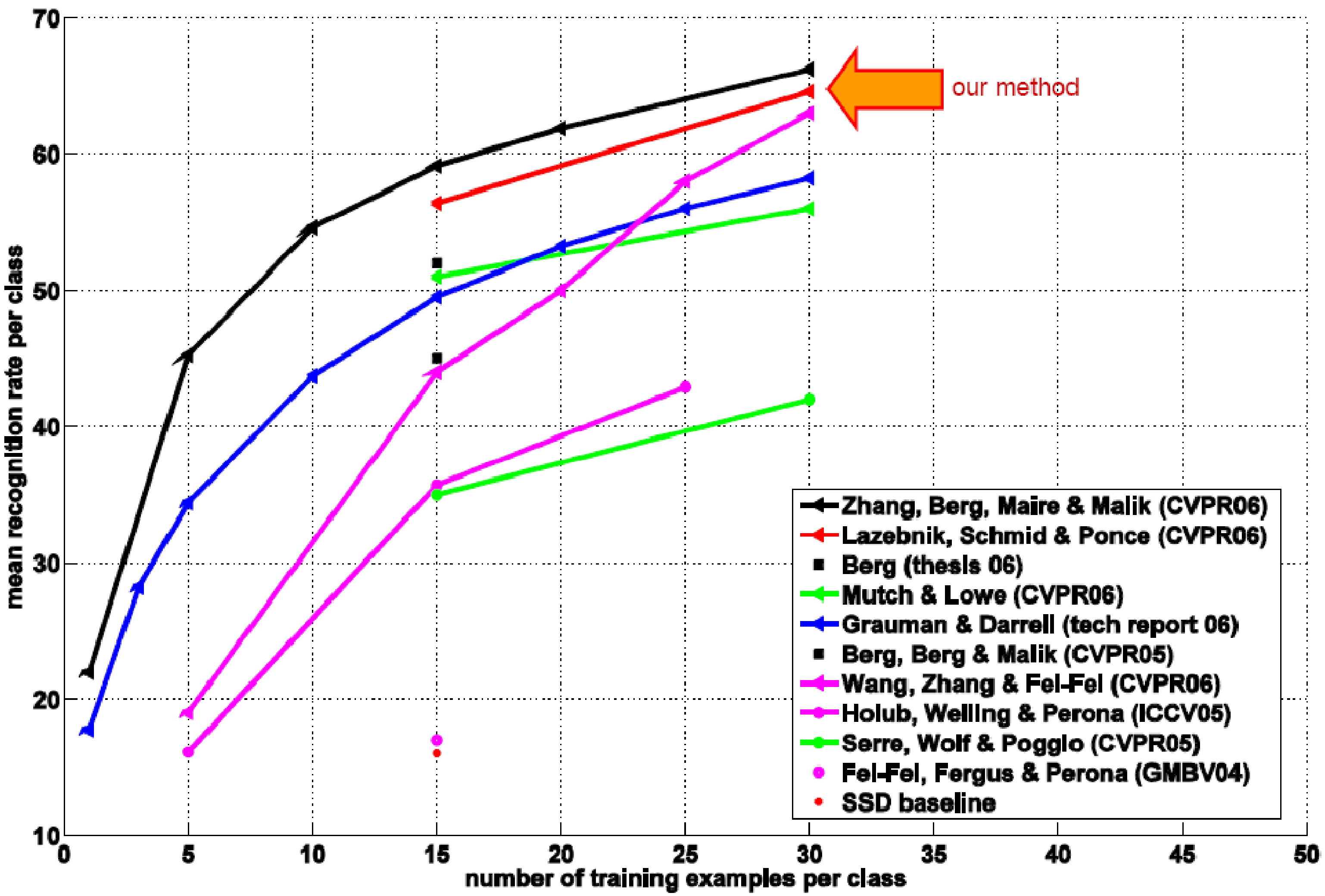


Multi-class classification results (30 training images per class)

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

Caltech101 comparison

Zhang, Berg, Maire & Malik, 2006

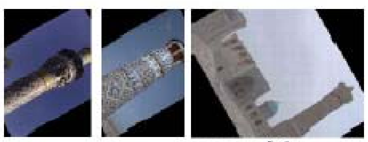


Caltech101 challenges

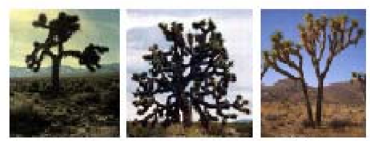
Top five confusions

class 1 / class 2	class 1 mis-classified as class 2	class 2 mis-classified as class 1
ketch / schooner	21.6	14.8
lotus / water lily	15.3	20.0
crocodile / crocodile head	10.5	10.0
crayfish / lobster	11.3	9.1
flamingo / ibis	9.5	10.4

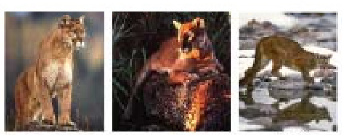
Easiest and hardest classes


minaret (97.6%)

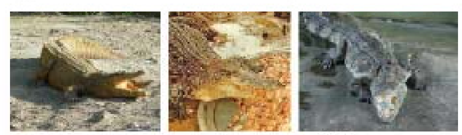

windsor chair (94.6%)


joshua tree (87.9%)


okapi (87.8%)


cougar body (27.6%)


beaver (27.5%)


crocodile (25.0%)


ant (25.0%)

- Sources of difficulty: lack of texture, camouflage, “thin” objects, highly deformable shape

PMK/SIFT Best Categories (1-5)

100%

100%

99.7%

99.1%

98.2%

PMK/SIFT Best Categories (6-10)

97.7%

97.4%

95.7%

95.3%

95.2%

PMK/SIFT 5 Worst Categories

7.7%

11.2%

11.5%

11.8%

12.3%

PMK/SIFT Most Confused Category Pairs

schooner

schoo·ner n.
A fore-and-aft rigged sailing vessel having at least two masts, with a foremast that is usually smaller than the other masts.

ketch

ketch n. Nautical
A two-masted fore-and-aft-rigged sailing vessel with a mizzenmast stepped aft of a taller mainmast but forward of the rudder.

PMK/SIFT Most Confused Category Pairs

lotus

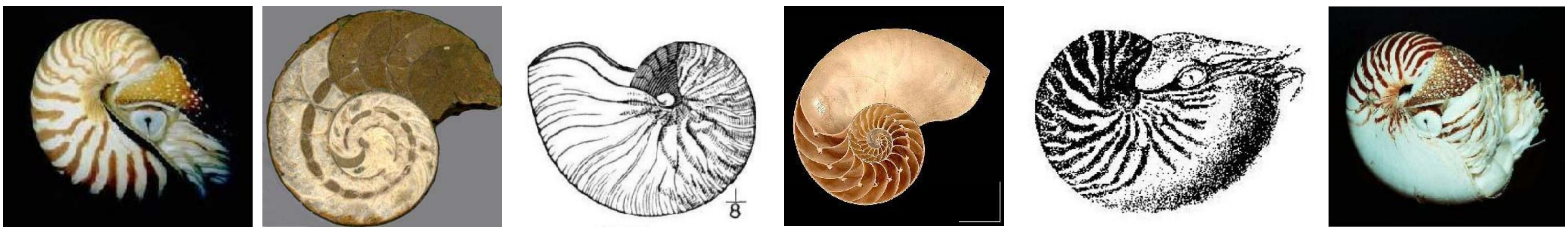
water lily

PMK/SIFT Most Confused Category Pairs

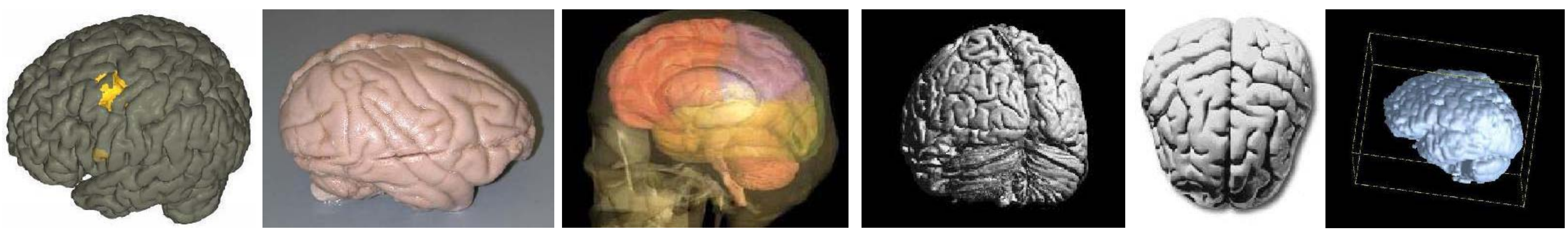
gerenuk

kangaroo

PMK/SIFT Most Confused Category Pairs



nautilus



brain

Fast intersection kernel SVMs for Realtime Object Detection

Subhransu Maji
UC Berkeley

Joint work with:
Alex Berg (Columbia University & UC Berkeley)
and
Jitendra Malik (UC Berkeley)

Fast intersection kernel SVMs for Realtime Object Detection

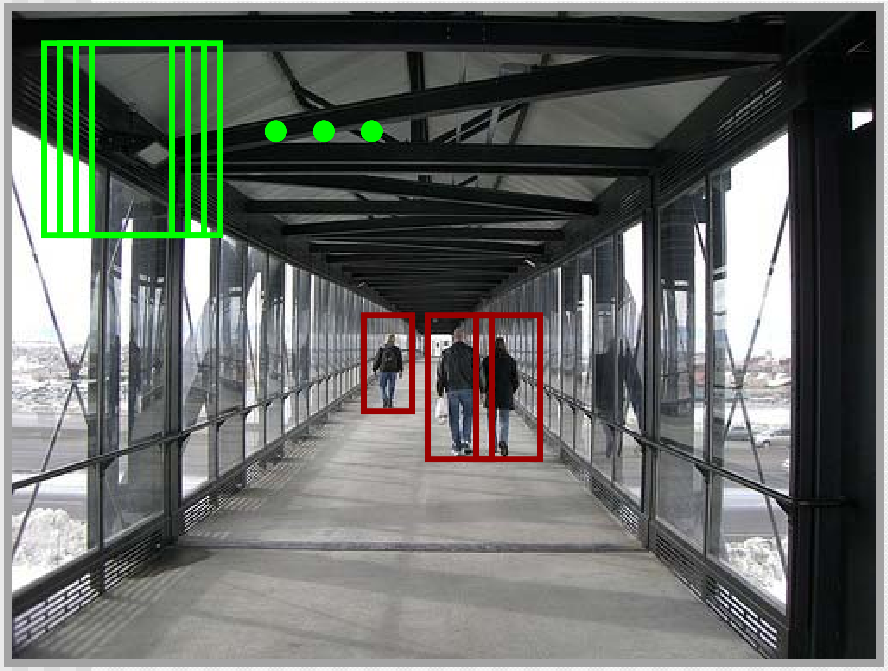
- IKSVM is a (simple) generalization of a linear SVM
- Can be evaluated very efficiently (sublinear in #SV)
- Other kernels (including $-\chi^2$) have a similar form
- Methods applicable to current most successful object recognition/detection strategies.

Maji, Berg & Malik, CVPR 2008

Detection: Is this an X?

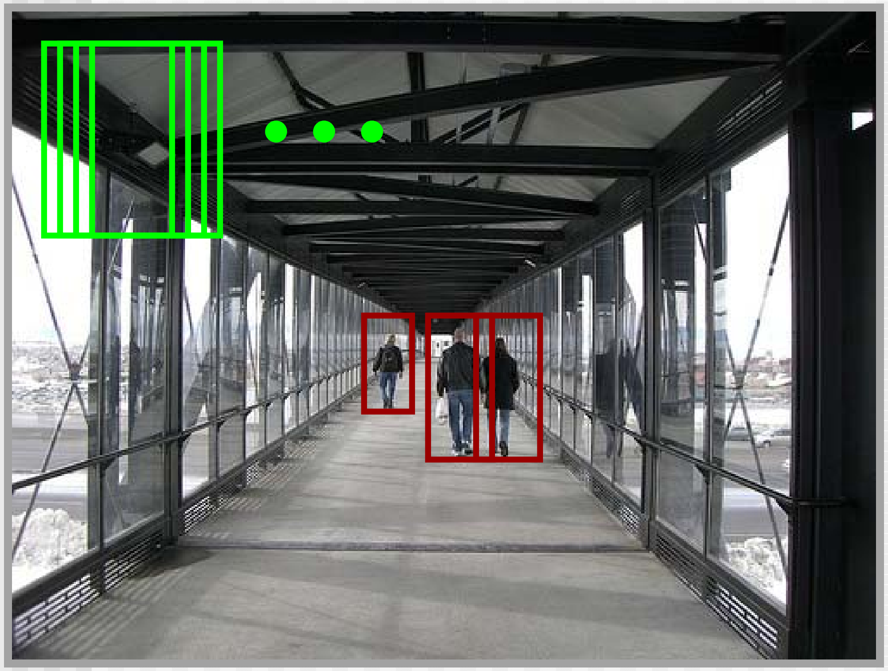
Ask this question over and over again,
varying position, scale, category, pose…
Speedups: hierarchical, early reject, feature sharing, cueing
but same underlying question!

Detection: Is this an X?



Ask this question over and over again,
varying position, scale, multiple categories…
Speedups: hierarchical, early reject, feature sharing,
but same underlying question!

Detection: Is this an X?



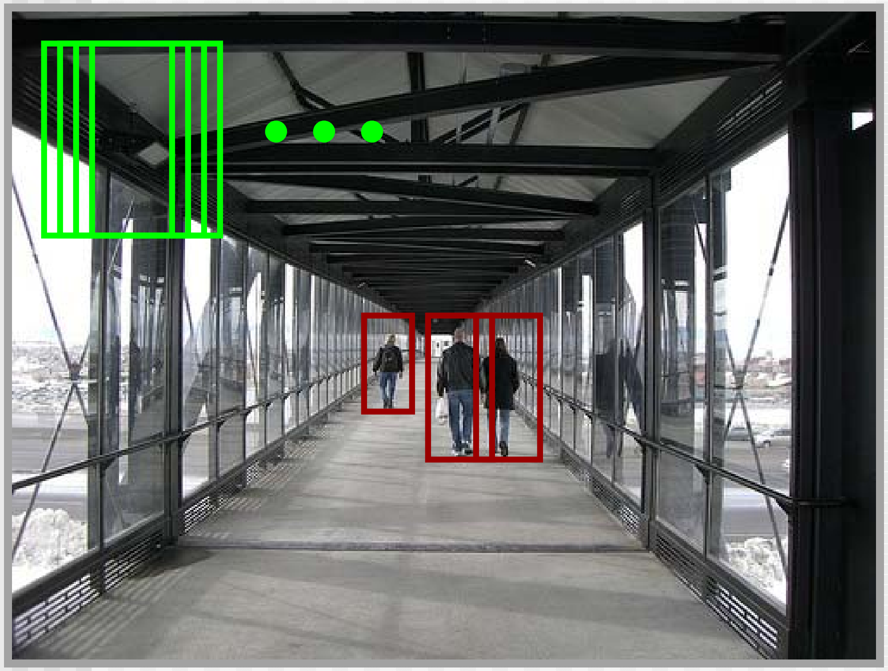
Boosted dec. trees, cascades
+ Very fast evaluation
- Slow training (esp. multi-class)

Linear SVM
+ Fast evaluation
+ Fast training
- Need to find good features

Non-linear kernelized SVM
+ Better class. acc. than linear
- Medium training
- Slow evaluation

Ask this question over and over again,
varying position, scale, multiple categories…
Speedups: hierarchical, early reject, feature sharing,
but same underlying question!

Detection: Is this an X?



Boosted dec. trees, cascades
 + Very fast evaluation
 - Slow training (esp. multi-class)

Linear SVM
 + Fast evaluation
 + Fast training
 - Need to find good features

This work

Non-linear kernelized SVM
 + Better class. acc. than linear
 - Medium training
 - Slow evaluation

Ask this question over and over again,
varying position, scale, multiple categories…
Speedups: hierarchical, early reject, feature sharing,
but same underlying question!

Outline

- What is Intersection Kernel SVM?
  - Brief Overview of Support Vector Machines
  - Multi-scale features based on Oriented Energy
- Algorithms
  - Algorithm to make classification fast (exact)
  - Algorithm to make classification very fast (approximate)
- Experimental Results
- Summary of where this matters

Outline

- What is Intersection Kernel SVM?
  - Brief Overview of Support Vector Machines
  - Multi-scale features based on Oriented Energy
- Algorithms
  - Algorithm to make classification fast (exact)
  - Algorithm to make classification very fast (approximate)
- Experimental Results
- Summary of where this matters

Support Vector Machines

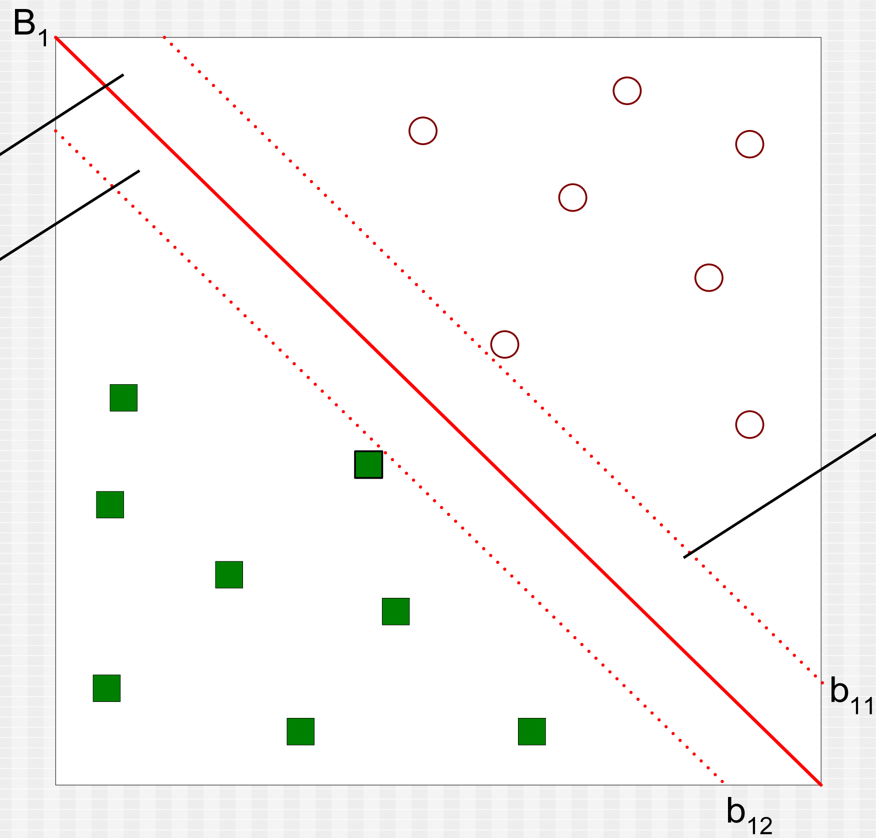
Examples are;

(x1,..,xn,y) with

y∈{-1.1}

$$\vec{w} \bullet \vec{x} + b = 0$$

$$\vec{w} \bullet \vec{x} + b = -1$$

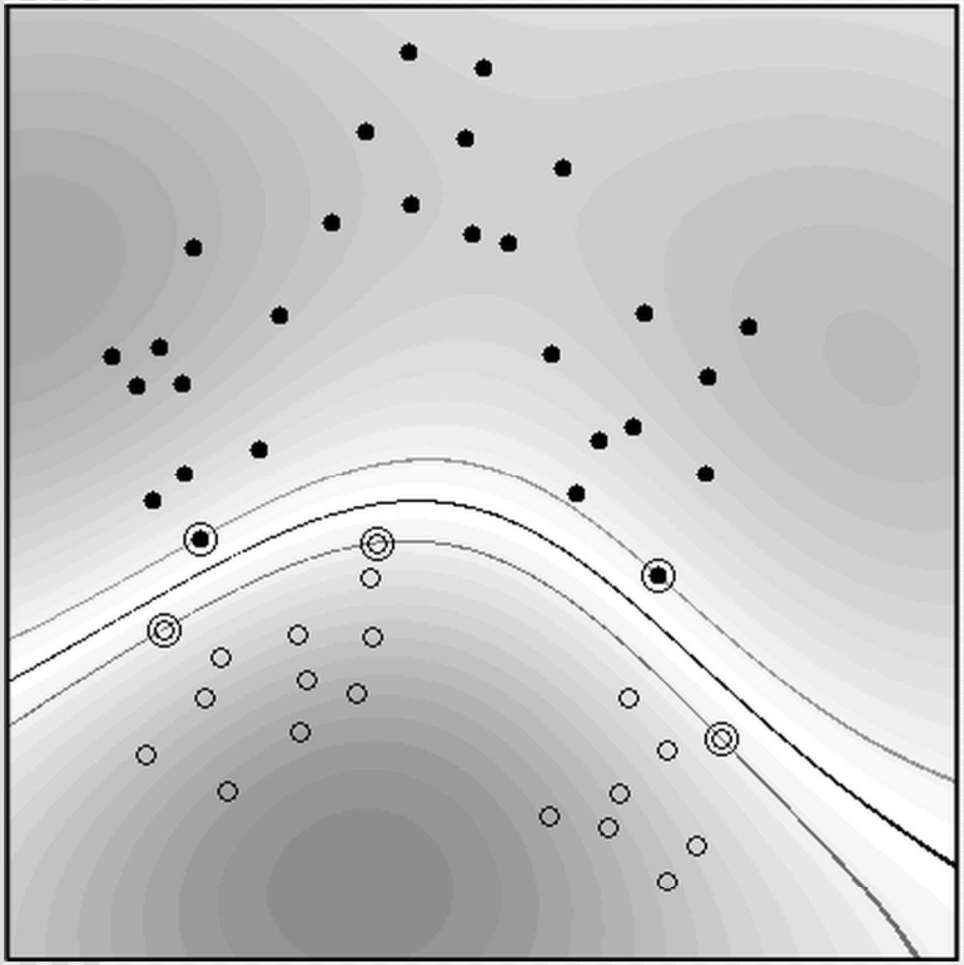


$$\vec{w} \bullet \vec{x} + b = +1$$

$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x} + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x} + b \leq -1 \end{cases}$$

$$\text{Margin} = \frac{2}{\|\vec{w}\|^2}$$

Kernel Support Vector Machines



Kernel Function

- Inner Product in Hilbert Space
- Learn Non Linear Boundaries

$$\mathrm{K}(\mathbf{x},\mathbf{y}) = \Phi(x)^T\Phi(y)$$

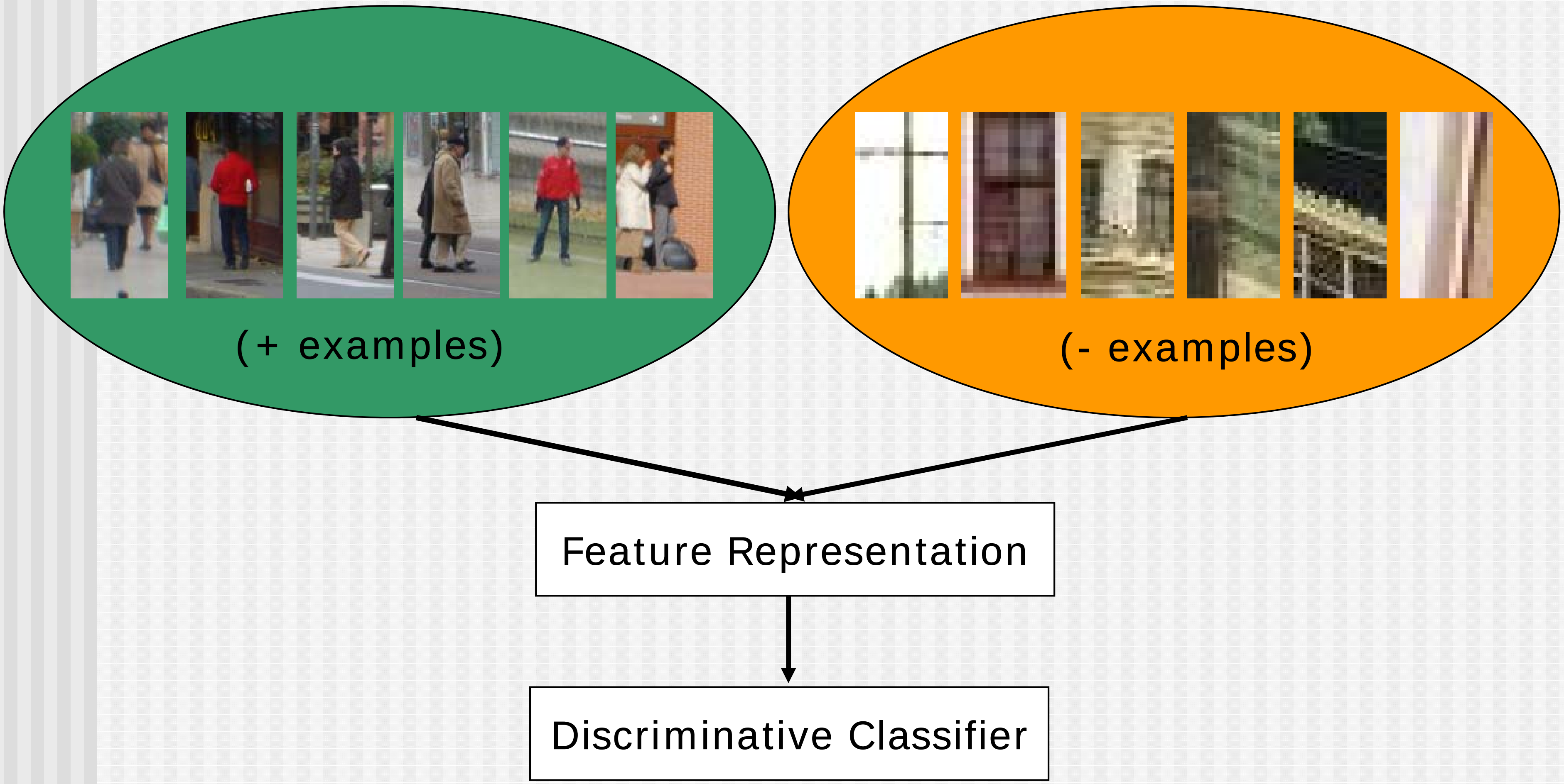
Gaussian Kernel

$$\mathrm{K}(\mathbf{x},\mathbf{y}) = \exp(-\gamma||x-y||^2)$$

Classification Function

$$\mathrm{h}(\mathbf{x}) = \mathrm{sign}\left(\sum_{i=1}^{n}\alpha_i k(x,x_i)+b\right)$$

Training Stage



Multiscale Oriented Energy feature

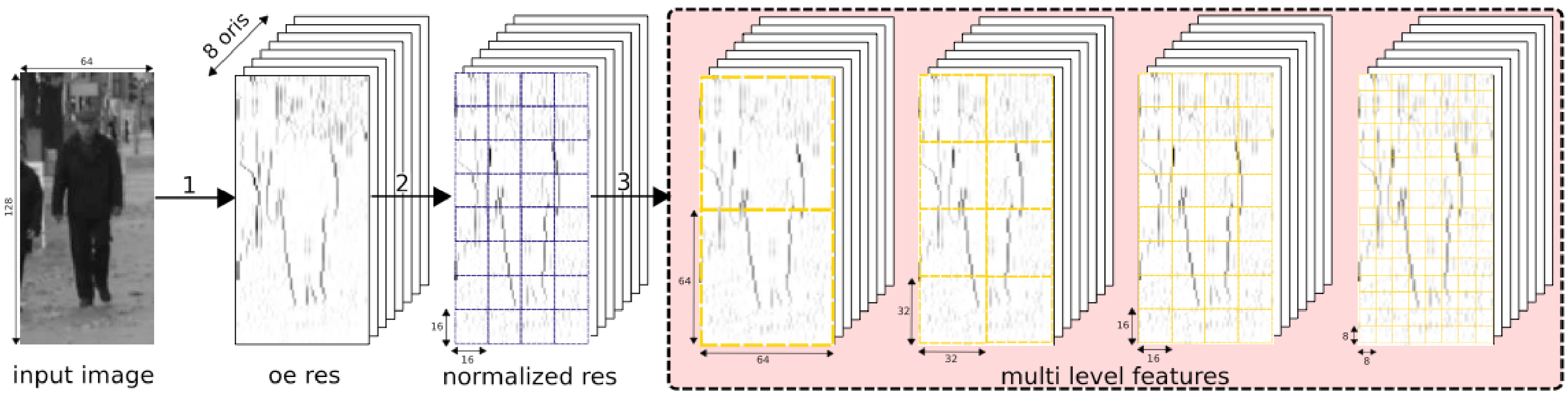


Figure 2. The three stage pipeline of the feature computation process. **(1)** The input grayscale image of size 64×128 is convolved with oriented filters ($\sigma = 1$) in 8 directions, to obtain oriented energy responses. **(2)** The responses are then L_1 normalized over all directions in each non overlapping 16×16 blocks independently to obtain normalized responses. **(3)** Multilevel features are then extracted by constructing histograms of oriented gradients by summing up the normalized response in each cell. The diagram depicts progressively smaller cell sizes from 64×64 to 8×8.

Concatenate orientation histograms for each orange region.

Differences from HOG:

-- Hierarchy of regions

-- Only performing L1 normalization once (at 16x16)

What is the Intersection Kernel?

Histogram Intersection kernel between histograms *a*, *b*

$$K(a,b) = \sum_{i=1}^{n} min(a_i, b_i)$$

$$a_i \geq 0$$
$$b_i \geq 0$$

What is the Intersection Kernel?

Histogram Intersection kernel between histograms *a*, *b*

$$K(a,b) = \sum_{i=1}^{n} min(a_i, b_i) \qquad \begin{array}{l} a_i \geq 0 \\ b_i \geq 0 \end{array}$$

K small -> *a*, *b* are different
K large -> *a*, *b* are similar

Intro. by Swain and Ballard 1991 to compare color histograms.
Odone et al 2005 proved positive definiteness.
Can be used directly as a kernel for an SVM.
Compare to χ^2
Generalizations: *Pyramid Match Kernel (Grauman et. al.),*
Spatial Pyramid Match Kernel (Lazebnik et.al)

Linear SVM, Kernelized SVM, IKSVM

Decision function is $\text{sign}\left(h(x)\right)$ where:

Linear: $$h(x) = w'x + b = \sum_{i=1}^{\#\text{dim}} w_i x_i + b$$

Non-linear Using Kernel $$h(x) = \sum_{j=1}^{\#\text{sv}} \alpha^j K(x, x^j) + b$$

Histogram Intersection Kernel $$= \sum_{j=1}^{\#\text{sv}} \left(\alpha^j \sum_{i=1}^{\#\text{dim}} \min(x_i, x_i^j) \right) + b$$

Kernelized SVMs slow to evaluate

Decision function is $\mathrm{sign}\left(h(x)\right)$ where:

Feature vector to evaluate

Sum over all support vectors

Kernel Evaluation

Feature corresponding to a support vector /

Arbitrary Kernel

$$h(x) = \sum_{j=1}^{\#\mathrm{sv}} \alpha^j K(x, x^j) + b$$

Histogram Intersection Kernel

$$h(x) = \sum_{j=1}^{\#\mathrm{sv}} \left(\alpha^j \sum_{i=1}^{\#\mathrm{dim}} \min(x_i, x_i^j) \right) + b$$

SVM with Kernel Cost:

Support Vectors x Cost of kernel comp.

IKSVM Cost:

Support Vectors x # feature dimensions

Algorithm 1

Decision function is $\mathrm{sign}\,(h(x))$ where:

$$h(x) = \sum_{j=1}^{\#\mathrm{sv}} \alpha^j \left(\sum_{i=1}^{\#\mathrm{dim}} \min(x_i, x_i^j) \right) + b$$

$$= \sum_{i=1}^{\#\mathrm{dim}} \left(\sum_{j=1}^{\#\mathrm{sv}} \alpha^j \min(x_i, x_i^j) \right) + b$$

$$= \sum_{i=1}^{\#\mathrm{dim}} h_i(x_i)$$

Just sort the support vector values in each coordinate, and pre-compute

$$h_i(x_i) = \sum_{j=1}^{\#\mathrm{sv}} \alpha^j \min(x_i, x_i^j) + b$$

$$= \sum_{x_i^j < x_i} \alpha^j x_i^j + \left(\sum_{x_i^j \geq x_i} \alpha^j \right) x_i$$

To evaluate, find position of x_i in the sorted support vector values x_i^j (cost: log #sv) look up values, multiply & add

Algorithm 1

~~#support vectors x #dimensions~~

log(#support vectors) x #dimensions

Decision function is $\text{sign}\left(h(x)\right)$ where:

$$\begin{aligned}
h(x) &= \sum_{j=1}^{\#\text{sv}} \alpha^j \left(\sum_{i=1}^{\#\text{dim}} \min(x_i, x_i^j) \right) + b \\
&= \sum_{i=1}^{\#\text{dim}} \left(\sum_{j=1}^{\#\text{sv}} \alpha^j \min(x_i, x_i^j) \right) + b \\
&= \sum_{i=1}^{\#\text{dim}} h_i(x_i)
\end{aligned}$$

$$\begin{aligned}
h_i(x_i) &= \sum_{j=1}^{\#\text{sv}} \alpha^j \min(x_i, x_i^j) + b \\
&= \sum_{x_i^j < x_i} \alpha^j x_i^j + \left(\sum_{x_i^j \geq x_i} \alpha^j \right) x_i
\end{aligned}$$

Just sort the support vector values in each coordinate, and pre-compute

To evaluate, find position of x_i in the sorted support vector values x_i^j (cost: log #sv) look up values, multiply & add

Algorithm 2

~~#support vectors x #dimensions~~

log(#support vectors) x #dimensions

Decision function is $\text{sign}\left(h(x)\right)$ where:

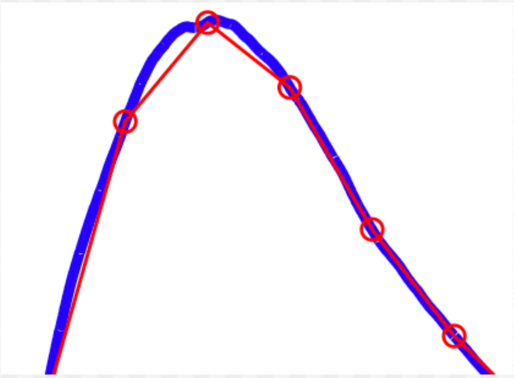
$$h(x) = \sum_{i=1}^{\#\text{dim}} \left(\sum_{j=1}^{\#\text{sv}} \alpha^j \min(x_i, x_i^j) \right) + b$$

$$= \sum_{i=1}^{\#\text{dim}} h_i(x_i)$$

$$h_i(x_i) = \sum_{j=1}^{\#\text{sv}} \alpha^j \min(x_i, x_i^j) + b$$

$$= \sum_{x_i^j < x_i} \alpha^j x_i^j + \left(\sum_{x_i^j \geq x_i} \alpha^j \right) x_i$$

For IK h_i is piecewise linear, and quite smooth, blue plot. We can *approximate* with fewer uniformly spaced segments, red plot. Saves time & space!



~~#support vectors x #dimensions~~

log(~~#support vectors) x #dimensions~~

constant x #dimensions

Algorithm 2

Decision function is $\mathrm{sign}\left(h(x)\right)$ where:

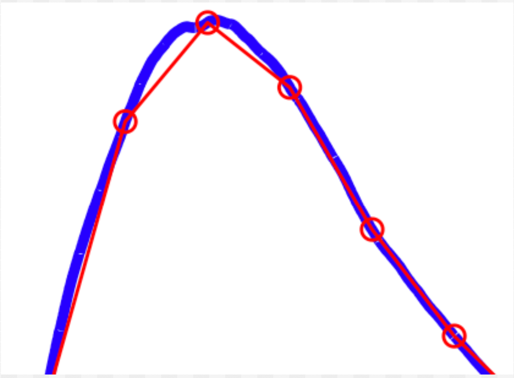
$$h(x) = \sum_{i=1}^{\#\mathrm{dim}} \left(\sum_{j=1}^{\#\mathrm{sv}} \alpha^j \min(x_i, x_i^j) \right) + b$$

$$= \sum_{i=1}^{\#\mathrm{dim}} h_i(x_i)$$

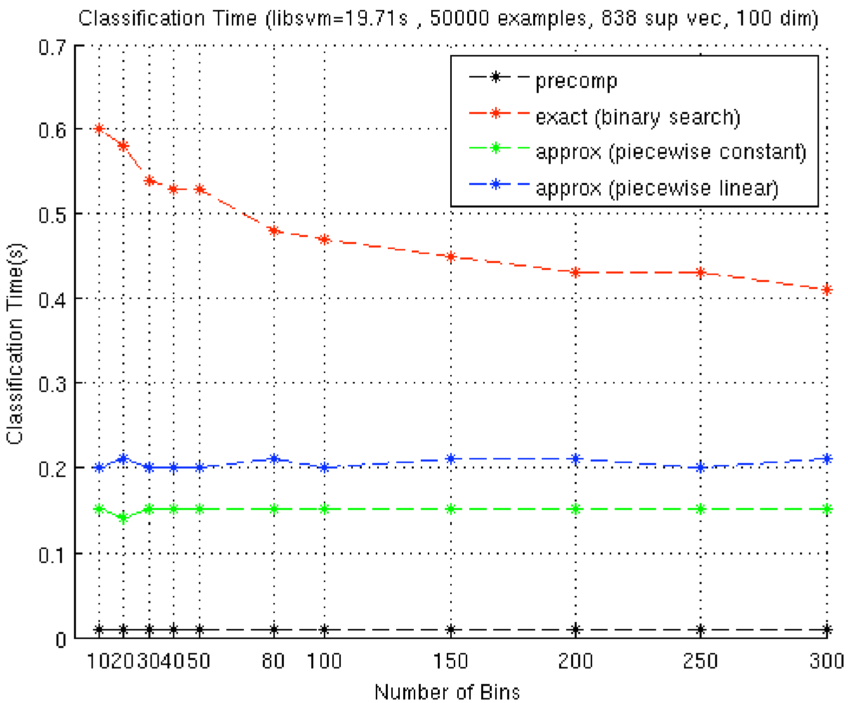
$$h_i(x_i) = \sum_{j=1}^{\#\mathrm{sv}} \alpha^j \min(x_i, x_i^j) + b$$

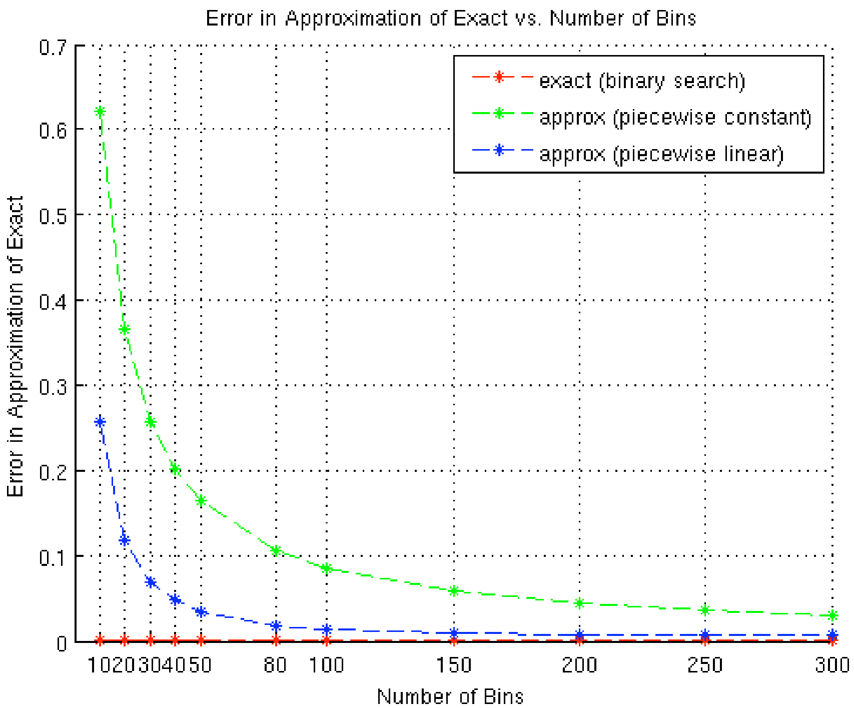
$$= \sum_{x_i^j < x_i} \alpha^j x_i^j + \left(\sum_{x_i^j \geq x_i} \alpha^j \right) x_i$$

For IK h_i is piecewise linear, and quite smooth, blue plot. We can *approximate* with fewer uniformly spaced segments, red plot. Saves time & space!



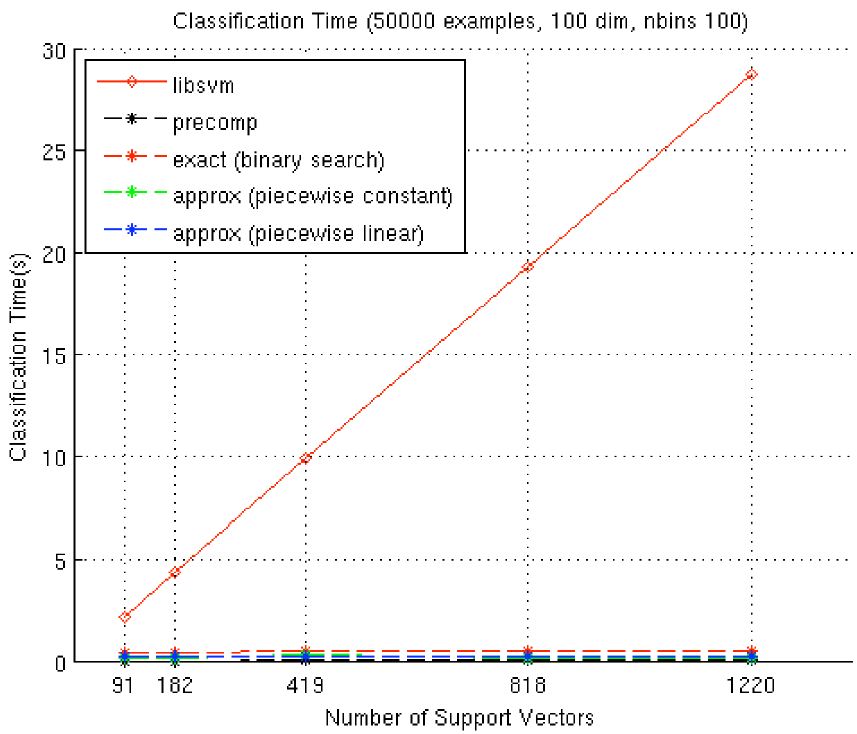
Toy Example : accuracy/runtime vs. #bins

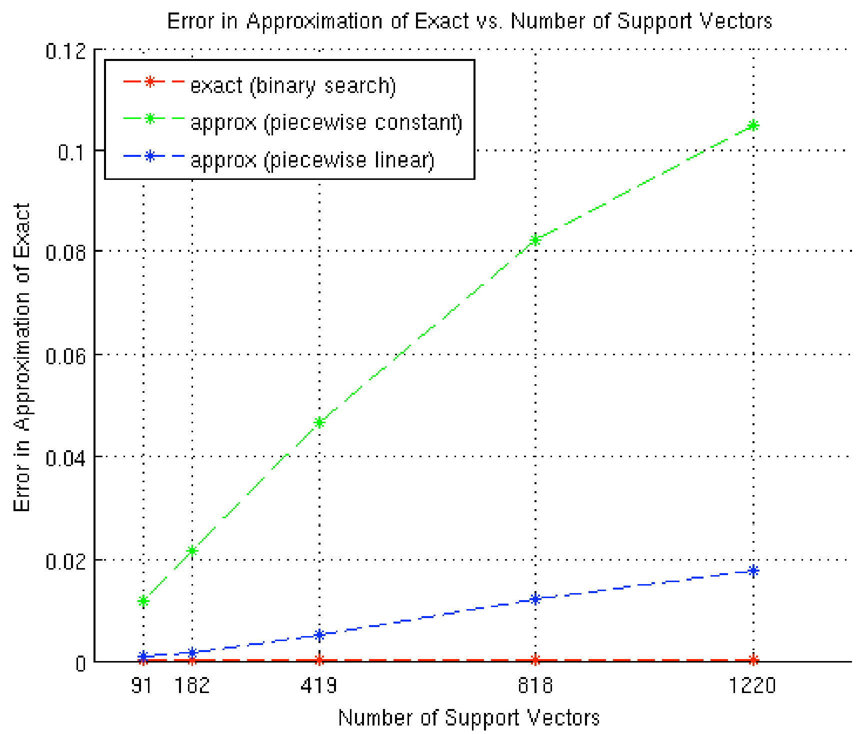




- Runtime independent of #bins (on left)
- Accuracy improves with #bins (on right)

Toy Example : accuracy/runtime vs. #sup vec





- Runtime independent of #sup vec! (for approximate)
- 2-3 orders of magnitude faster than LibSVM.
- Runtime memory requirement independent of #sup vec!

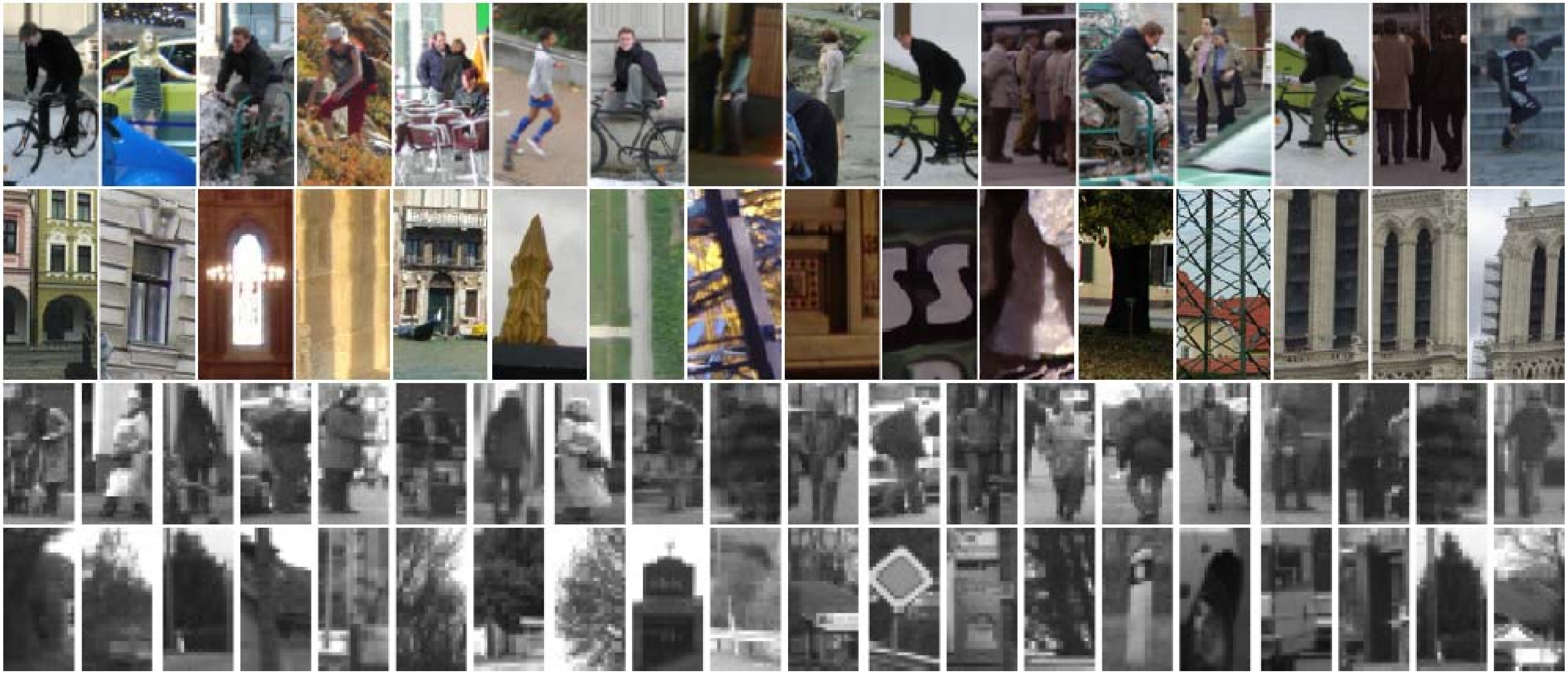
Results - INRIA Pedestrian Dataset

Classification Method	Detection Rate (2 FPPI)	Speedup
Linear SVM	43.12 %	-
IKSVM (binary search)	86.59 %	**473**×
IKSVM (piecewise linear)	86.59 %	**2594**×
IKSVM (piecewise constant)	86.59 %	**3098**×
Dalal & Triggs [8]	79.63 %	-
Dalal & Triggs* [8]	82.51 %	-

Fig. 1. Detection Rate at 2 FPPI on INRIA Person dataset for various methods. We use 300 bins to approximate both the piecewise linear and piecewise constant classifiers. The final classifier has 6416 support vectors and we get about **473**× speedup using the binary search, **2594**× using the piecewise linear approximation and **3098**× speedup using piecewise constant approximation, without any loss in accuracy over the exact method. The performance using linear SVM is on the same features is significantly worse. We also compare our method with the state of the art Dalal & Triggs detector which uses a linear SVM with a different set of features. All the detectors are run densely with a stride of 8px along the width and height and a scaleratio of $2^{1/8}$ (*scaleratio of $1.05 \sim 2^{1/16}$).

- Outperforms linear significantly using pHOG features.
- About 3-4x slower than linear SVM. Most time spent on computing features anyway.
- IKSVM on HOG beats linear on HOG (not shown in the table)

Errors



Results - DC Pedestrians/Caltech-101

Classification Method	Accuracy(%)	Speedup
Linear SVM	72.19 ± 4.40	-
IKSVM (binary search)	89.06 ± 1.42	**485**×
IKSVM (piecewise linear)	89.03 ± 1.39	**2253**×
IKSVM (piecewise constant)	88.83 ± 1.39	**3100**×

Fig. 3. Accuracy on Daimler Crysler Pedestrians dataset for various methods. We use 100 bins to approximate both the piecewise linear and piecewise constant classifiers. The final classifier has 5140 ± 392 support vectors and we get about **485**× speedup using the binary search, **2253**× using the piecewise linear approximation and **3100**× speedup using piecewise constant approximation, without any significant loss in accuracy over the exact method. The performance using linear SVM is on the same features is significantly worse.

	15 examples, 115.2 ± 15 SVs		30 examples, 185.0 ± 26 SVs	
Classification Method	Accuracy(%)	Speedup	Accuracy(%)	Speedup
Linear SVM	38.79 ± 0.94	-	44.33 ± 1.33	-
IKSVM (binary search)	50.15 ± 0.61	**11**×	56.49 ± 0.78	**17**×
IKSVM (piecewise linear)	50.10 ± 0.65	**37**×	56.59 ± 0.77	**62**×
IKSVM (piecewise contant)	49.83 ± 0.62	**45**×	56.11 ± 0.94	**76**×

Fig. 2. Classification Accuracy of various methods on Caltech-101 dataset. Accuracy is reported using 15(left) and 30(right) training examples per category and tested on 50 images averaged over 5 random splits. 60 bins are used to approximate the classifier for both the piecewise linear and piecewise constant approximation. The accuracy of the approximations is similar (within the variance) of the exact method (binary search IKSVM) while being significantly better than a linear SVM.

Results - Single Scale UIUC Cars

Classification Method	Performance(%)	Speedup
Linear SVM	89.8	-
IKSVM (binary search)	98.5	23×
IKSVM (piecewise linear)	98.5	65×
IKSVM (piecewise constant)	98.5	83×
Agarwal & Roth [1]	79.0	-
Garg et al. [12]	88.0	-
Fregus et al. [11]	88.5	-
ISM [19]	97.5	-
Mutch & Lowe [20]	99.6	-
Lampert et al. [17]	98.5	-

Fig. 4. Performance at Equal Error Rate on UIUC Single Scale Cars for various methods. We use 50 bins to approximate both the piecewise linear and piecewise constant classifiers. The IKSVM classifier has 212 support vectors and we get an average of baout 23× speedup using the binary search, 65× using the piecewise linear approximation and 83× speedup using piecewise constant approximation, without any loss in accuracy over the exact method. The performance using linear SVM is significantly worse.

Results – ETHZ Dataset

Dataset: Ferrari et al., ECCV 2006

255 images, over 5 classes

training = half of positive images for a class

+ same number from the other classes (1/4 from each)

testing = **all** other images

large scale changes; extensive clutter

Results – ETHZ Dataset

- Beats many current techniques without any changes to our features/classification framework.
- Shape is an important cue (use Pb instead of OE)
- Recall at 0.3 False Positive per Image (shown below)

Method	Applelogo	Bottle	Giraffe	Mug	Swan	Avg
PAS*	65.0	89.3	72.3	80.6	64.7	76.7
Our	86.1	81.0	62.1	78.0	100	81.4

*Ferarri *et.al*, IEEE PAMI - 08

Other kernels allow similar trick

Decision function is $\text{sign}\,(h(x))$ where:

IKSVM

$$\begin{aligned} h(x) &= \sum_{j=1}^{\#sv} \alpha^j \left(\sum_{i=1}^{\#dim} \min(x_i, x_i^j) \right) + b \\ &= \sum_{i=1}^{\#dim} \left(\sum_{j=1}^{\#sv} \alpha^j \min(x_i, x_i^j) \right) + b \\ &= \sum_{i=1}^{\#dim} h_i(x_i) \end{aligned}$$

h_i are piece-wise linear, uniformly spaced piece-wise linear approx. is fast.

$-\chi^2$ SVM

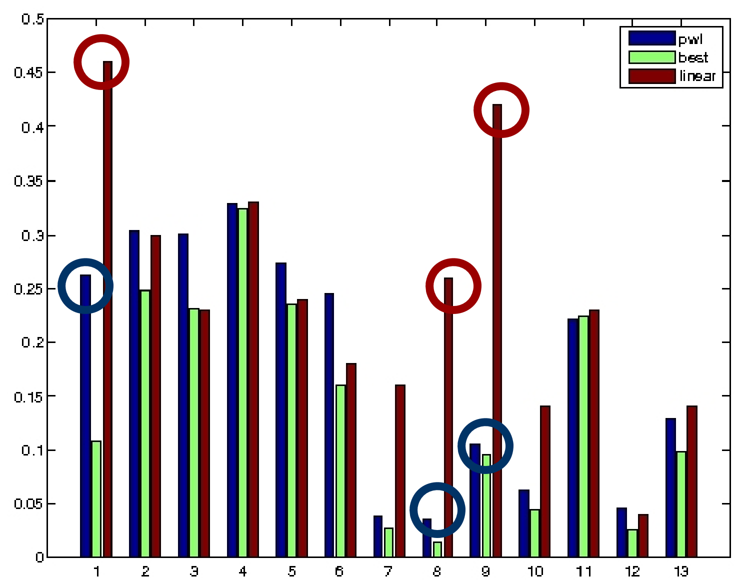
$$\begin{aligned} h(x) &= \sum_{j=1}^{\#sv} \alpha^j \left(\sum_{i=1}^{\#dim} \frac{(x_i - x_i^j)^2}{\frac{1}{2}(x_i + x_i^j)} \right) + b \\ &= \sum_{i=1}^{\#dim} \left(\sum_{j=1}^{\#sv} \alpha^j \frac{(x_i - x_i^j)^2}{\frac{1}{2}(x_i + x_i^j)} \right) + b \\ &= \sum_{i=1}^{\#dim} h_i(x_i) \end{aligned}$$

h_i not piece-wise linear, but we can still use an approximation for fast evaluation.

Results outside computer vision

Accuracy of IK vs Linear on Text classification

Accuracy Values					
Classification Method	R8	R52	20Ng	Cade12	WebKb
SVM (Linear Kernel)	0.9666(1)	0.9322(1)	0.8155(0.04)	0.5650(0.05)	0.8796(0.04)
SVM (Intersection Kernel)	0.9693(1)	0.9326(0.8)	0.8115(0.05)	0.5777(0.10)	0.9105(0.04)



Error rate of directly
+ *iksvm (blue)*
+ *best kernel (green)*
+ *linear (red)*
on SVM benchmark datasets

Conclusions

- Exact evaluation in O(log #SV), approx in O(1) (same as linear!)
- Runtime for approximate is O(1) (same as linear!)
- Significantly outperforms linear on variety of vision/non vision datasets
- Technique applies to any additive kernel (e.g. pyramid match kernel, spatial pyramid match kernel, –chi^2, etc)
- Represents some of the best Caltech 256, Pascal VOC 2007 methods.
- Training time is much worse compared to linear (Dual coordinate descent, PEGASOS)
- Inside news! Train Additive Kernel SVMs quickly using online stochastic gradient descent.
- Trains IKSVM based INRIA pedestrian detector ~50K feats of 4K dim in 100s. (compared to 3-4hours using LibSVM).

Discriminatively Trained Mixtures of Deformable Part Models

Pedro Felzenszwalb and Ross Girshick
University of Chicago

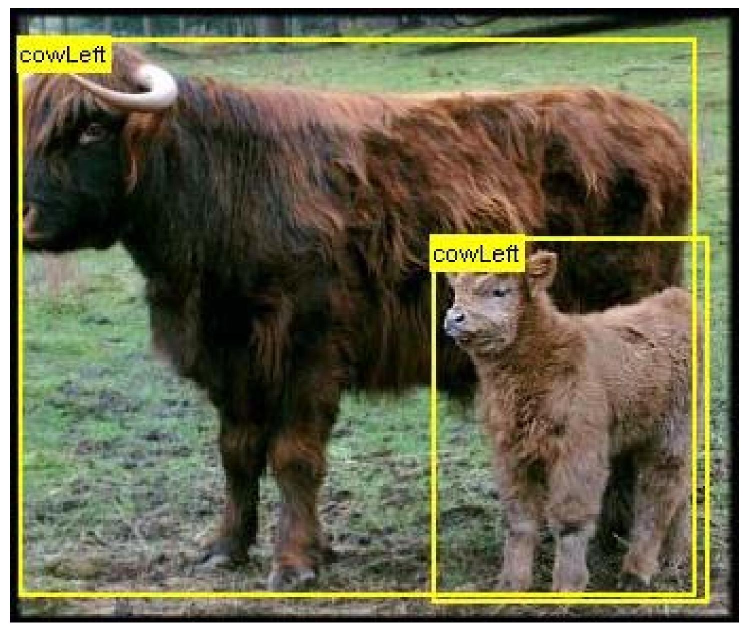
David McAllester
Toyota Technological Institute at Chicago

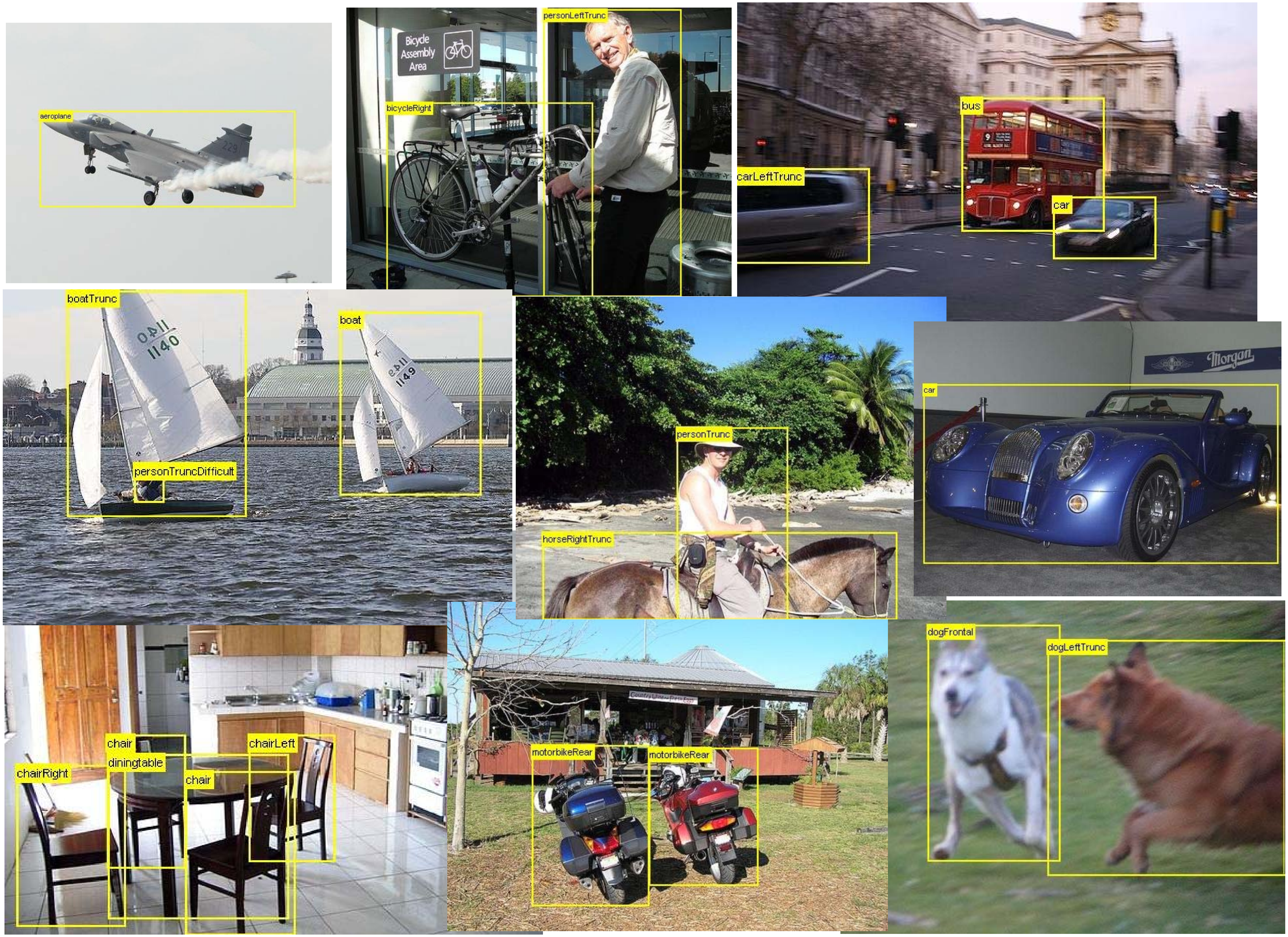
Deva Ramanan
UC Irvine

http://www.cs.uchicago.edu/~pff/latent

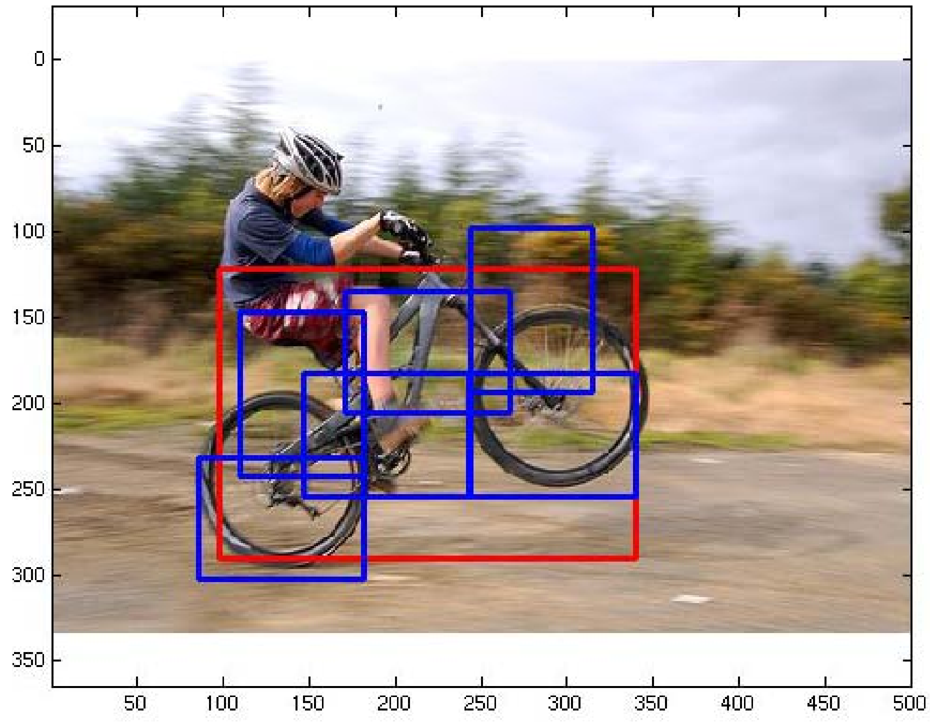
PASCAL Challenge

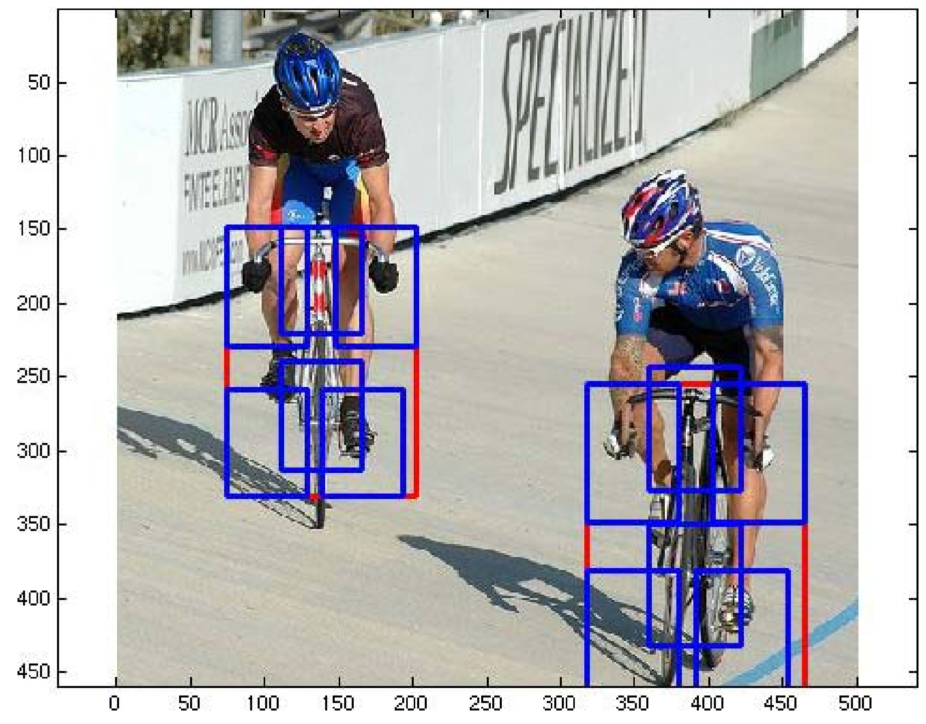
- ~10,000 images, with ~25,000 target objects.
  - Objects from 20 categories (person, car, bicycle, cow, table...).
  - Objects are annotated with labeled bounding boxes.




aeroplane
Bicycle Assembly Area
bicycleRight
personLeftTrunc
bus
carLeftTrunc
car
boatTrunc
1140
personTruncDifficult
boat
1149
personTrunc
horseRightTrunc
Morgan
car
chair
diningtable
chairLeft
chairRight
chair
motorbikeRear
motorbikeRear
dogFrontal
dogLeftTrunc

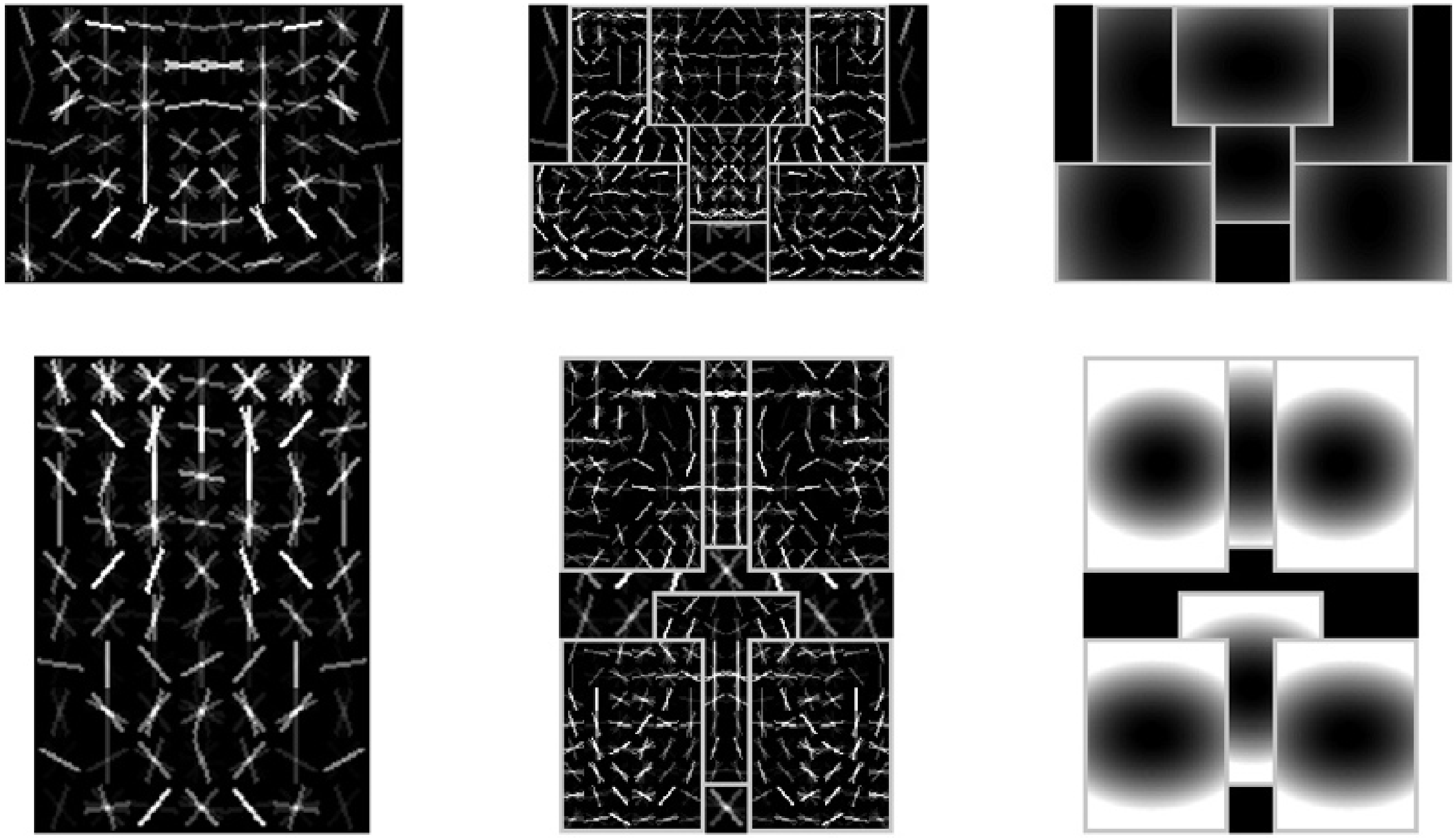
Model Overview





- Mixture of deformable part models (pictorial structures)
- Each component has global template + deformable parts
- Fully trained from bounding boxes alone

2 component bicycle model

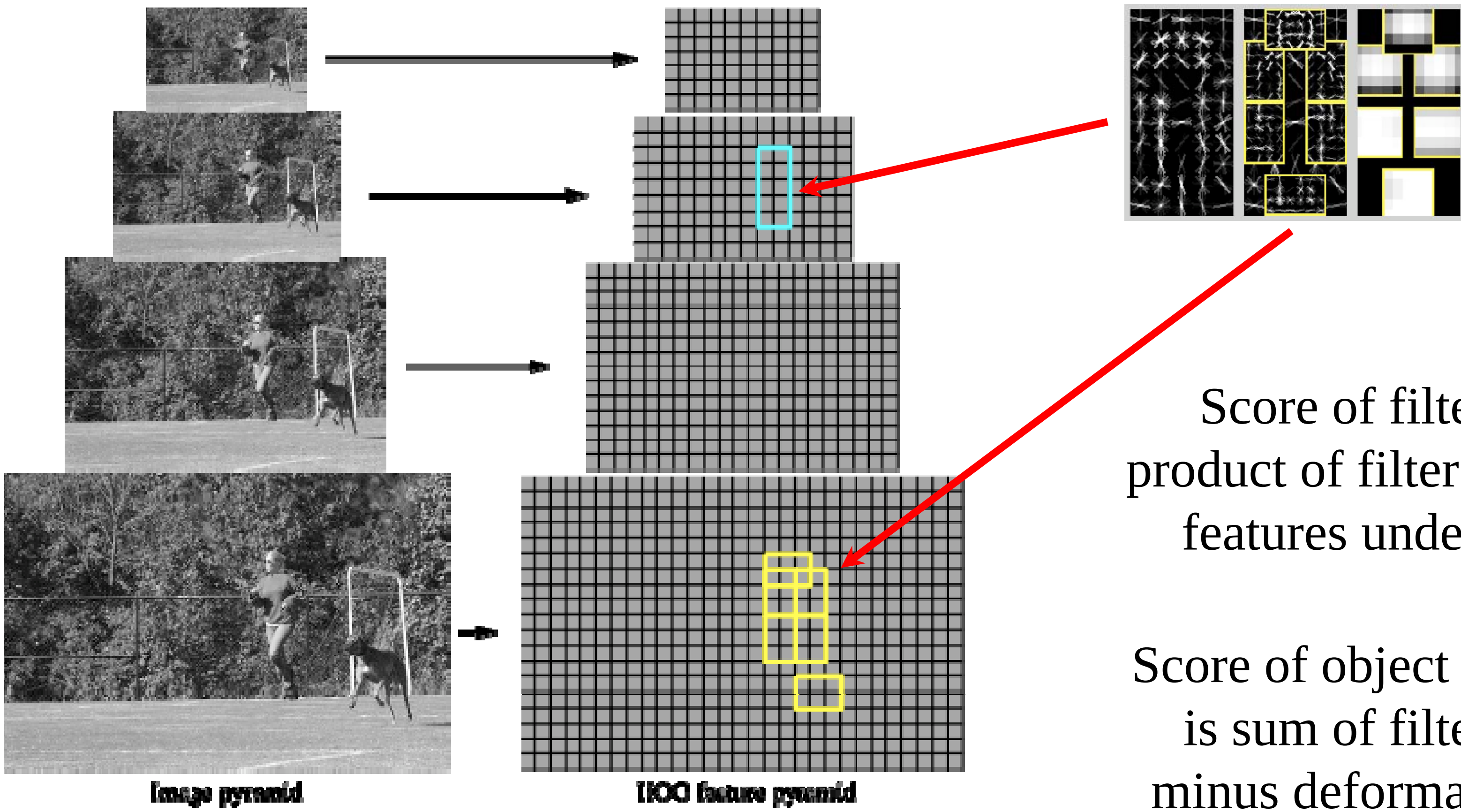


root filters
coarse resolution

part filters
finer resolution

deformation
models

Object Hypothesis



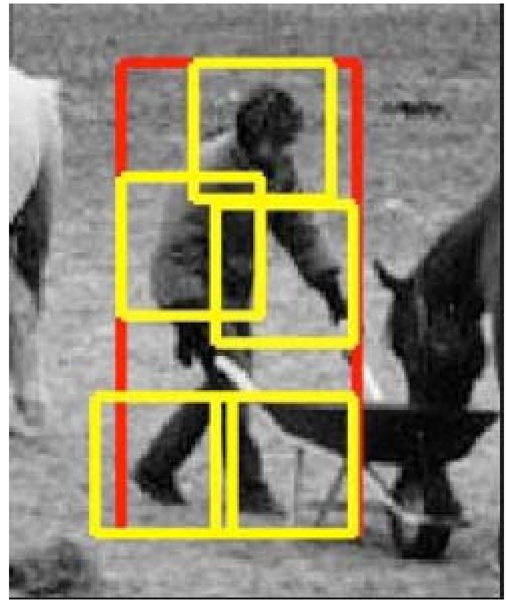
Score of filter is dot product of filter with HOG features underneath it

Score of object hypothesis is sum of filter scores minus deformation costs

Multiscale model captures features at two resolutions

Model

$$f_w(x) = w \cdot \Phi(x)$$



$$f_w(x) = \max_z w \cdot \Phi(x, z)$$

Z = vector of part offsets

$\Phi(x, z)$= vector of HOG features (from root filter & appropriate part sub-windows) and part offsets

Latent SVM

$$f_w(x) = \max_z w \cdot \Phi(x, z)$$

Linear in w if z is fixed

$$\text{Training data: } (x_1, y_1), \ldots, (x_n, y_n) \text{ with } y_i \in \{-1, 1\}$$

$$\text{Learning: find } w \text{ such that } y_i f_w(x_i) > 0$$

$$w^* = \operatorname*{argmin}_w \lambda ||w||^2 + \sum_{i=1}^{n} \max(0, 1 - y_i f_w(x_i))$$

Regularization

Hinge loss

Latent SVM training

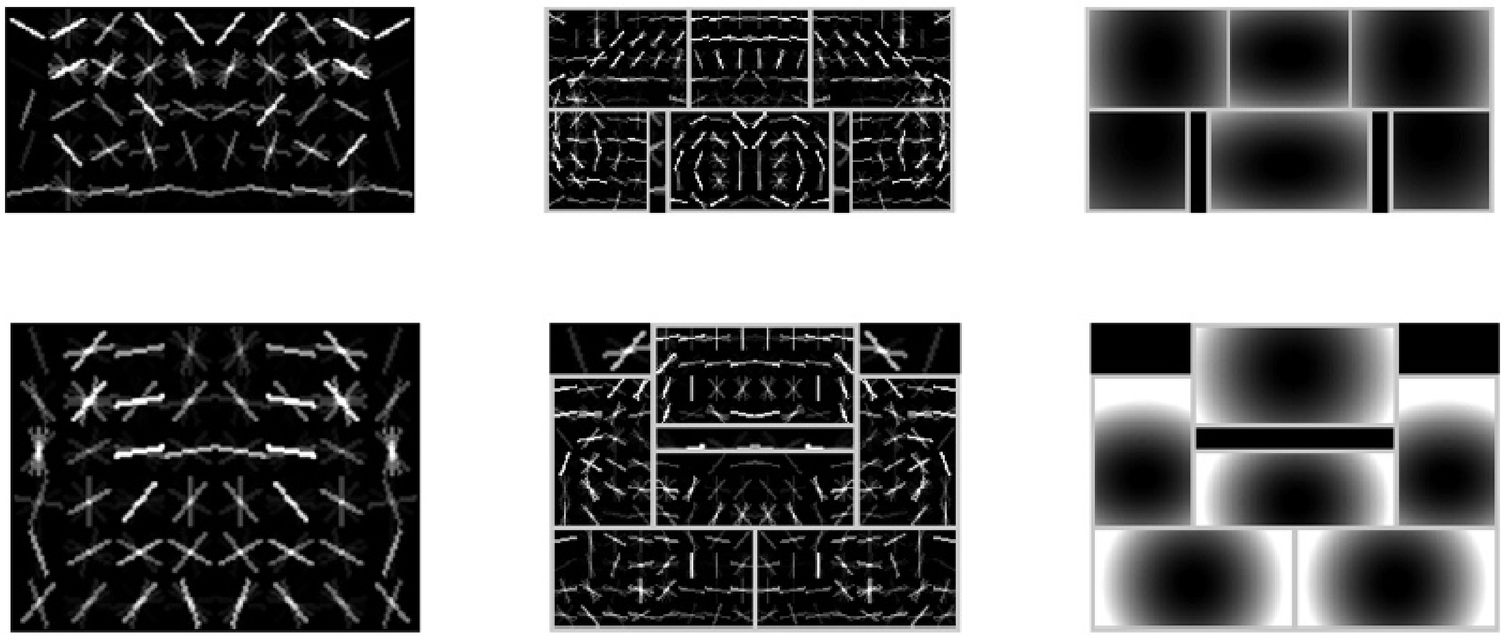
$$w^* = \underset{w}{\operatorname{argmin}}\, \lambda||w||^2 + \sum_{i=1}^{n} \max(0, 1 - y_i f_w(x_i))$$

- Non-convex optimization
- Huge number of negative examples
- Convex if we fix z for positive examples
- Optimization:
  - Initialize w and iterate:
    - Pick best z for each positive example
    - Optimize w via gradient descent with data mining

Initializing *w*

- For k component mixture model:
- Split examples into k sets based on bounding box aspect ratio
- Learn k root filters using standard SVM
  - Training data: warped positive examples and random windows from negative images (Dalal & Triggs)
- Initialize parts by selecting patches from root filters
  - Subwindows with strong coefficients
  - Interpolate to get higher resolution filters
  - Initialize spatial model using fixed spring constants

Car model

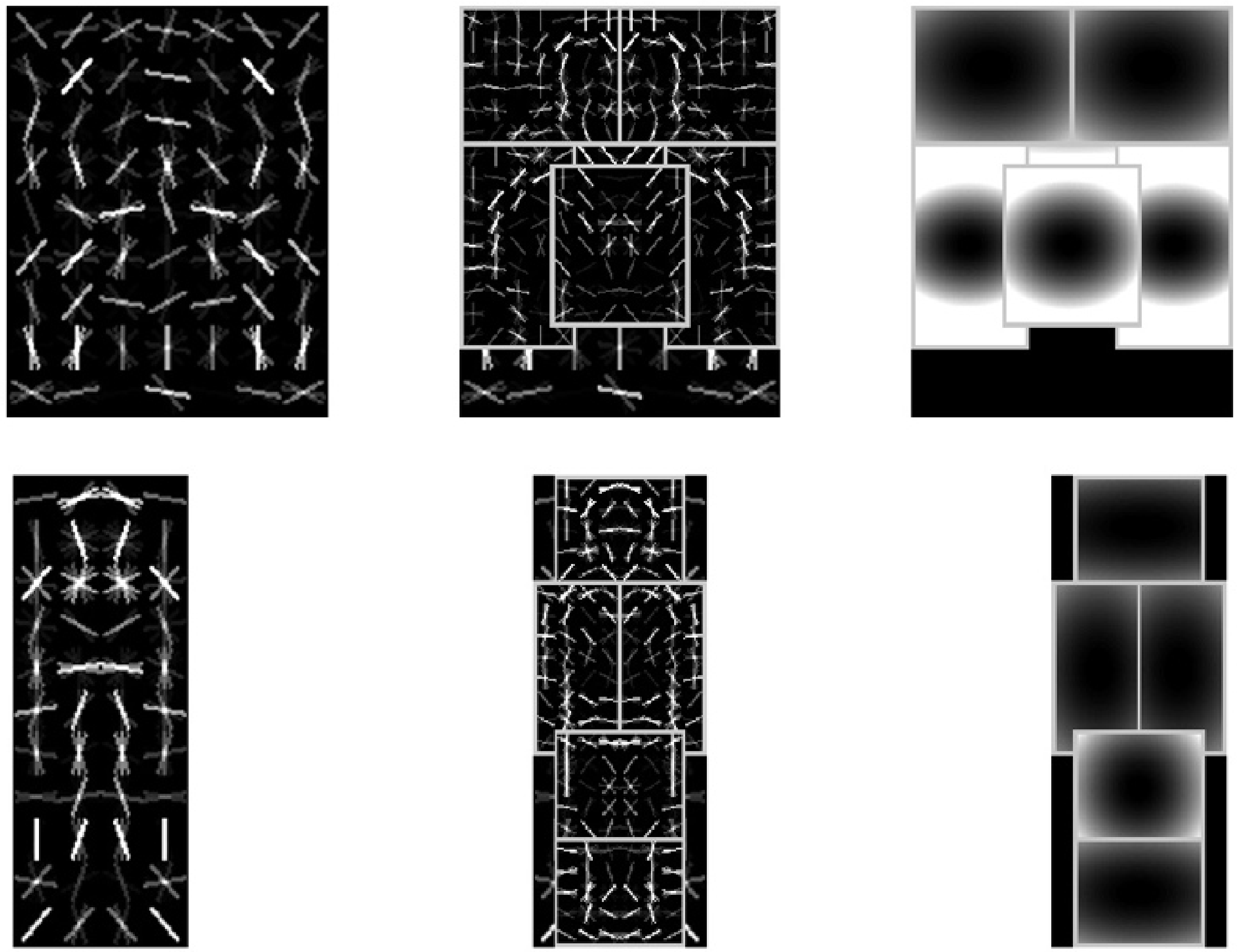


root filters
coarse resolution

part filters
finer resolution

deformation
models

Person model

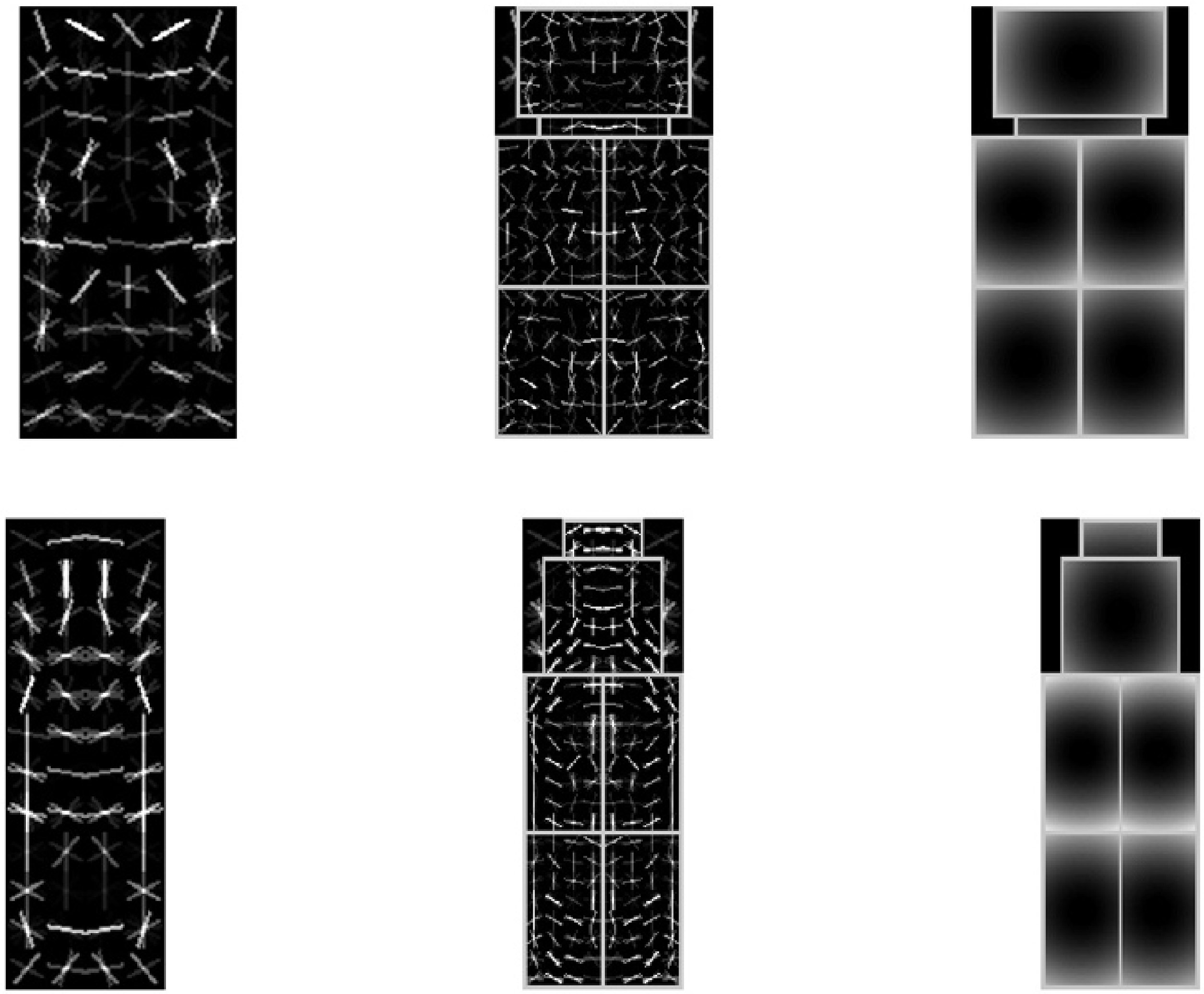


root filters
coarse resolution

part filters
finer resolution

deformation
models

Bottle model

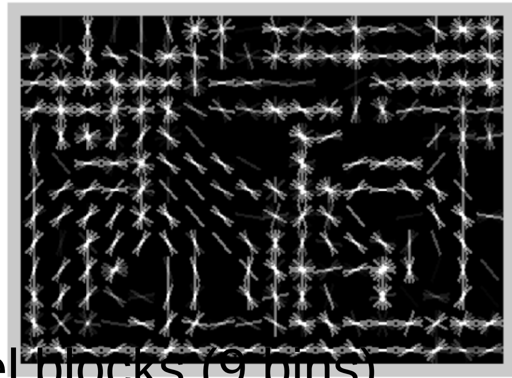


root filters
coarse resolution

part filters
finer resolution

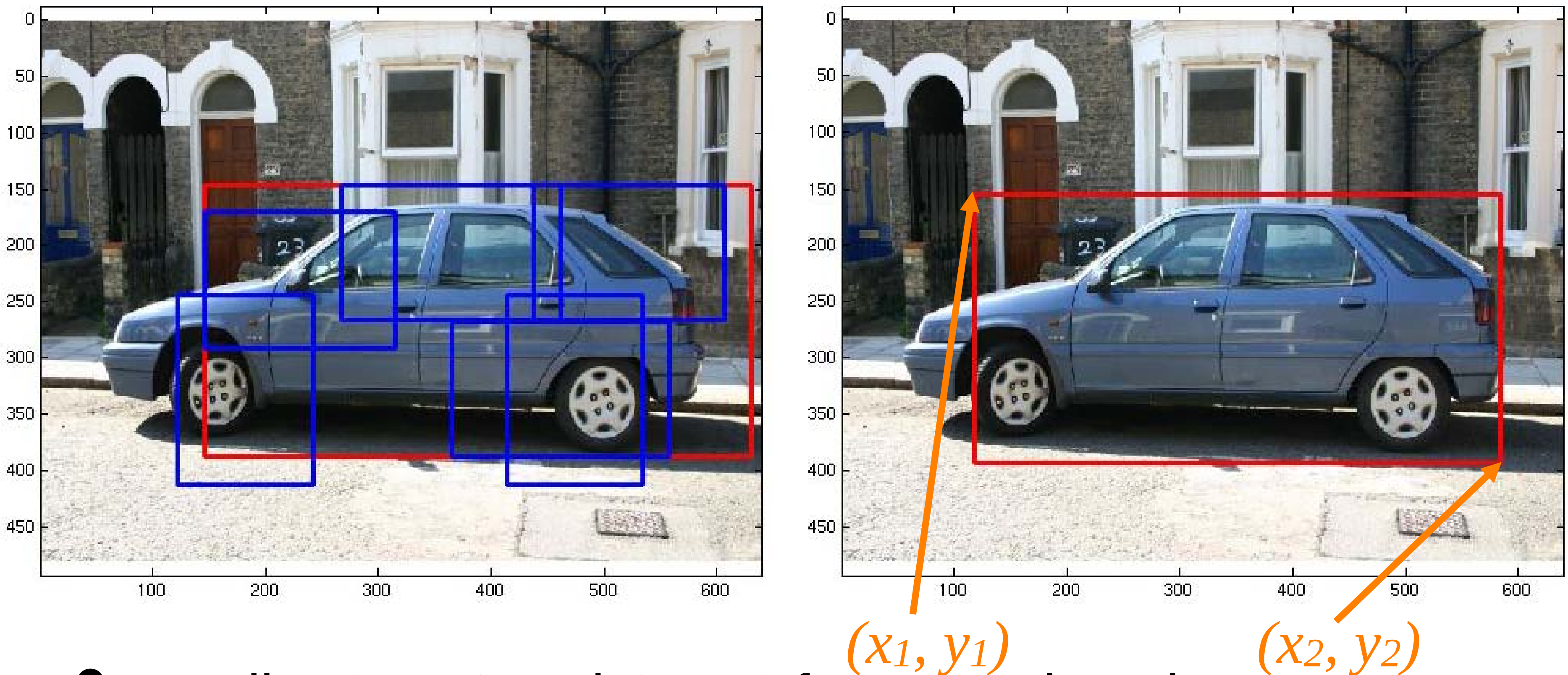
deformation
models

Histogram of Gradient (HOG) features



- Dalal & Triggs:
  - Histogram gradient orientations in 8x8 pixel blocks (9 bins)
  - Normalize with respect to 4 different neighborhoods and truncate
  - 9 orientations * 4 normalizations = 36 features per block
- PCA gives ~10 features that capture all information
  - Fewer parameters, speeds up convolution, but costly projection at runtime
- Analytic projection: spans PCA subspace and easy to compute
  - 9 orientations + 4 normalizations = 13 features
- We also use 2*9 contrast sensitive features for 31 features total

Bounding box prediction

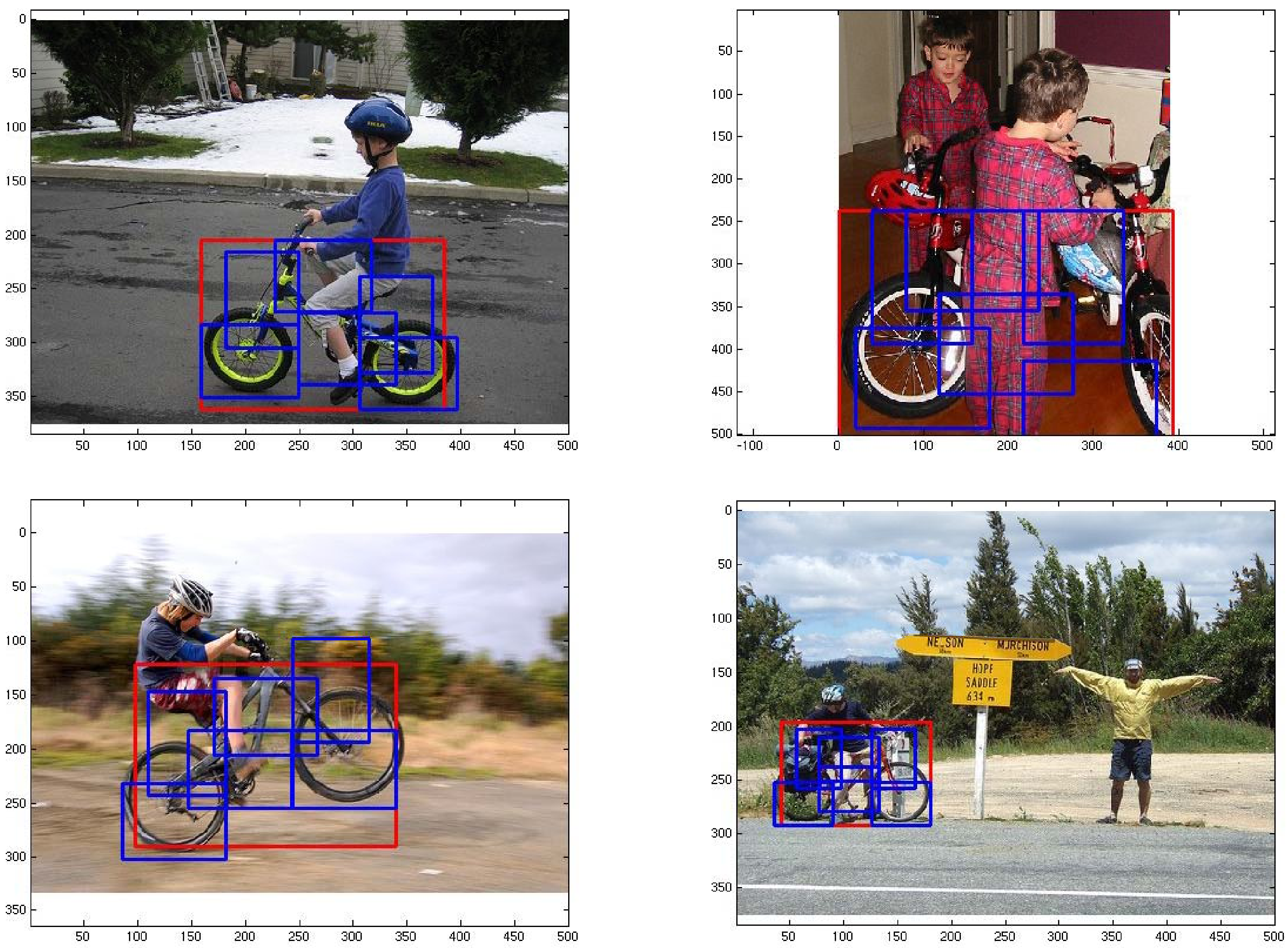


- predict (x_1, y_1) and (x_2, y_2) from part locations
- linear function trained using least-squares regression

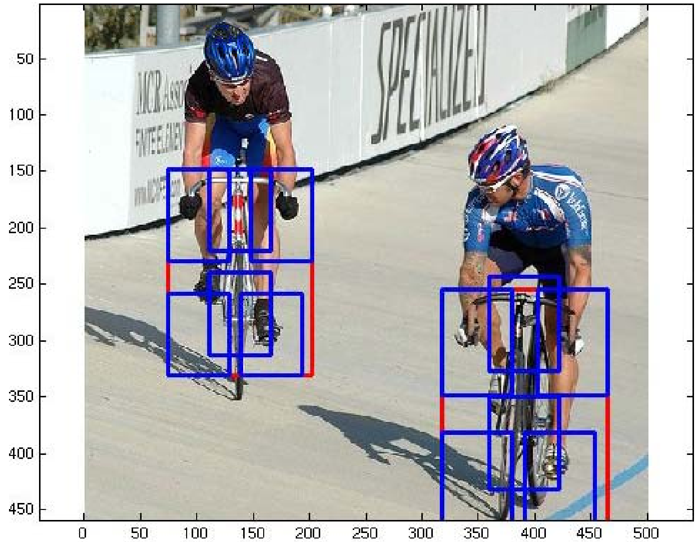
Context rescoring

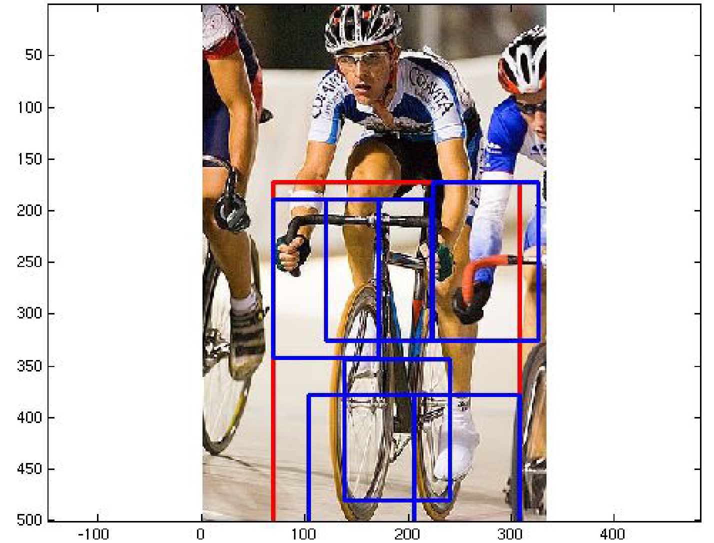
- Rescore a detection using "context" defined by all detections
- Let v_i be the max score of detector for class i in the image
- Let s be the score of a particular detection
- Let $(x_1,y_1), (x_2,y_2)$ be normalized bounding box coordinates
- $f = (s, x_1, y_1, x_2, y_2, v_1, v_2 \ldots , v_{20})$
- Train class specific classifier
  - f is positive example if true positive detection
  - f is negative example if false positive detection

Bicycle detection

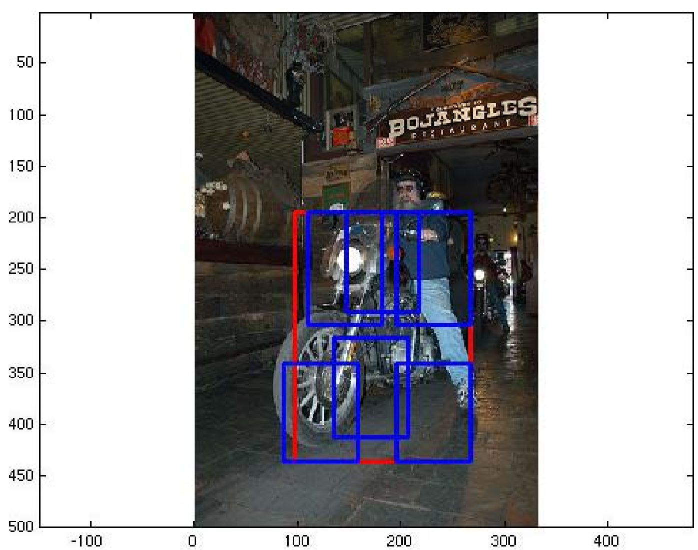


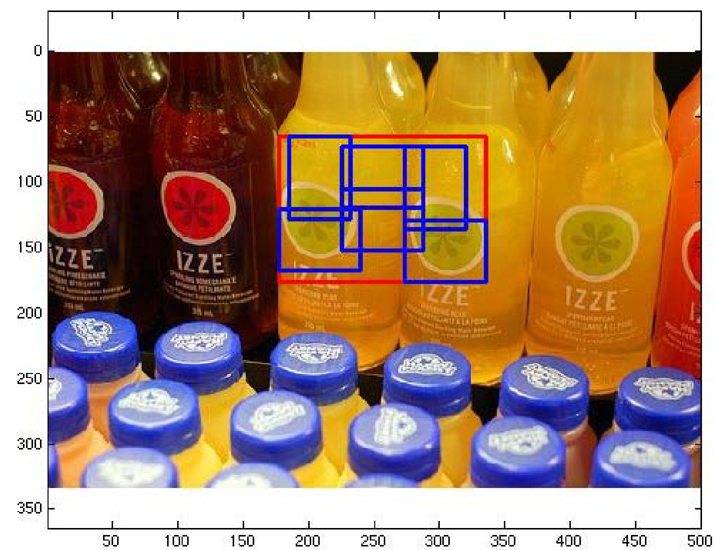
More bicycles



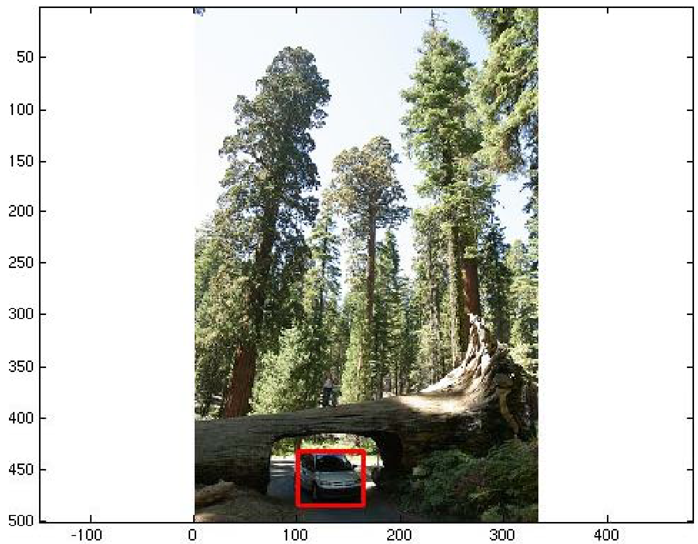


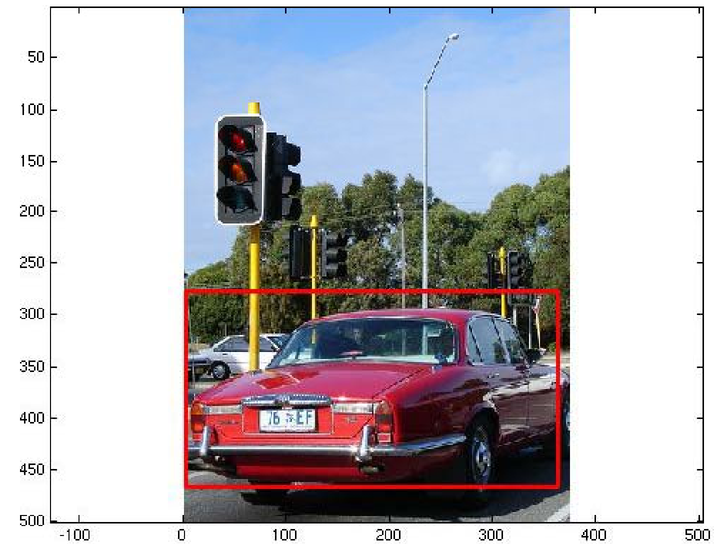
False positives

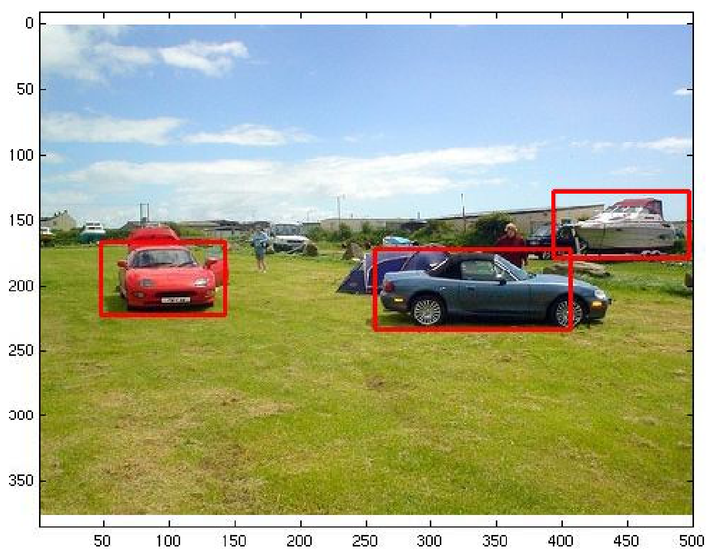


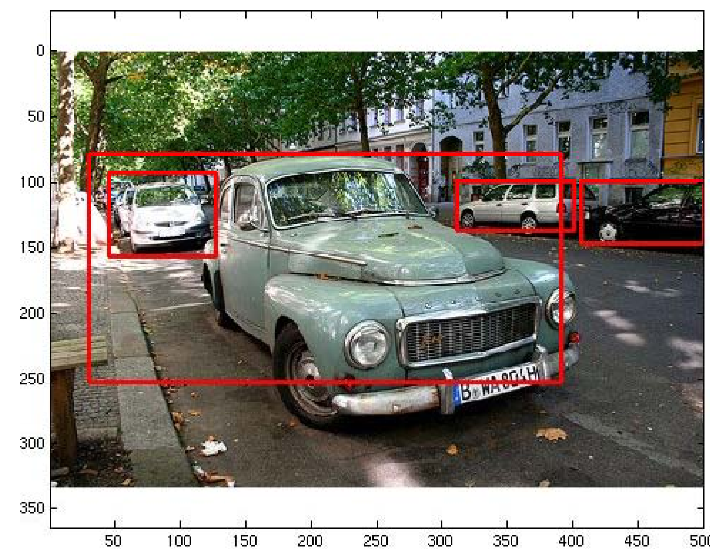


Car

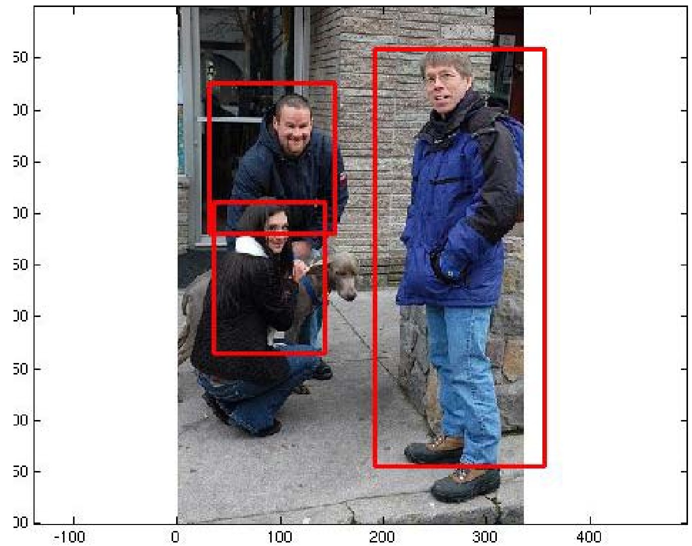

50
100
150
200
250
300
350
400
450
500
-100
0
100
200
300
400


50
100
150
200
250
300
350
400
450
500
-100
0
100
200
300
400
500


0
50
100
150
200
250
300
350
50
100
150
200
250
300
350
400
450
500


0
50
100
150
200
250
300
350
50
100
150
200
250
300
350
400
450
500

Person



Bottle

Horse

Code

Source code for the system and models trained on PASCAL 2006, 2007 and 2008 data are available here:

http://www.cs.uchicago.edu/~pff/latent

Today: Advanced kernels

- SVM-BOW
- Pyramid and spatial-pyramid match
- Fast IK
- Latent-part SVM models

Correspondence and Pyramid-based techniques

- C. Berg, T. L. Berg, and J. Malik, "Shape matching and object recognition using low distortion correspondences," in CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)
- K. Grauman and T. Darrell, "The pyramid match kernel: discriminative classification with sets of image features," ICCV, vol. 2, 2005, pp. 1458-1465 Vol. 2
- S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," CVPR, vol. 2, 2006, pp. 2169-2178
- S. Maji, A. C. Berg, and J. Malik, "Classification using intersection kernel support vector machines is efficient," in Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, 2008, pp. 1-8.
- K. Grauman and T. Darrell, "Approximate correspondences in high dimensions," in In NIPS, vol. 2006.
- A. Bosch, A. Zisserman, and X. Munoz, "Representing shape with a spatial pyramid kernel," in CIVR '07: Proceedings of the 6th ACM international conference on Image and video retrieval

Discriminative approaches (SVM, HCRF)

- Classic SVM on “bags of features”:

  C. Dance, J. Willamowski, L. Fan, C. Bray, and G. Csurka, "Visual categorization with bags of keypoints," in ECCV International Workshop on Statistical Learning in Computer Vision, 2004.

- ISM + SVM + Local Kernels:

  M. Fritz; B. Leibe; B. Caputo; B. Schiele: Integrating Representative and Discriminant Models for Object Category Detection, ICCV'05, Beijing, China, 2005

- Local SVM:

  H. Zhang, A. C. Berg, M. Maire, and J. Malik, "Svm-knn: Discriminative nearest neighbor classification for visual category recognition," in CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Washington, DC, USA: IEEE Computer Society, 2006, pp. 2126-2136.

- “Latent” SVM with deformable parts:

  P. Felzenszwalb, D. Mcallester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in IEEE International Conference on Computer Vision and Pattern Recognition (CVPR) Anchorage, Alaska, June 2008., June 2008.

- Hidden Conditional Random Fields:

  Y. Wang and G. Mori, “Learning a Discriminative Hidden Part Model for Human Action Recognition”, Advances in Neural Information Processing Systems (NIPS), 2008