Object Detection and Recognition

Jana Kosecka
George Mason University

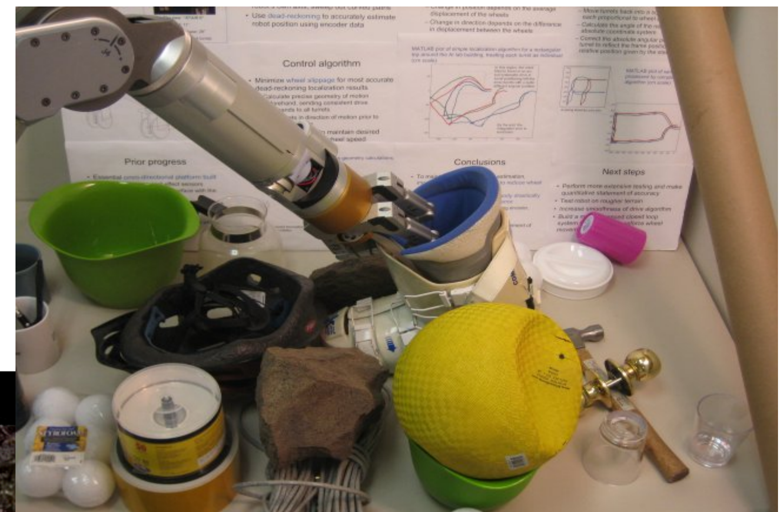
GEORGE MASON UNIVERSITY

Some slides courtesy of L. Lazebnik, K. Grauman, R. Fergus, Y. LeCunn, R. Girshik, Nister & Stewenius

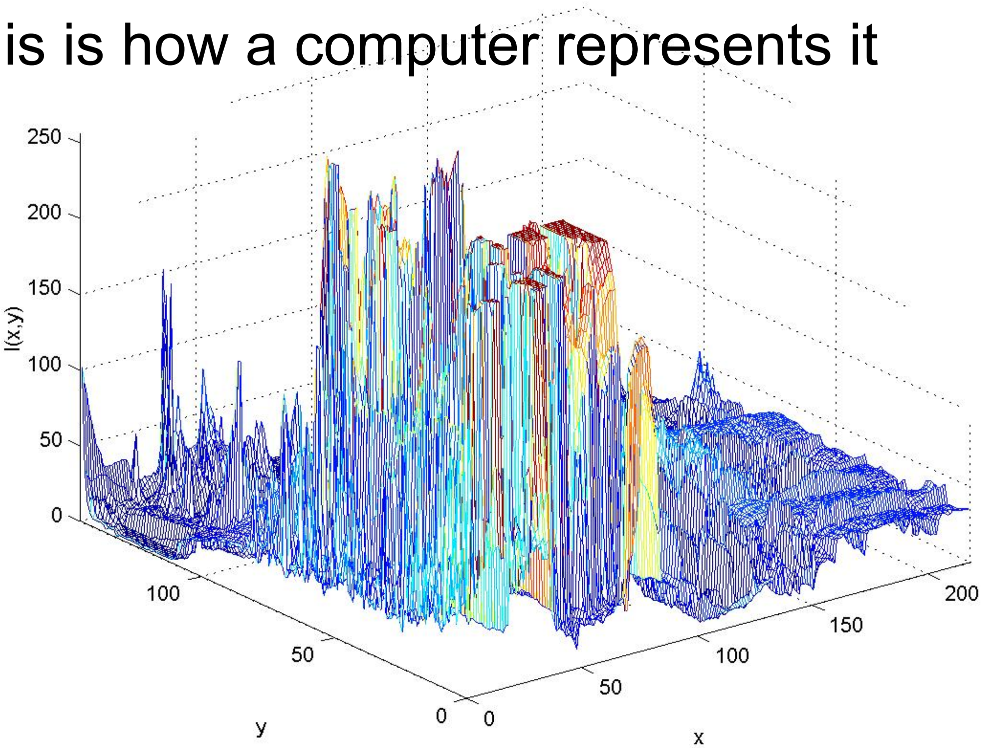
Topics

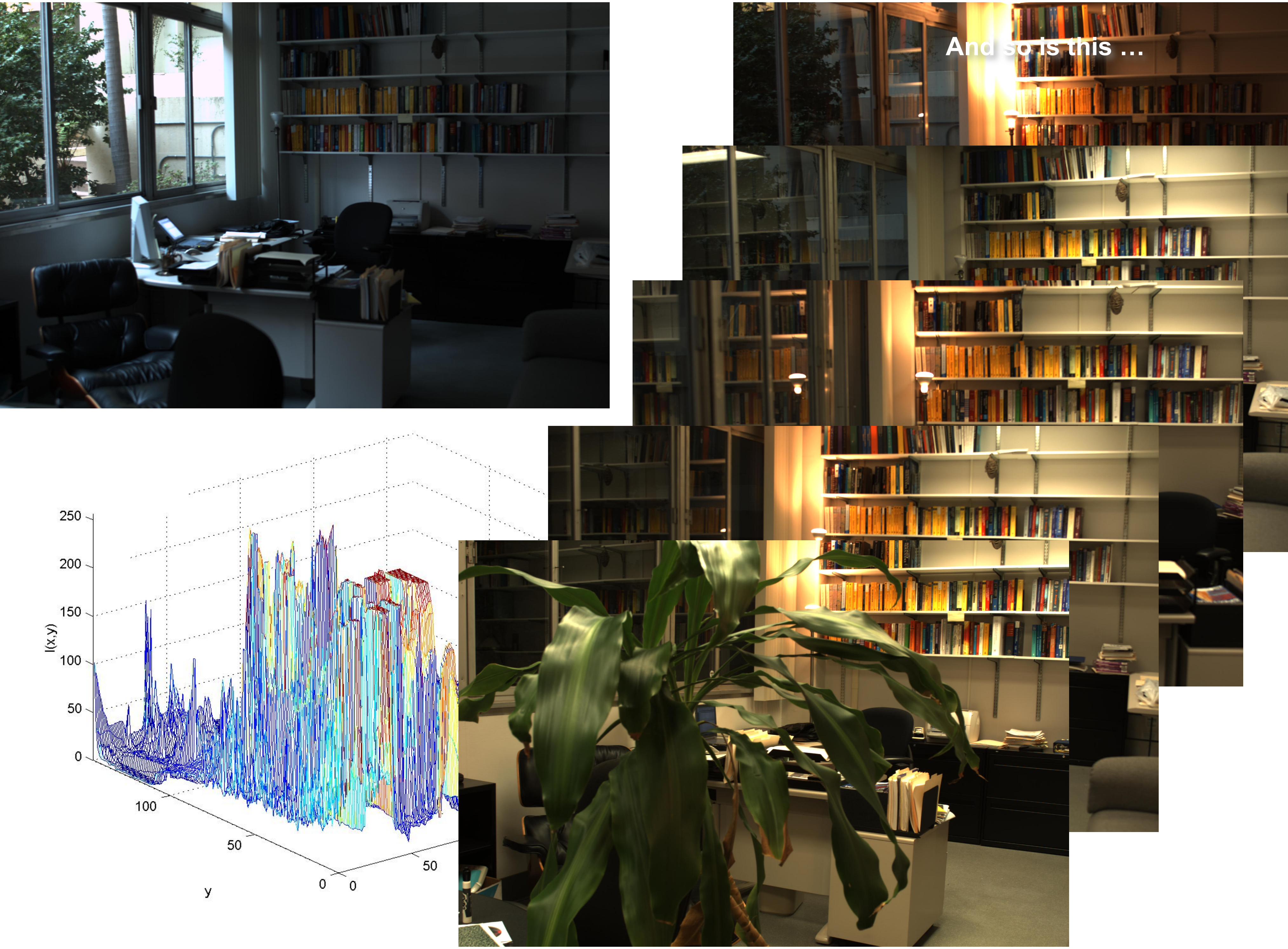
- Object Instance Detection/Recognition
- Object Category Detection/Recognition

Dalal and Triggs, CVPR 2005

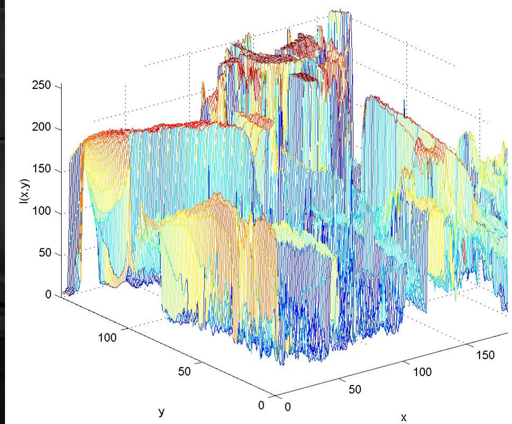
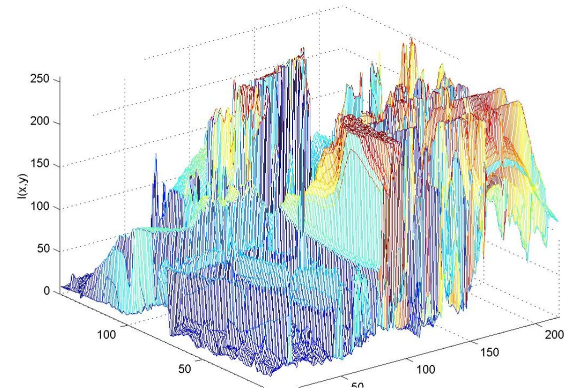
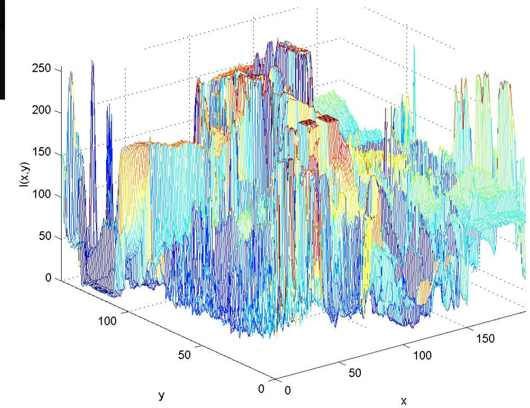
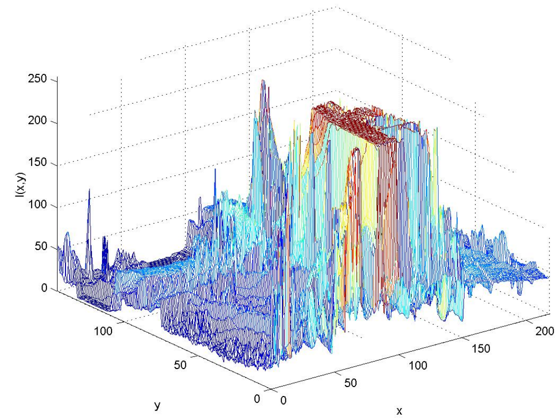
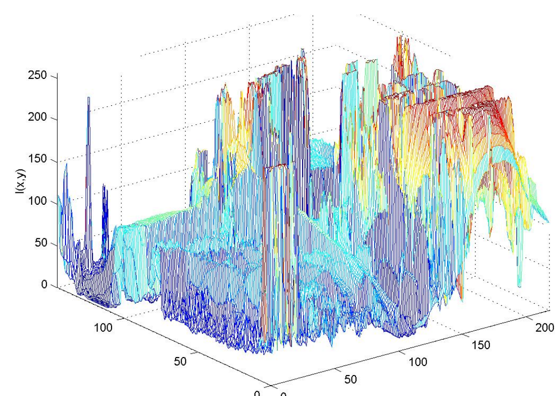


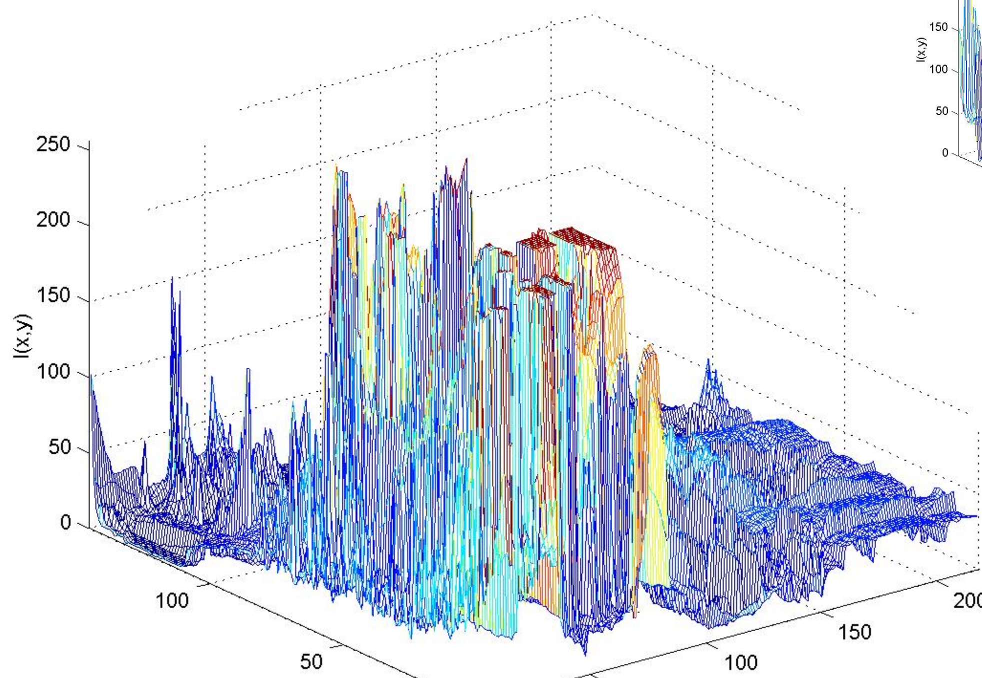
This is how a computer represents it




And so is this …
I(x,y)
250
200
150
100
50
0
y
100
50
0
0
50

And so are these!



We need to extract some “invariant”, i.e. what is common to all these images (they are all images of an office)

☹ truly invariant (photometric and geometric) representations do not exist

Challenges: viewpoint variation

Michelangelo 1475-1564

slide credit: Fei-Fei, Fergus & Torralba

Challenges: illumination

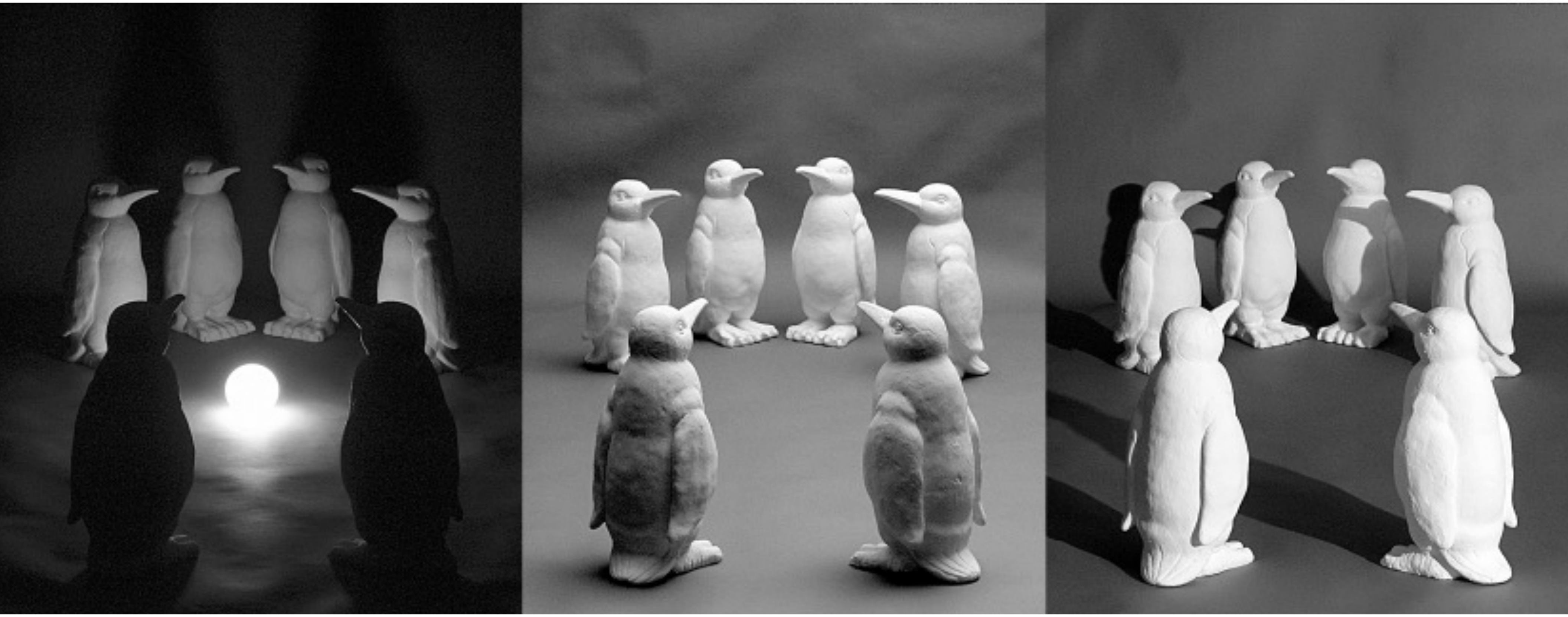
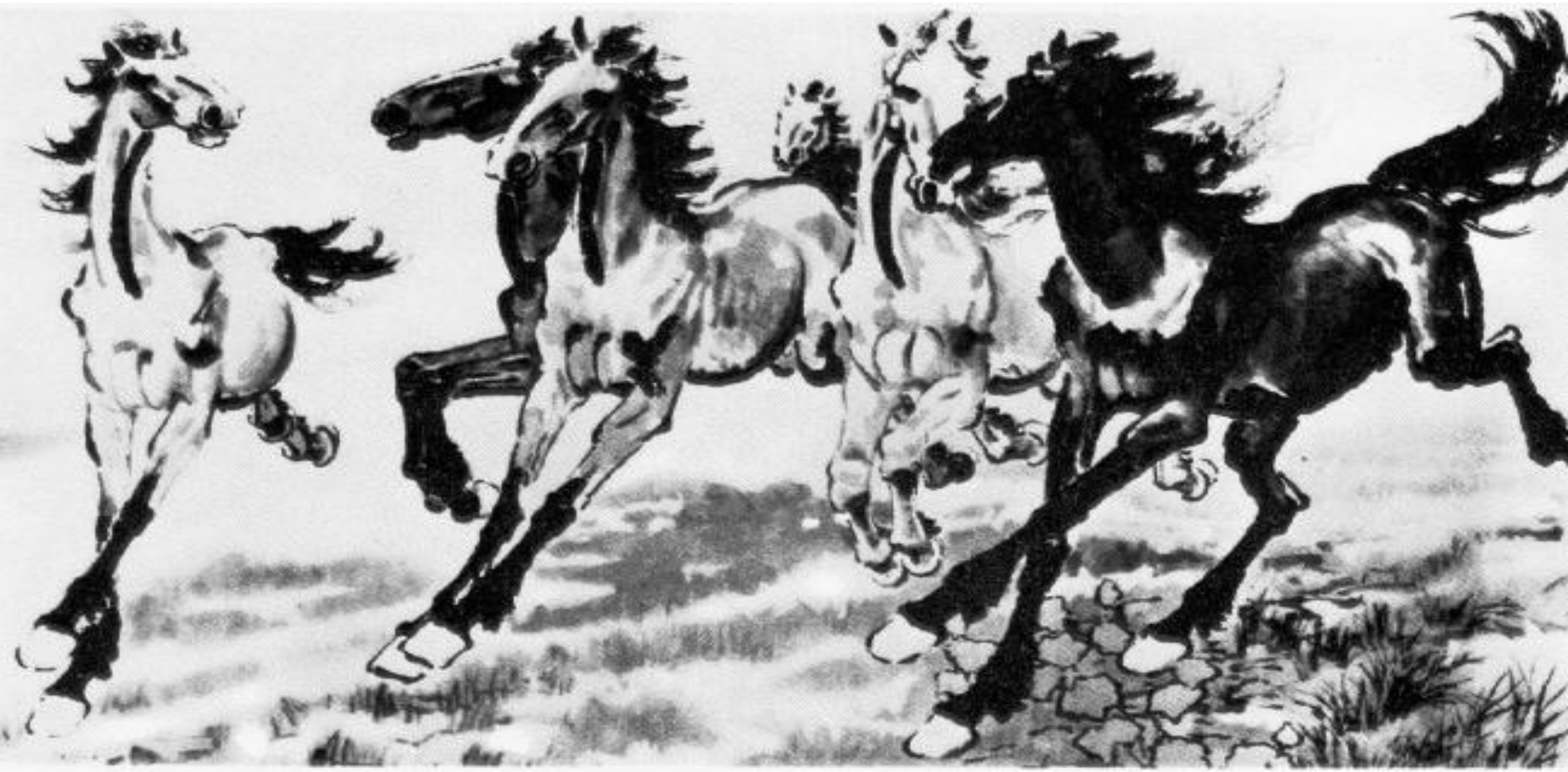


image credit: J. Koenderink

Challenges: scale

slide credit: Fei-Fei, Fergus & Torralba

Challenges: deformation



Xu, Beihong 1943

slide credit: Fei-Fei, Fergus & Torralba

Challenges: occlusion

Magritte, 1957

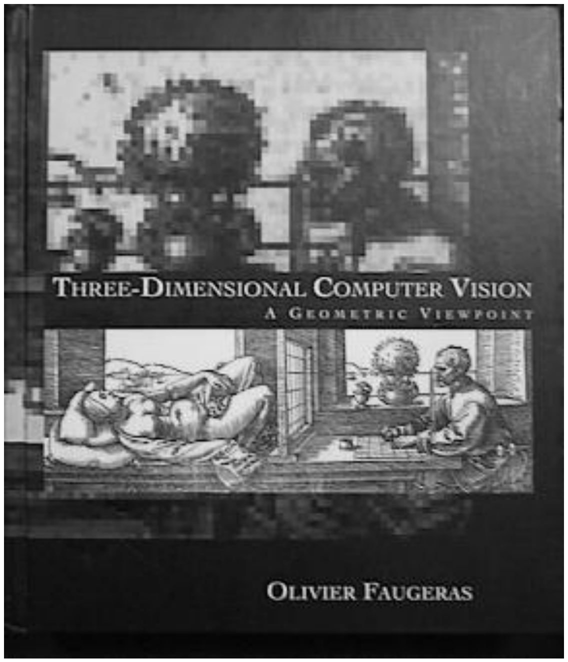
BUMMER! THIS IS IMPOSSIBLE!

  - THM: [Weiss, 1991]: There exists NO generic viewpoint invariant!
  - THM: [Chen et al., 2003]: There exists NO photometric invariant!!

- So, how do we (primates) solve the problem?

Improved Invariance Handling

Want to find



… in here

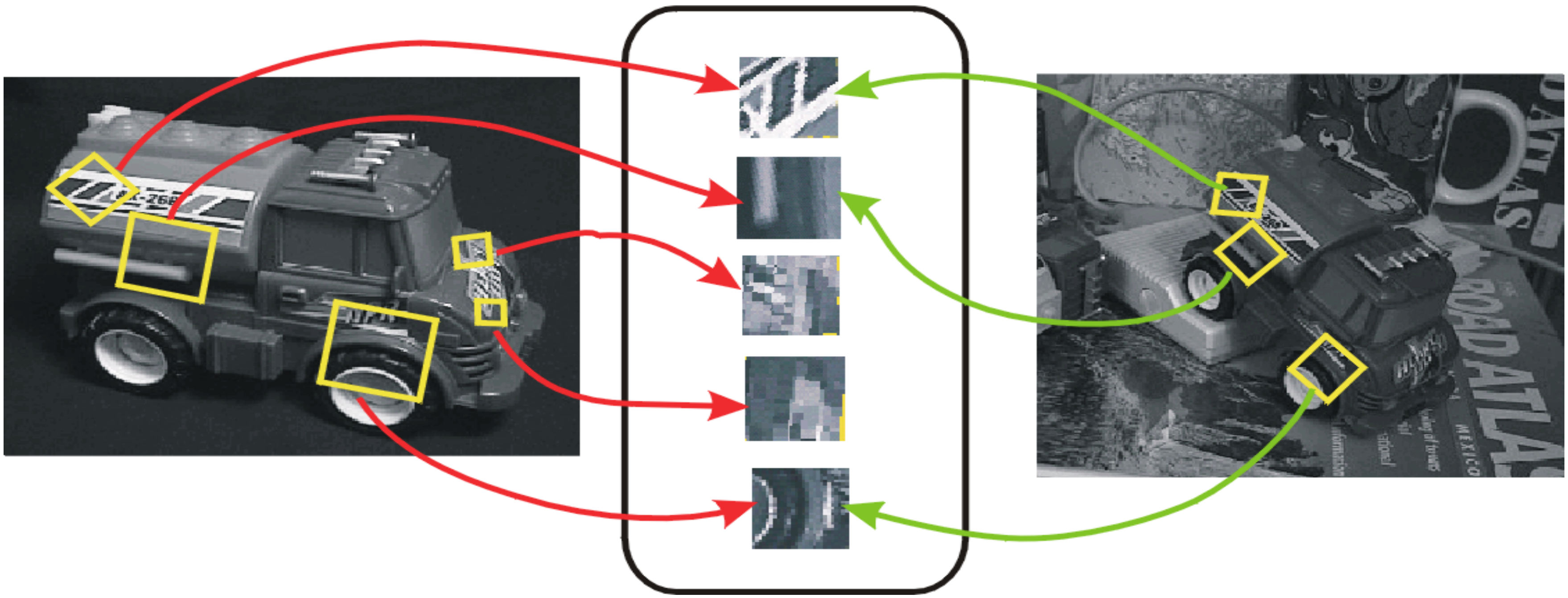
SIFT Features

- Invariances: Yes
  - Scaling Yes
  - Rotation Yes
  - Illumination Not really
  - Deformation
- Provides
  - Good localization Yes

Distinctive image features from scale-invariant keypoints. David G. Lowe, International Journal of Computer Vision, 60, 2 (2004), pp. 91-110.

Invariant Local Features

- Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters



SIFT Features

Advantages of invariant local features

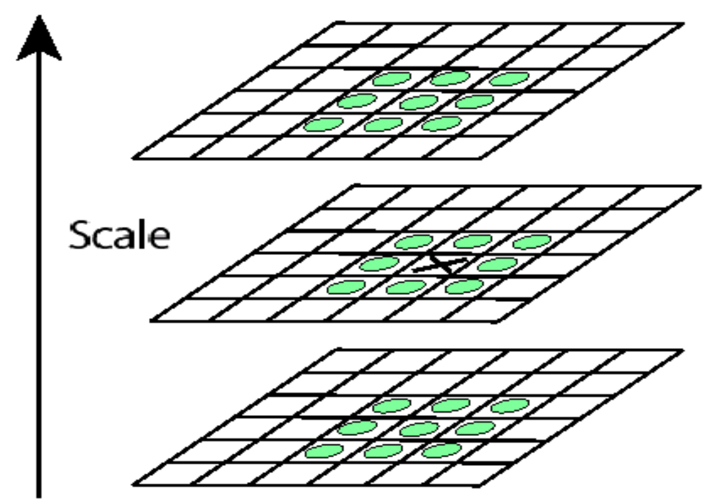
- **Locality:** features are local, so robust to occlusion and clutter (no prior segmentation)
- **Distinctiveness:** individual features can be matched to a large database of objects
- **Quantity:** many features can be generated for even small objects
- **Efficiency:** close to real-time performance
- **Extensibility:** can easily be extended to wide range of differing feature types, with each adding robustness

Key point localization

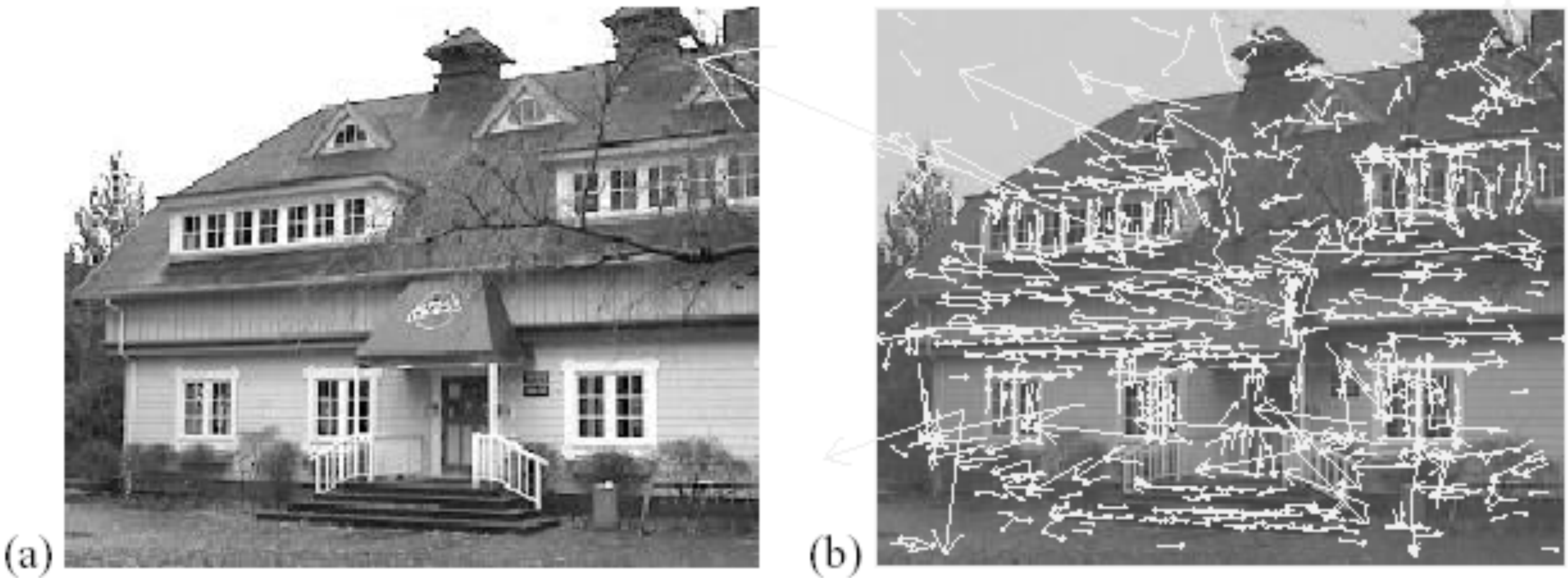
- In D. Lowe's paper image is decomposed to octaves (consecutively sub-sampled versions of the same image)
- Instead of convolving with large kernels

  within an octave kernels are kept the same
- Detect maxima and minima of difference-of-Gaussian in scale space
- Look for 3x3 neighbourhood in scale

  and space



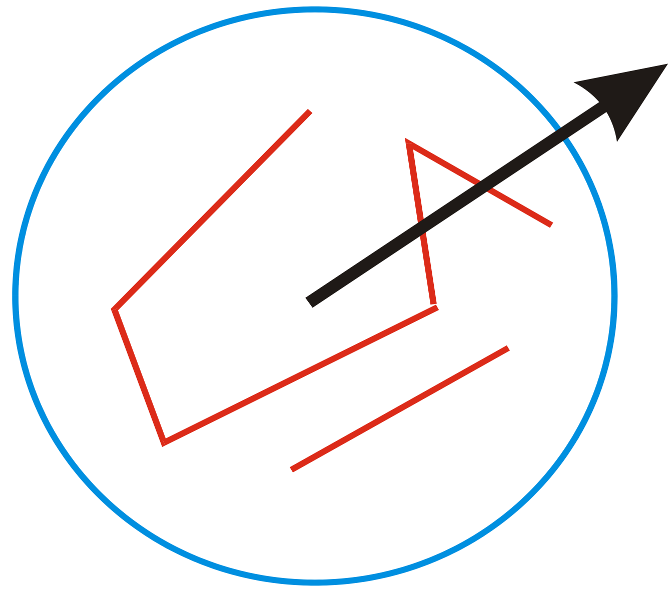
Example of keypoint detection

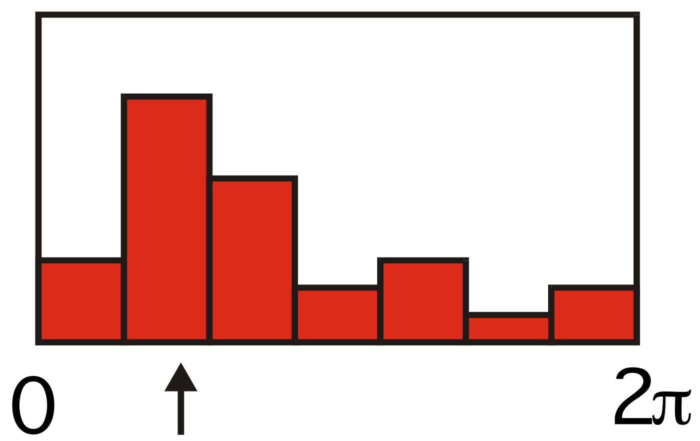


(a) 233x189 image
(b) 832 DOG extrema
(c) 729 above threshold

Select canonical orientation

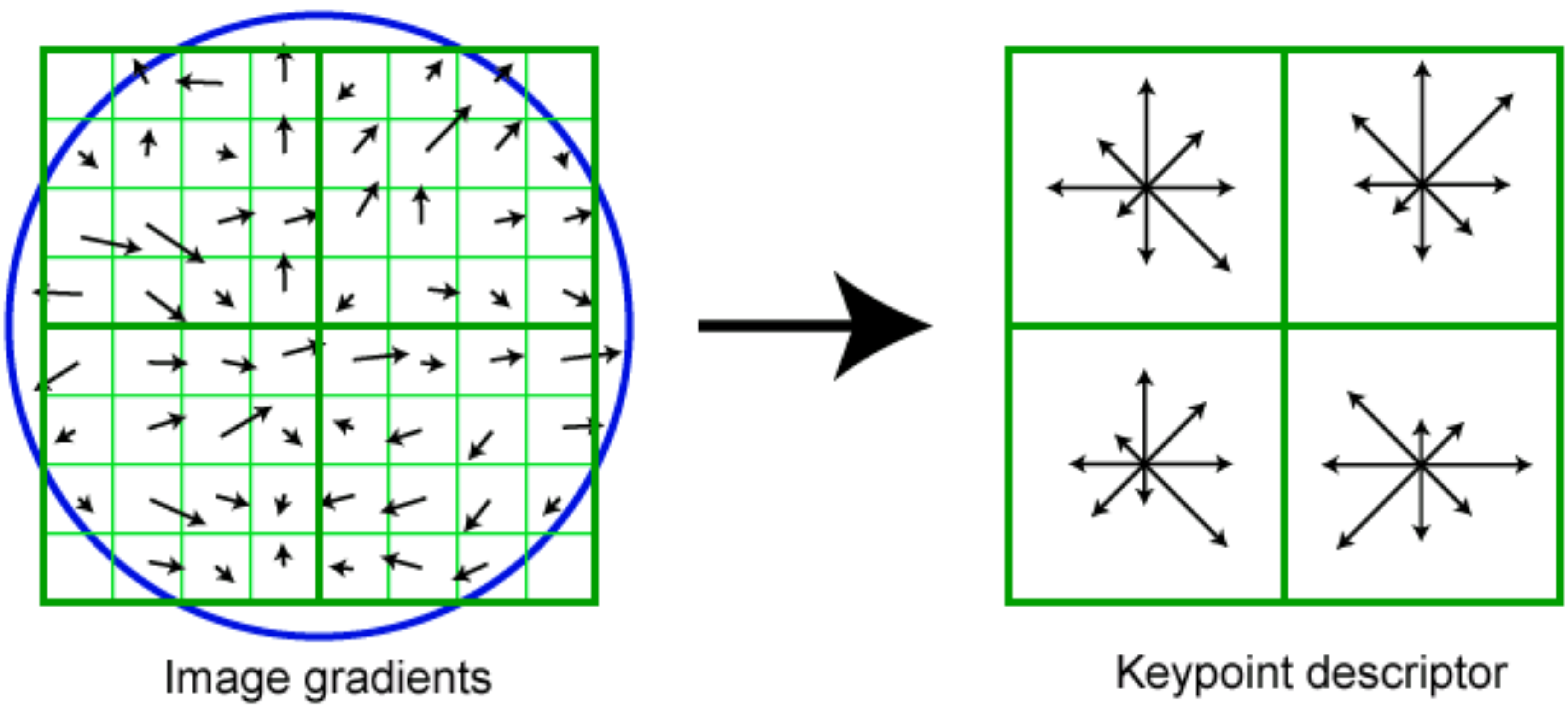
- Create histogram of local gradient directions computed at selected scale
- Assign canonical orientation at peak of smoothed histogram
- Each key specifies stable 2D coordinates (x, y, scale, orientation)





SIFT vector formation

- Thresholded image gradients are sampled over 16x16 array of locations in scale space
- Create array of orientation histograms
- 8 orientations x 4x4 histogram array = 128 dimensions



Nearest-neighbor matching to feature database

- Hypotheses are generated by **approximate nearest neighbor** matching of each feature to vectors in the database
  - SIFT use best-bin-first (Beis & Lowe, 97) modification to k-d tree algorithm
  - Use heap data structure to identify bins in order by their distance from query point

- **Result:** Can give speedup by factor of 1000 while finding nearest neighbor (of interest) 95% of the time

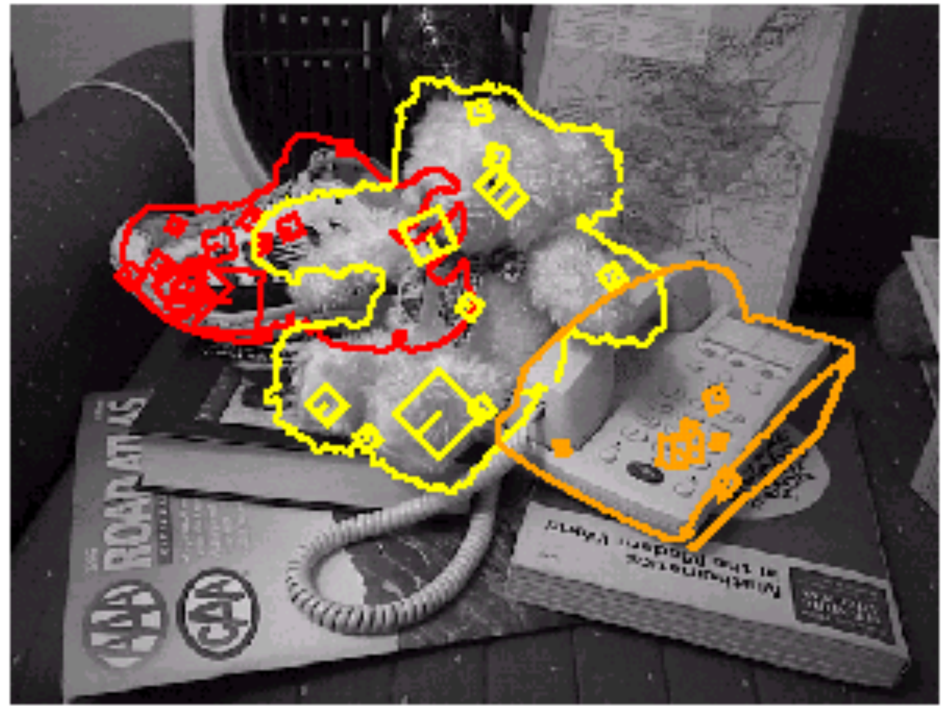
3D Object Recognition

- Extract outlines with background subtraction

3D Object Recognition

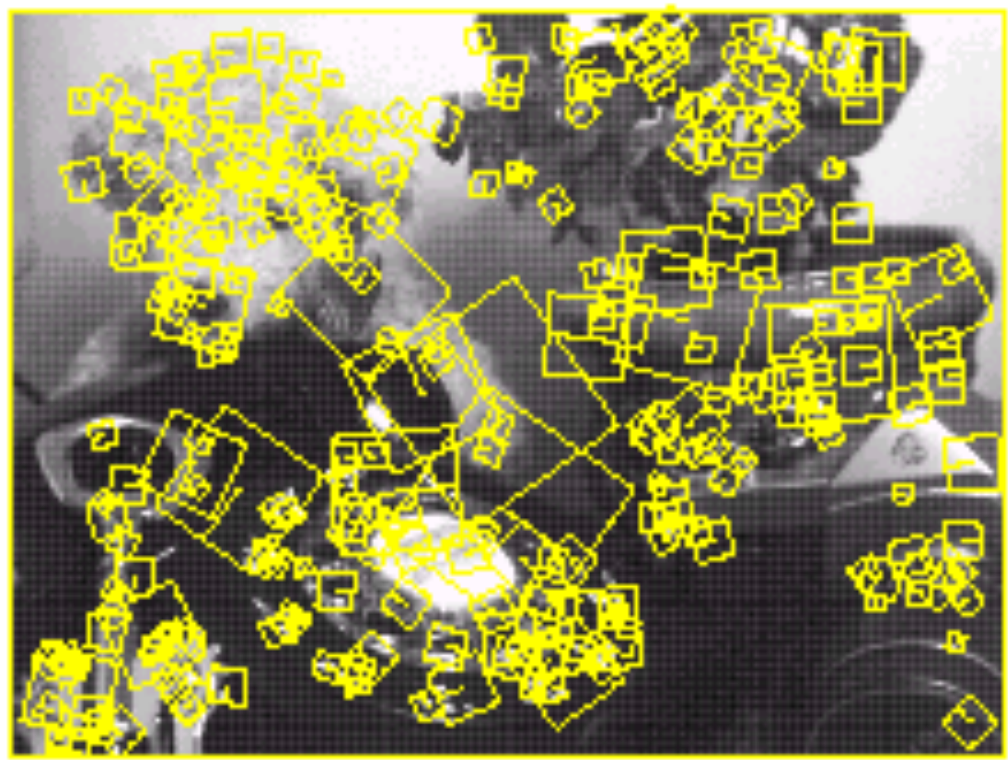
- Only 3 keys are needed for recognition, so extra keys provide robustness
- Affine model is no longer as accurate

Recognition under occlusion



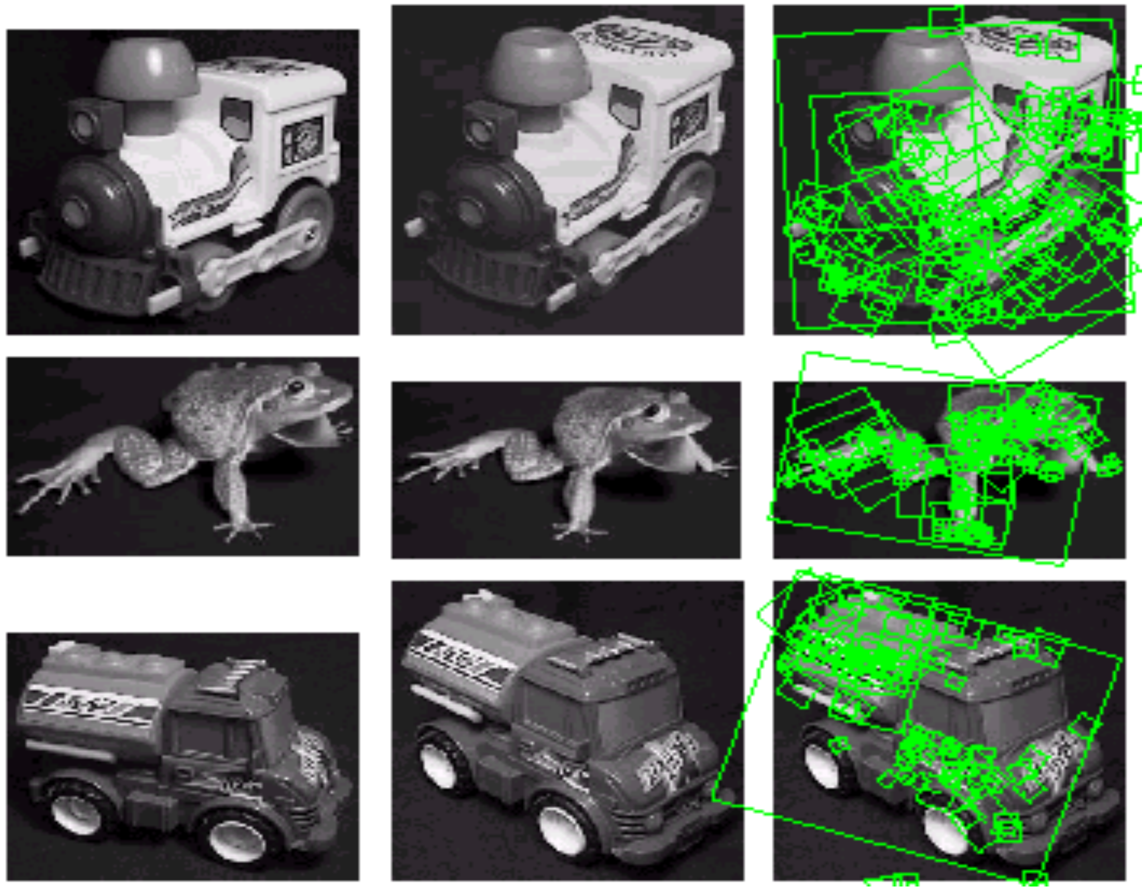
Test of illumination invariance

- Same image under differing illumination



273 keys verified in final match

Examples of view interpolation



Location recognition

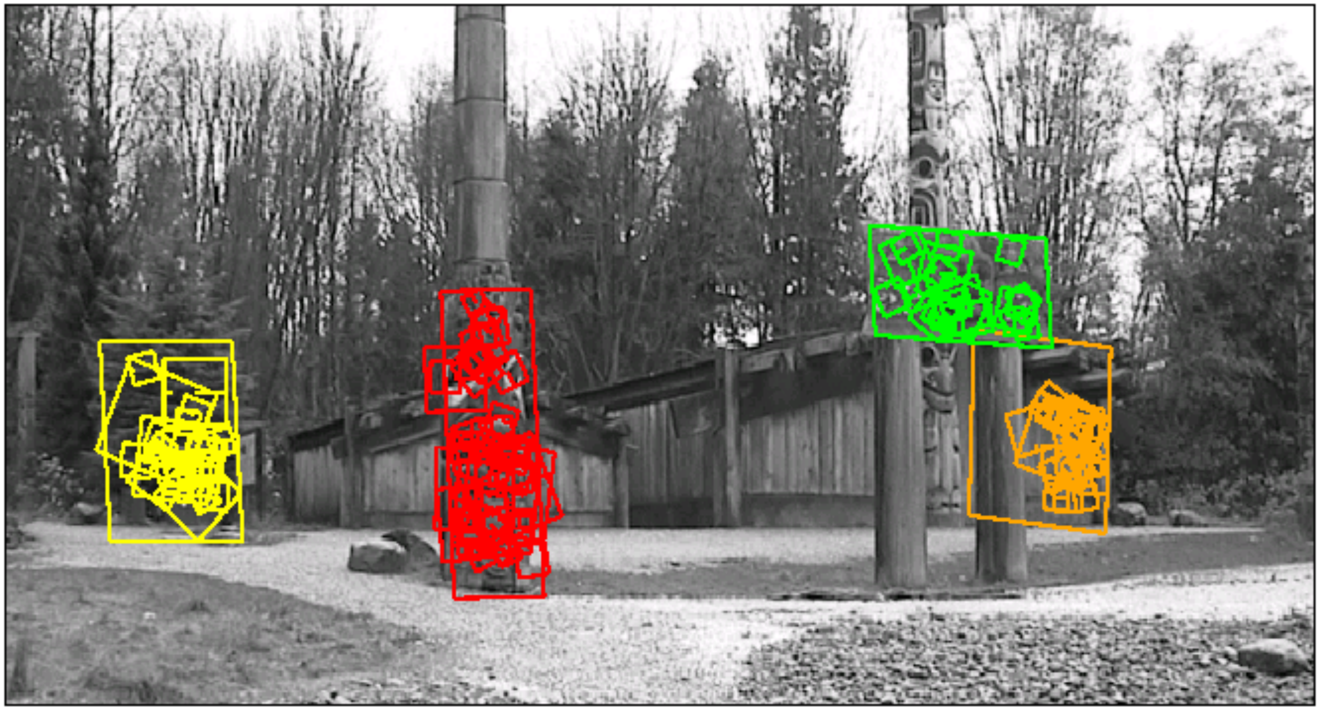
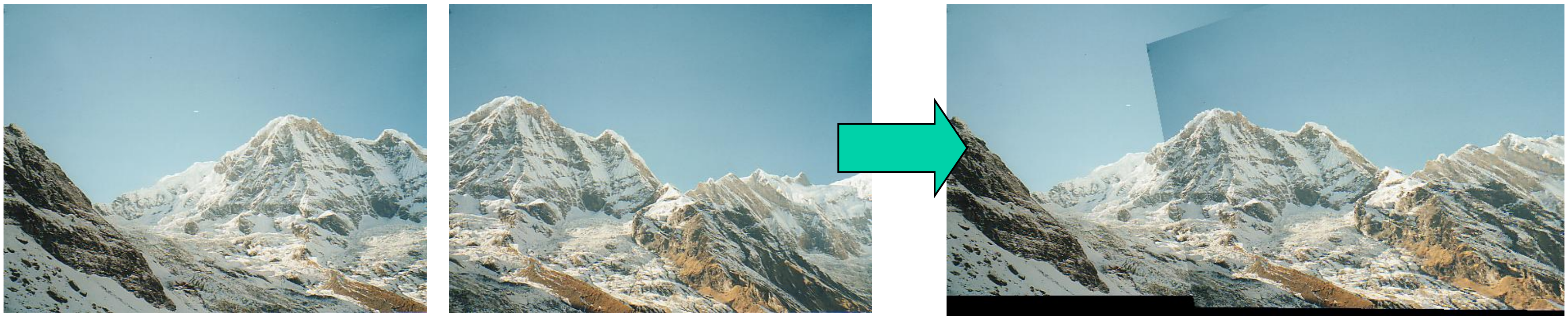
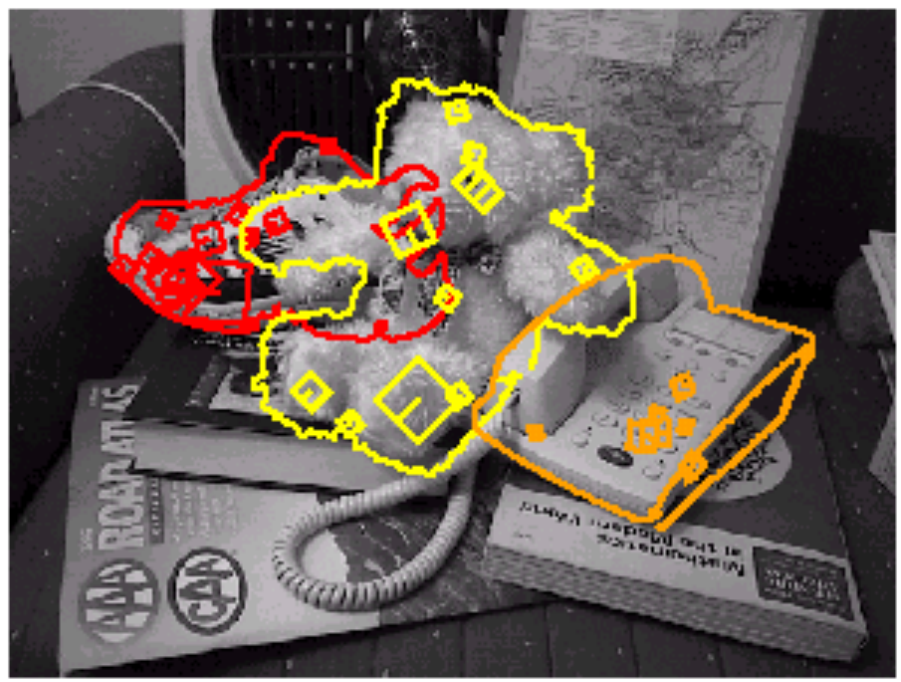


Image alignment: Challenges



Small degree of overlap

Intensity changes



Occlusion,
clutter

Invariant Local Features

- Model verification
- For each set of features matched to object O – verify whether they are geometrically consistent
- Examine all clusters with at least 3 features
- Perform least-squares affine fit to model.
- Discard outliers and perform top-down check for additional features.
- Evaluate probability that match is correct

Solution for affine parameters

- Affine transform of [x,y] to [u,v]:

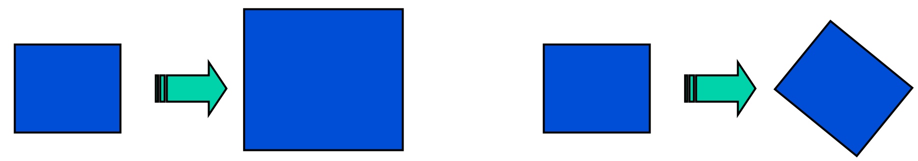
$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

- Rewrite to solve for transform parameters:

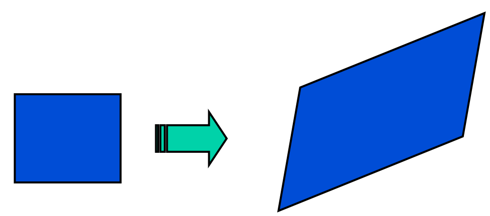
$$\begin{bmatrix} x & y & 0 & 0 & 1 & 0 \\ 0 & 0 & x & y & 0 & 1 \\ & & \cdots & & & \\ & & \cdots & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_x \\ t_y \end{bmatrix} = \begin{bmatrix} u \\ v \\ \vdots \end{bmatrix}$$

2D transformation models

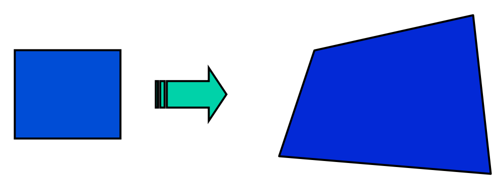
- Similarity (translation, scale, rotation)



- Affine

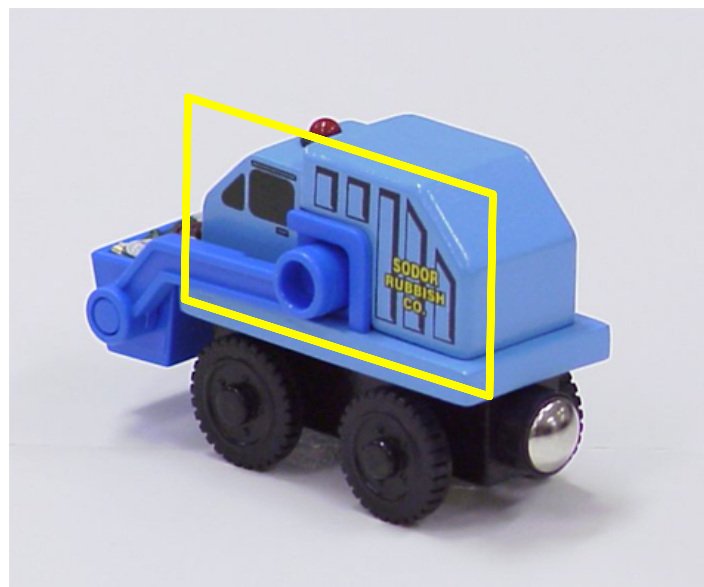


- Projective (homography)

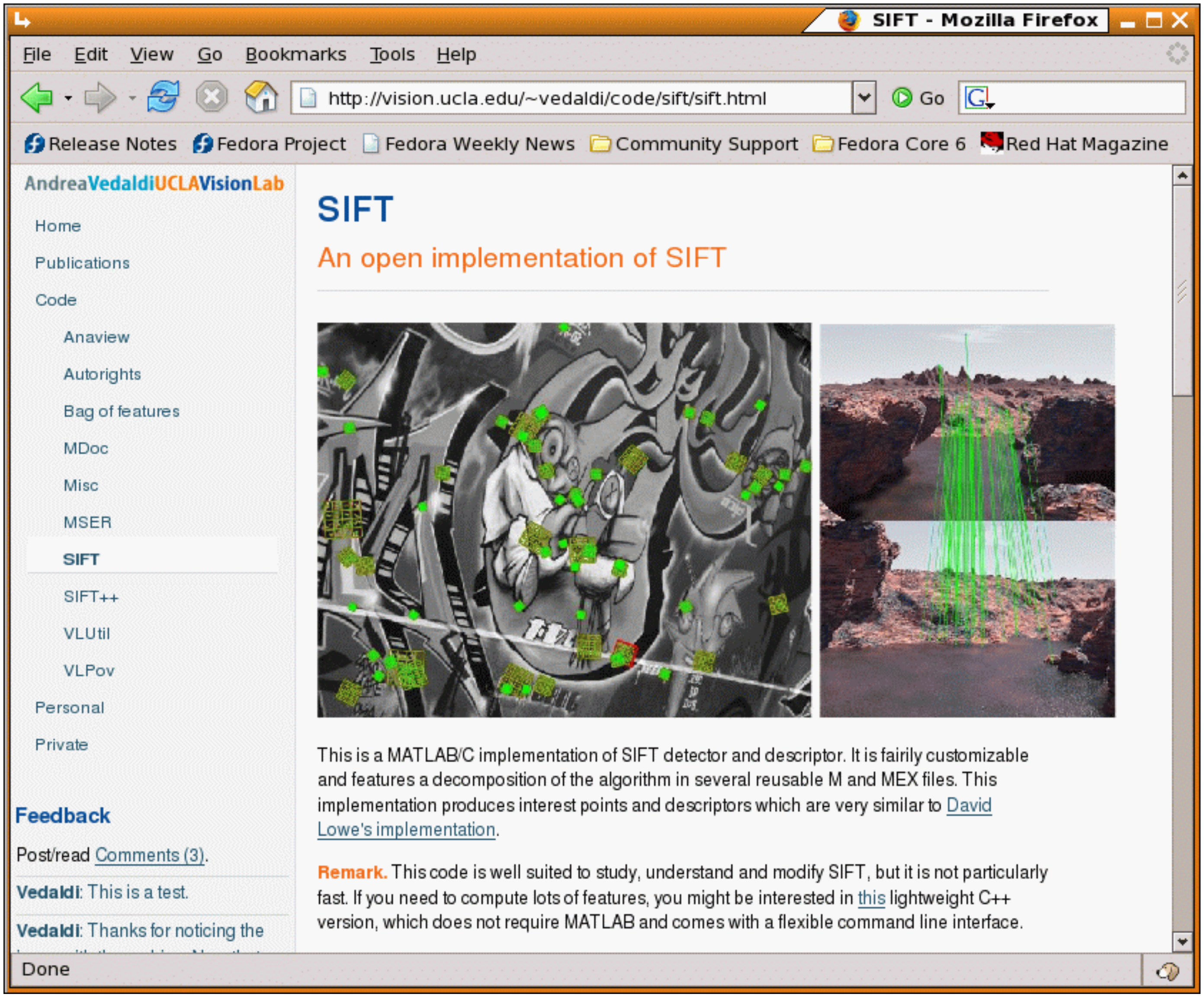


Let's start with affine transformations

- Simple fitting procedure (linear least squares)
- Approximates viewpoint changes for roughly planar objects and roughly orthographic cameras
- Can be used to initialize fitting for more complex models



SOFTWARE for Matlab (at UCLA, Oxford)
www.VLFeat.org

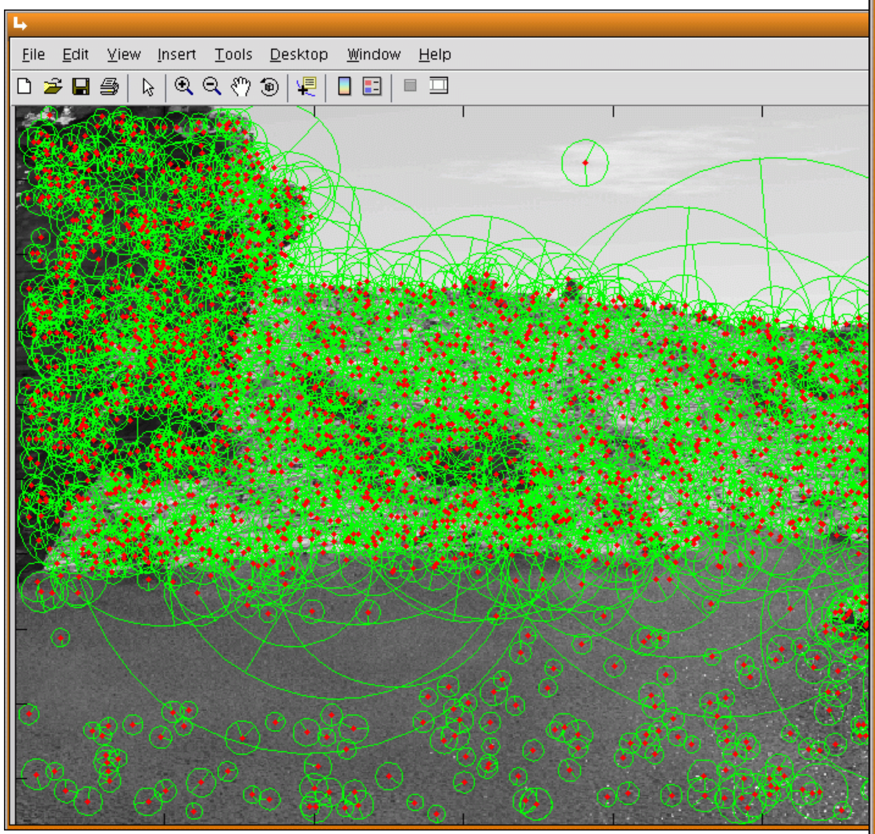


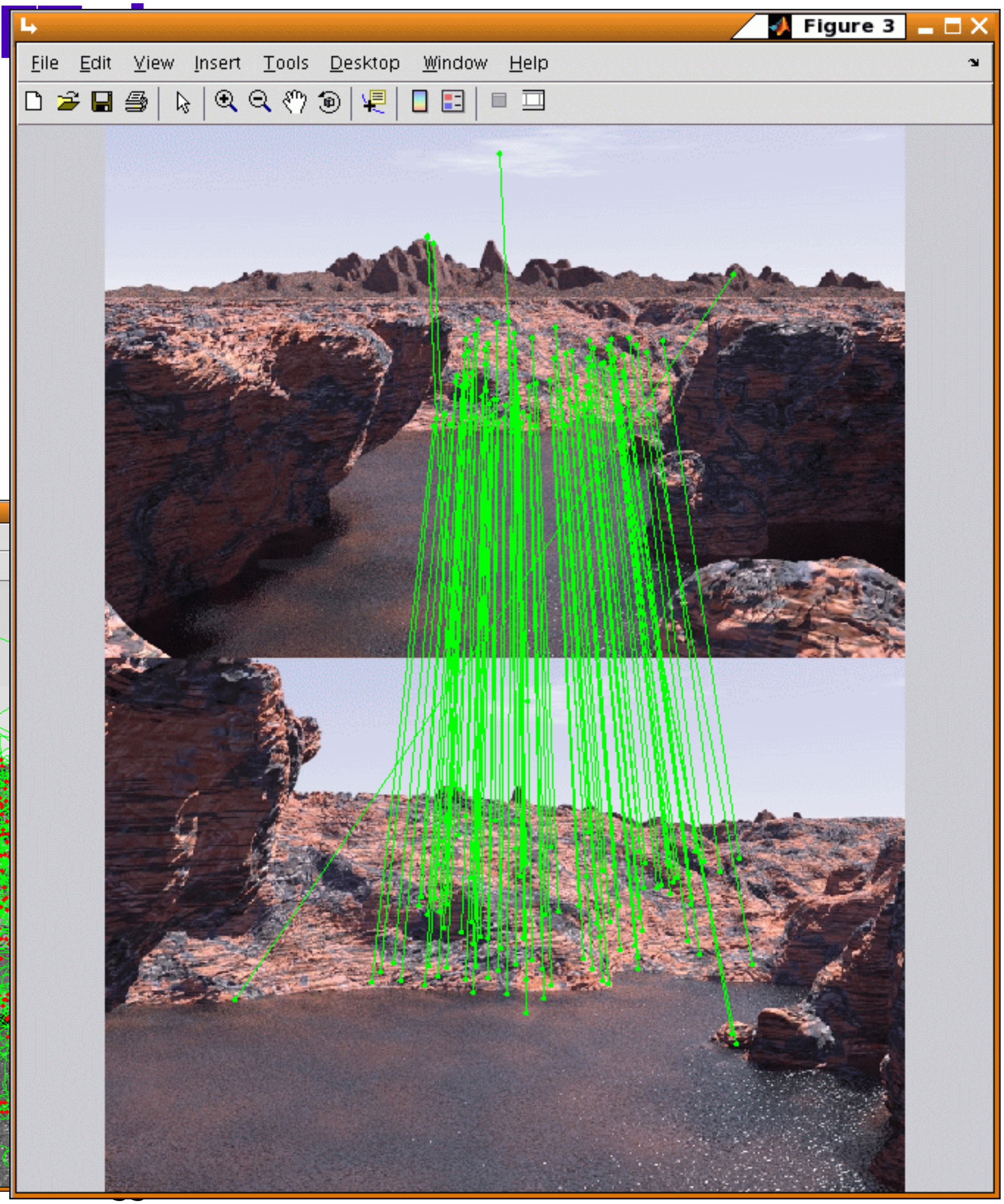
SIF

Run

sift_compile

sift_demo2





SIFT On-A-Slide

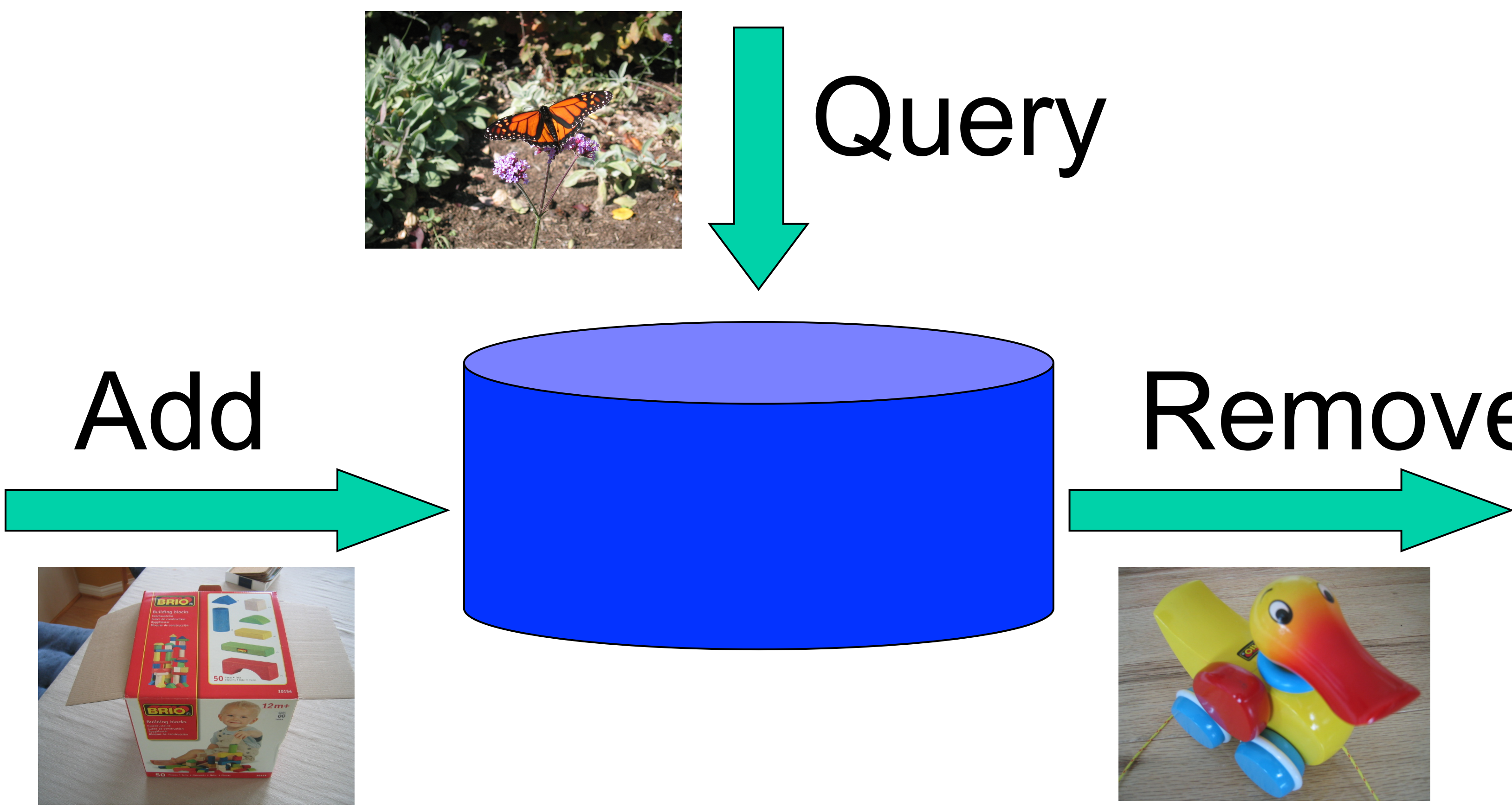
1. Enforce invariance to scale: Compute Gaussian difference max, for many different scales; non-maximum suppression, find local maxima: keypoint candidates
2. Localizable corner: For each maximum fit quadratic function. Compute center with sub-pixel accuracy by setting first derivative to zero.
3. Eliminate edges: Compute ratio of eigenvalues, drop keypoints for which this ratio is larger than a threshold.
4. Enforce invariance to orientation: Compute orientation, to achieve rotation invariance, by finding the strongest second derivative direction in the smoothed image (possibly multiple orientations). Rotate patch so that orientation points up.
5. Compute feature signature: Compute a "gradient histogram" of the local image region in a 4x4 pixel region. Do this for 4x4 regions of that size. Orient so that largest gradient points up (possibly multiple solutions). Result: feature vector with 128 values (15 fields, 8 gradients).
6. Enforce invariance to illumination change and camera saturation: Normalize to unit length to increase invariance to illumination. Then threshold all gradients, to become invariant to camera saturation.

Nearest-neighbor matching to feature database

- Hypotheses are generated by **approximate nearest neighbor** matching of each feature to vectors in the database
  - SIFT use best-bin-first (Beis & Lowe, 97) modification to k-d tree algorithm
  - Use heap data structure to identify bins in order by their distance from query point

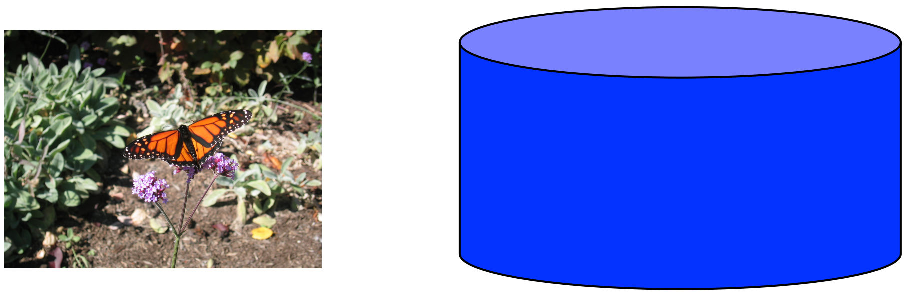
- **Result:** Can give speedup by factor of 1000 while finding nearest neighbor (of interest) 95% of the time

Adding, Querying and Removing Images at full speed

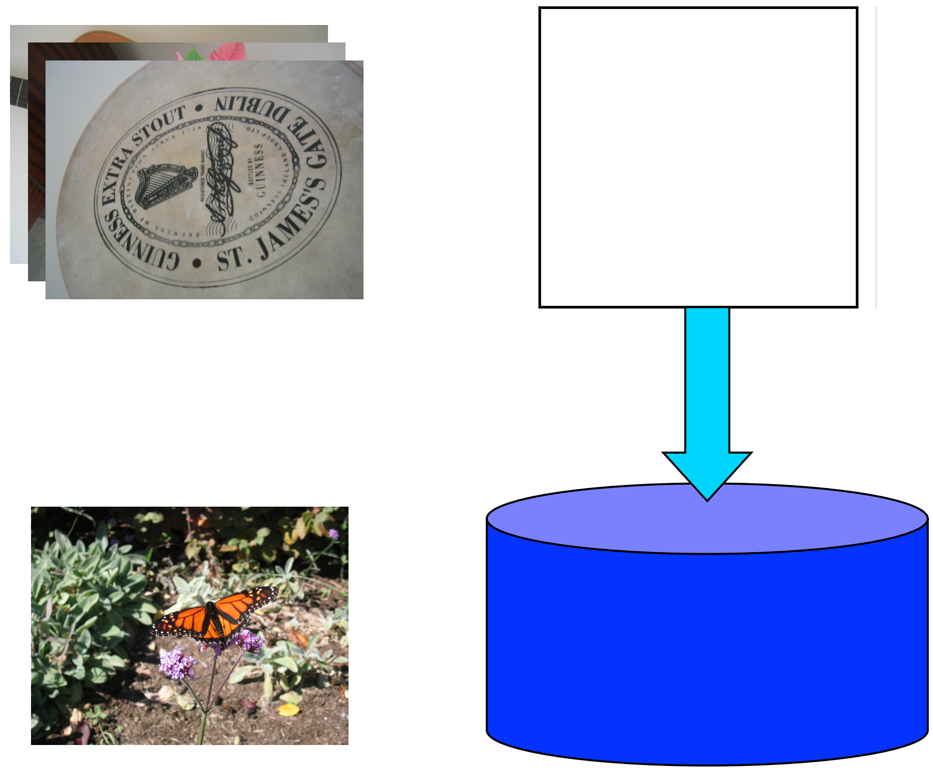


Training and Addition are Separate

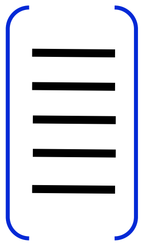
Common Approach



Our approach

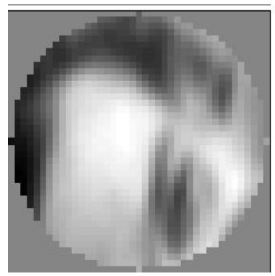


1. Feature extraction

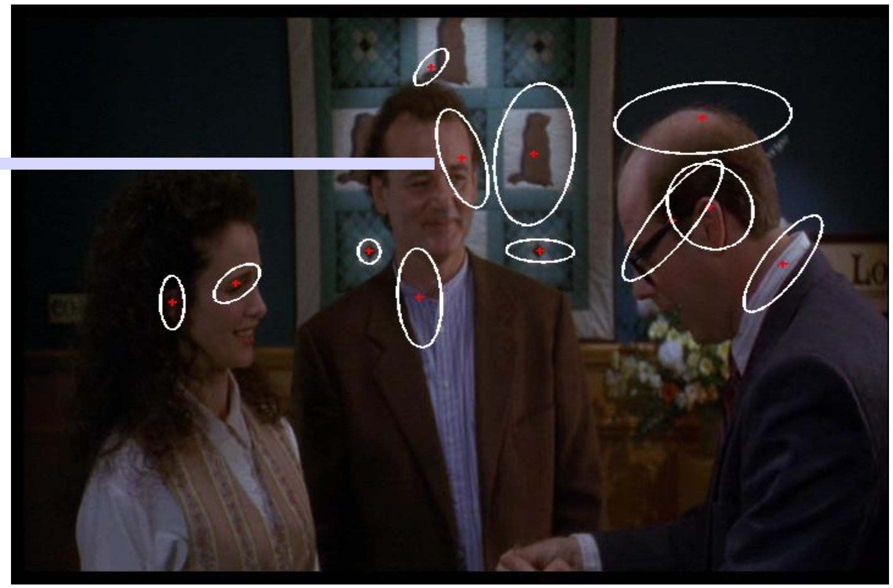


Compute SIFT descriptor

[Lowe’99]



Normalize patch



Detect patches

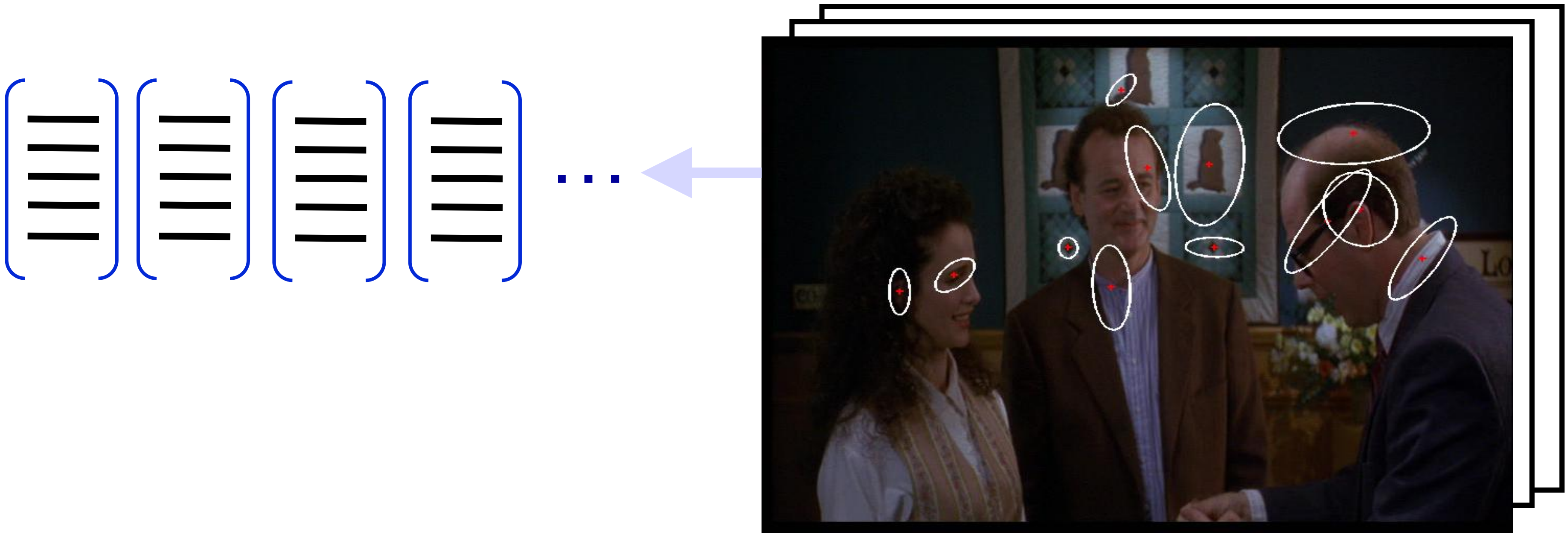
[Mikojaczyk and Schmid ’02]

[Mata, Chum, Urban & Pajdla, ’02]

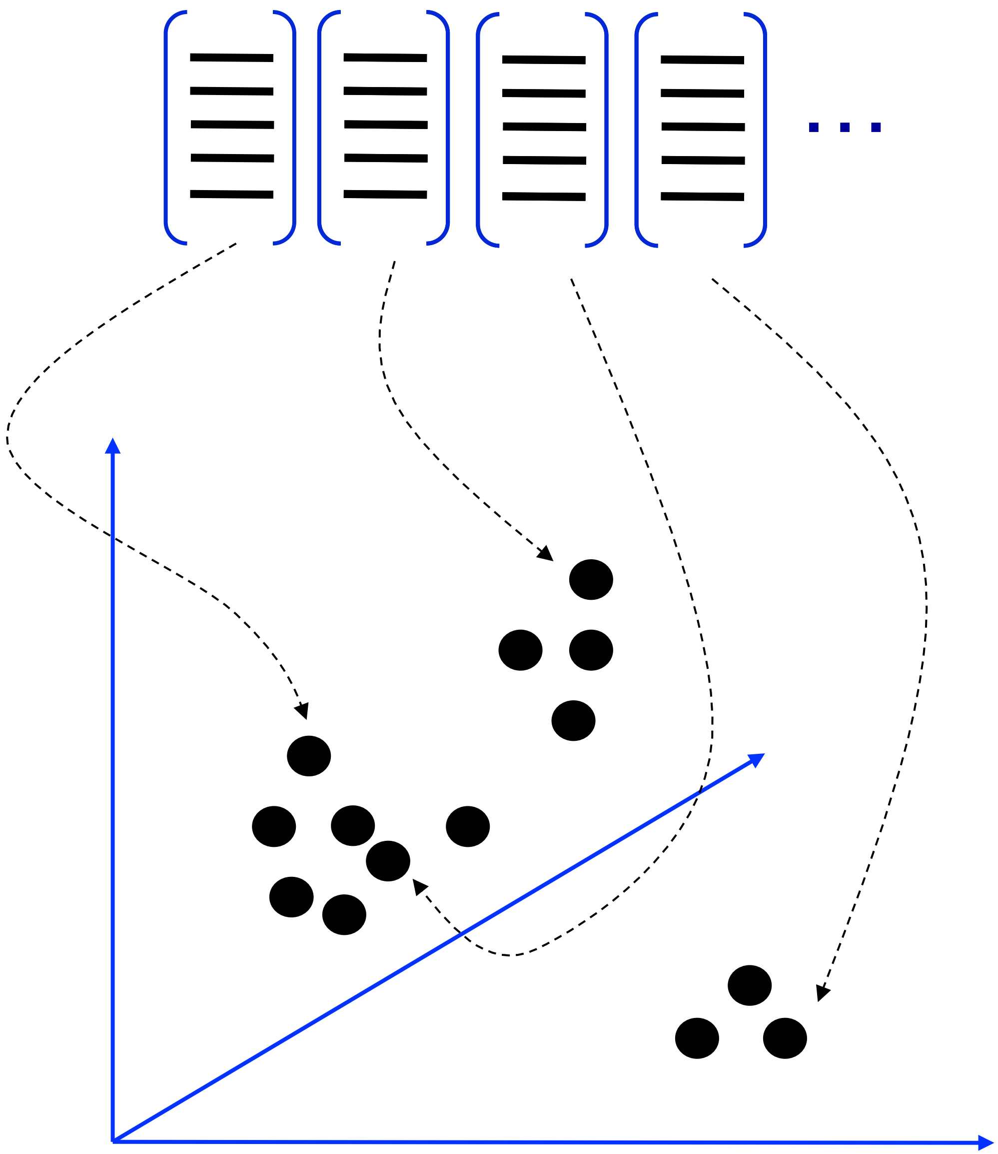
[Sivic & Zisserman, ’03]

Slide credit: Josef Sivic

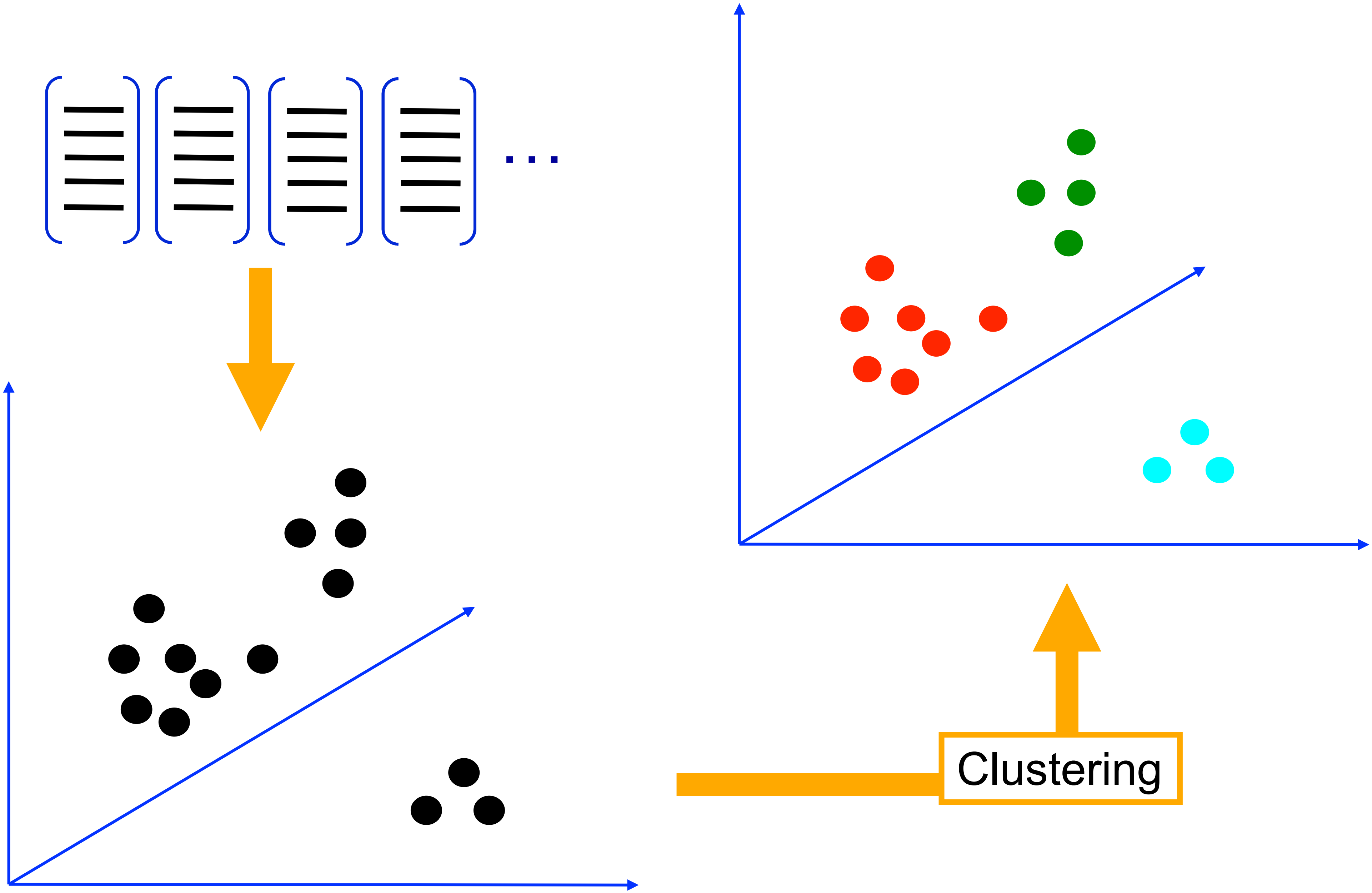
1. Feature extraction



2. Learning the visual vocabulary

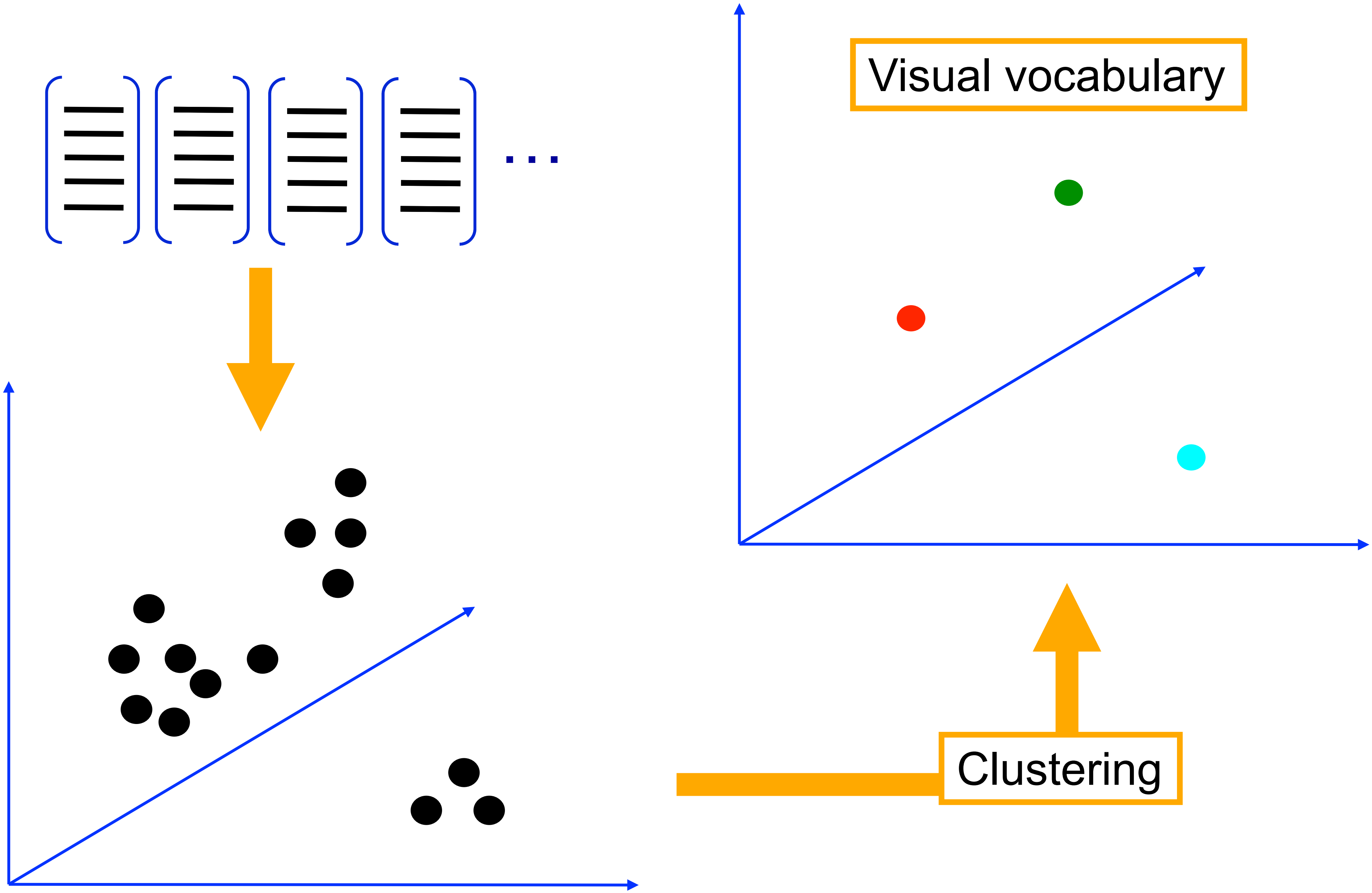


2. Learning the visual vocabulary



Slide credit: Josef Sivic

2. Learning the visual vocabulary



Slide credit: Josef Sivic

K-means clustering

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

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

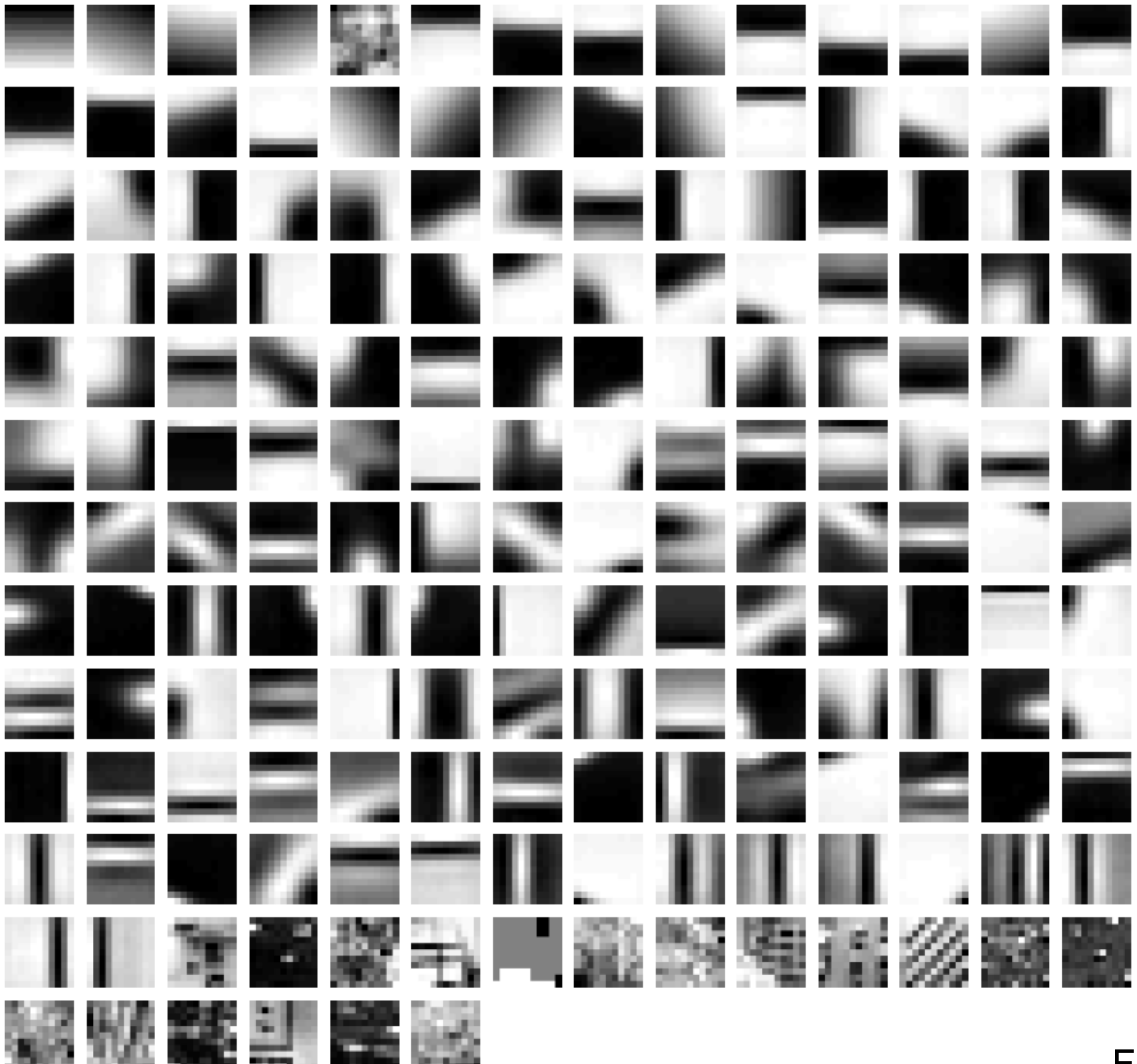
- Algorithm:
- Randomly initialize K cluster centers
- Iterate until convergence:
  - Assign each data point to the nearest center
  - Recompute each cluster center as the mean of all points assigned to it

From clustering to vector quantization

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

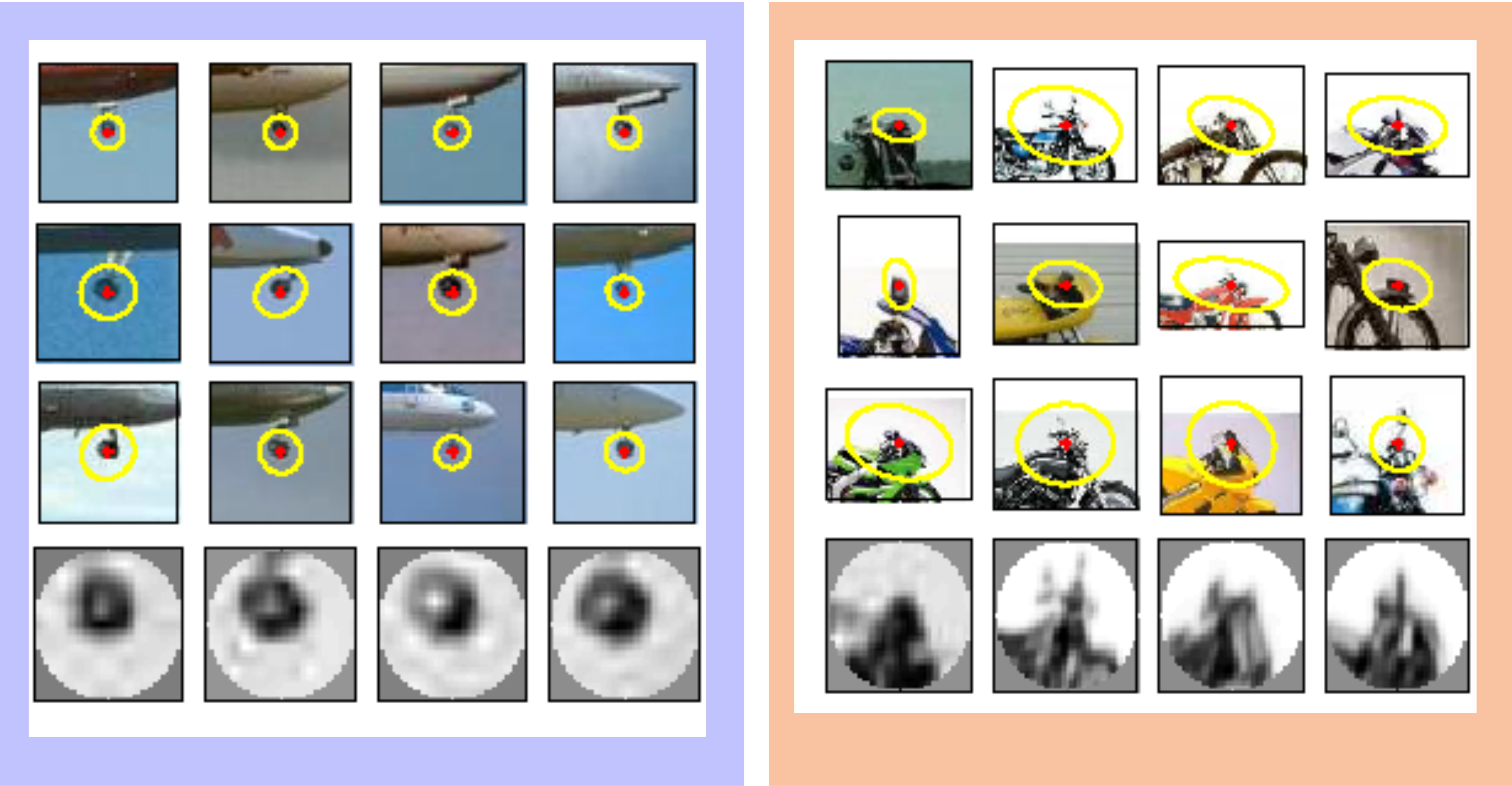
- The codebook is used for quantizing features
  - A *vector quantizer* takes a feature vector and maps it to the index of the nearest codevector in a codebook

Example visual vocabulary



Fei-Fei et al. 2005

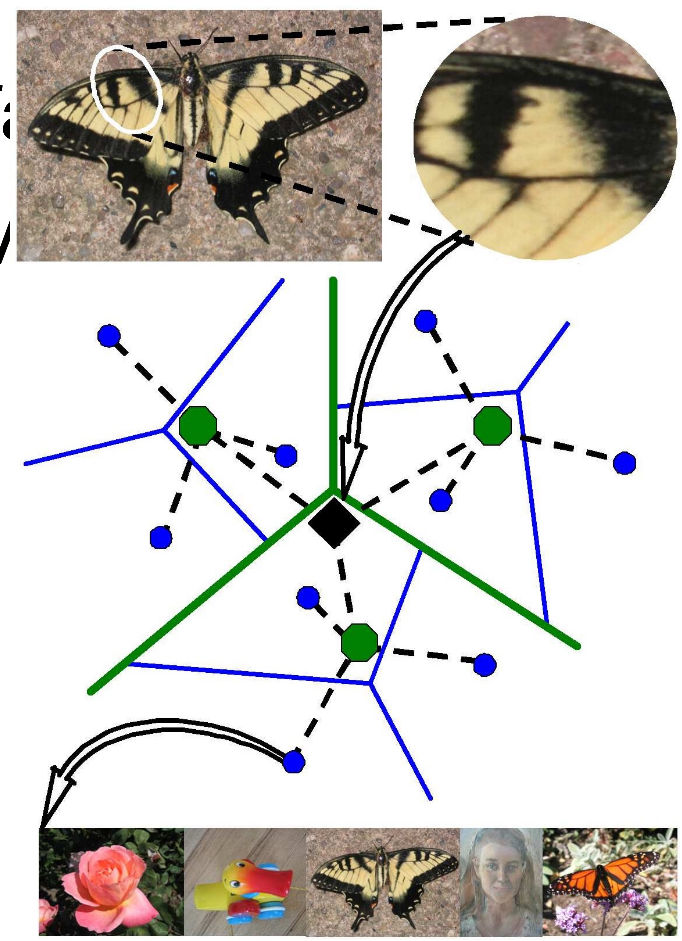
Image patch examples of visual words



Sivic et al. 2005

Visual vocabularies: Issues

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



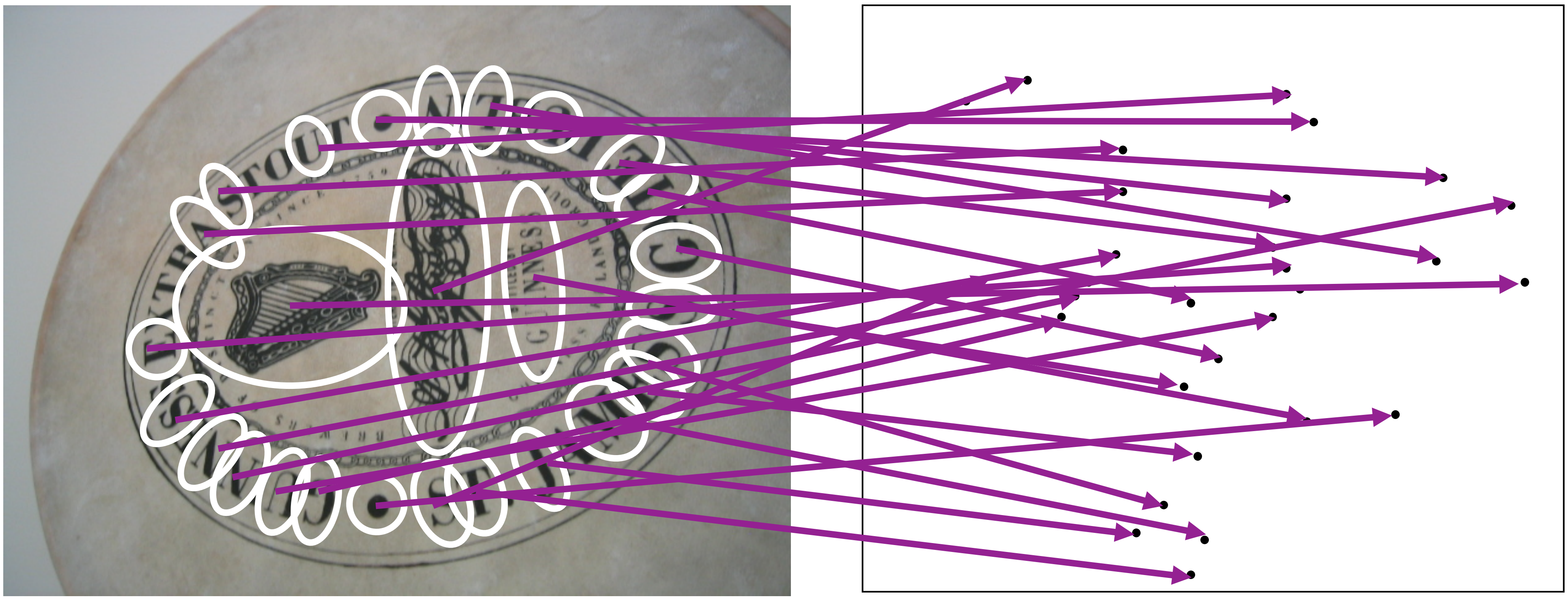
Hierarchical k-means

- We have many, many of these features
- 100000 images ~1000 features per image
- If we can get repeatable, discriminative features,
- then recognition can scale to very large databases
- using the vocabulary tree and indexing approach

- Quantize the feature descriptor space + efficient search
- Flat k-means , Approximate Nearest Neighbour Methods
- Hierarchical k-means - Nister&Stewenius [CVPR 2006]
- Visual vocabulary trees

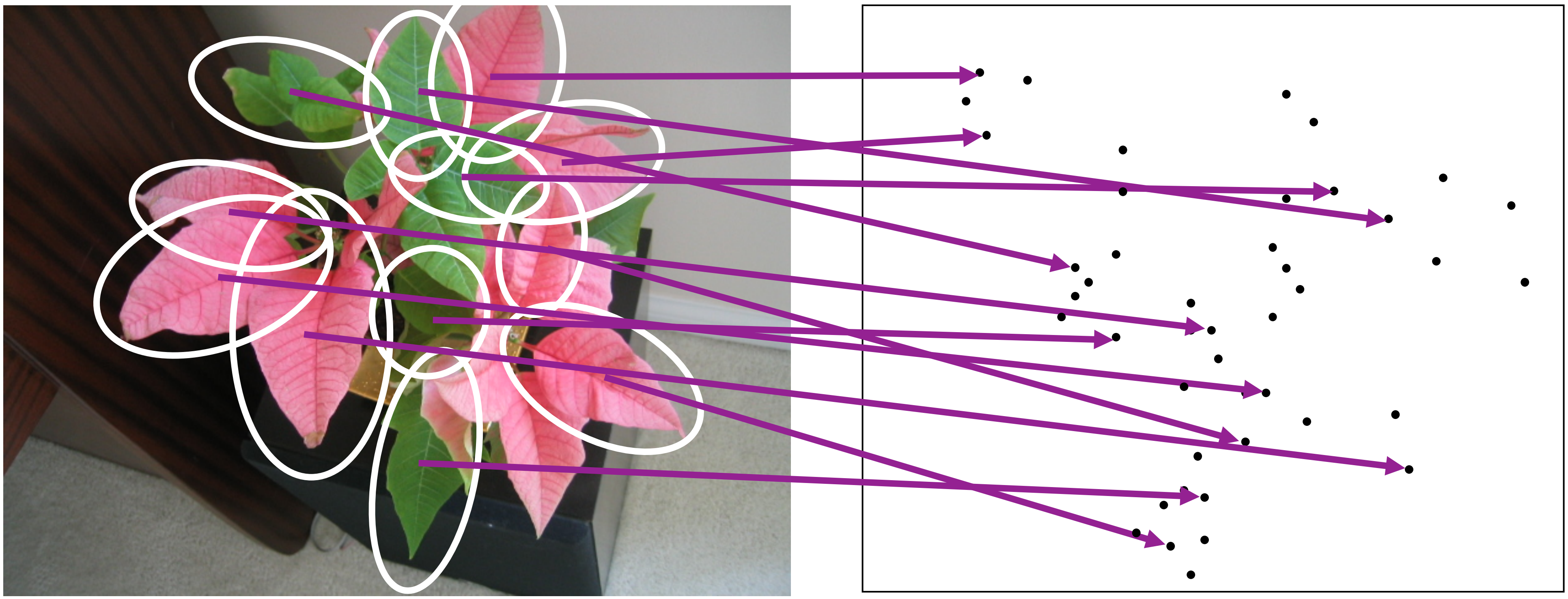
Slides from Nister & Stewenius 06

Building Visual Vocabulary Tree



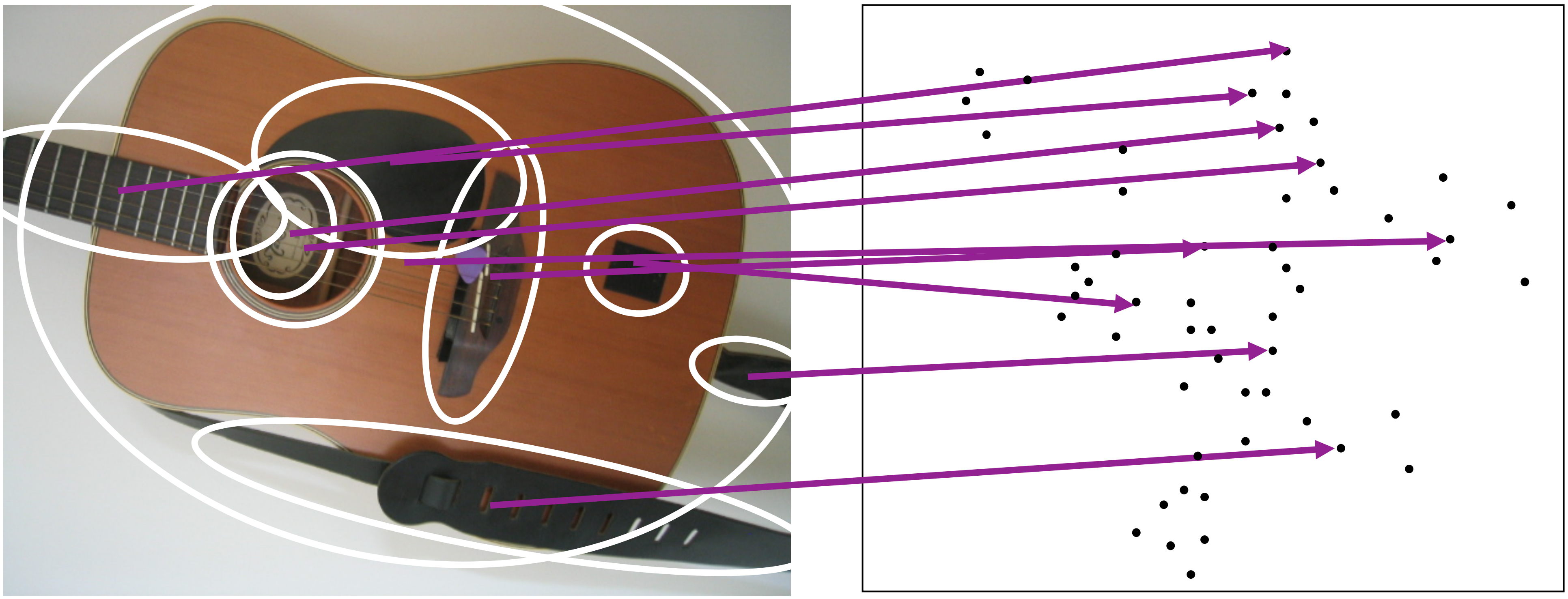
Slides from Nister & Stewenius 06

Building Visual Vocabulary Tree



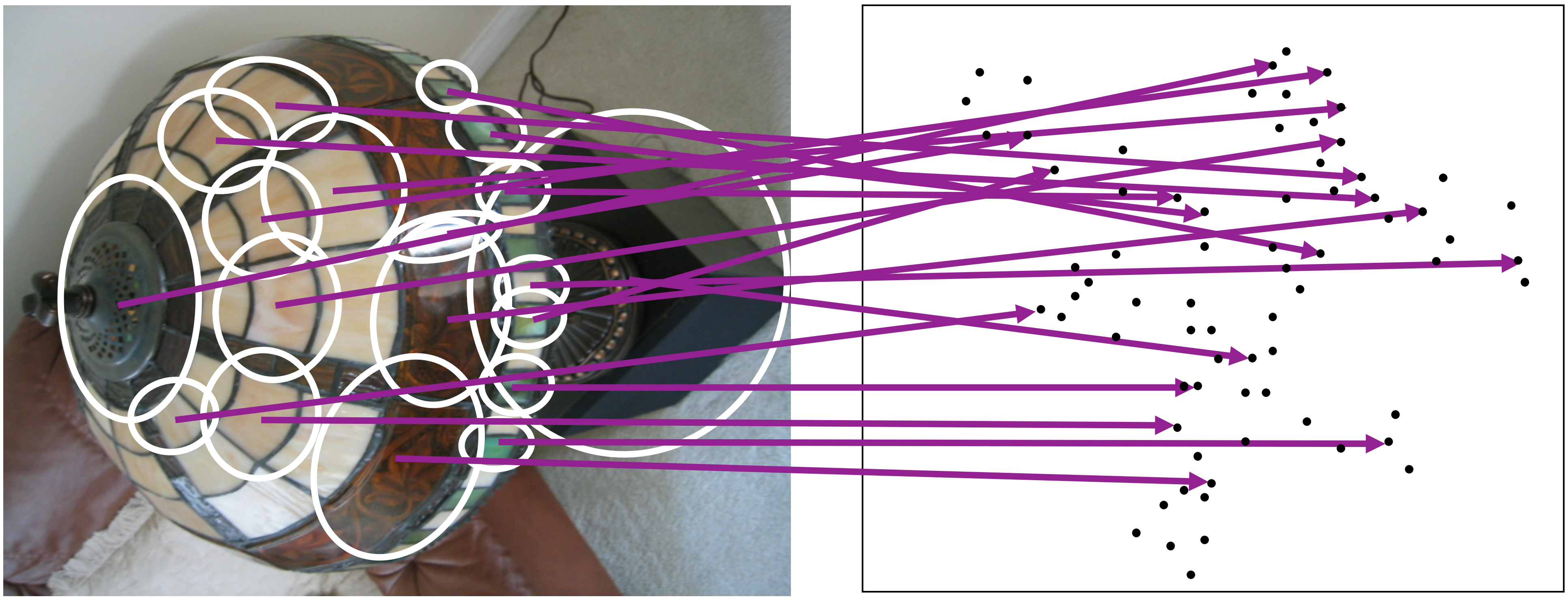
Slides from Nister & Stewenius 06

Building Visual Vocabulary Tree



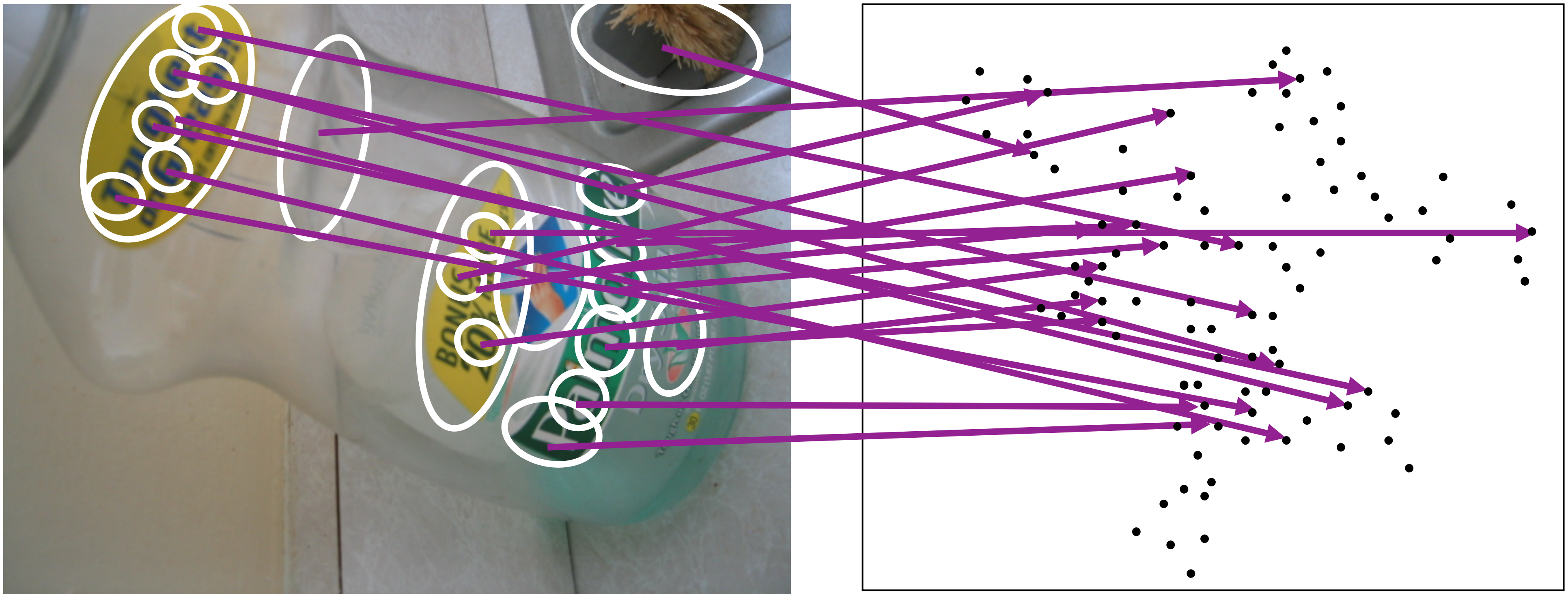
Slides from Nister & Stewenius 06

Building Visual Vocabulary Tree



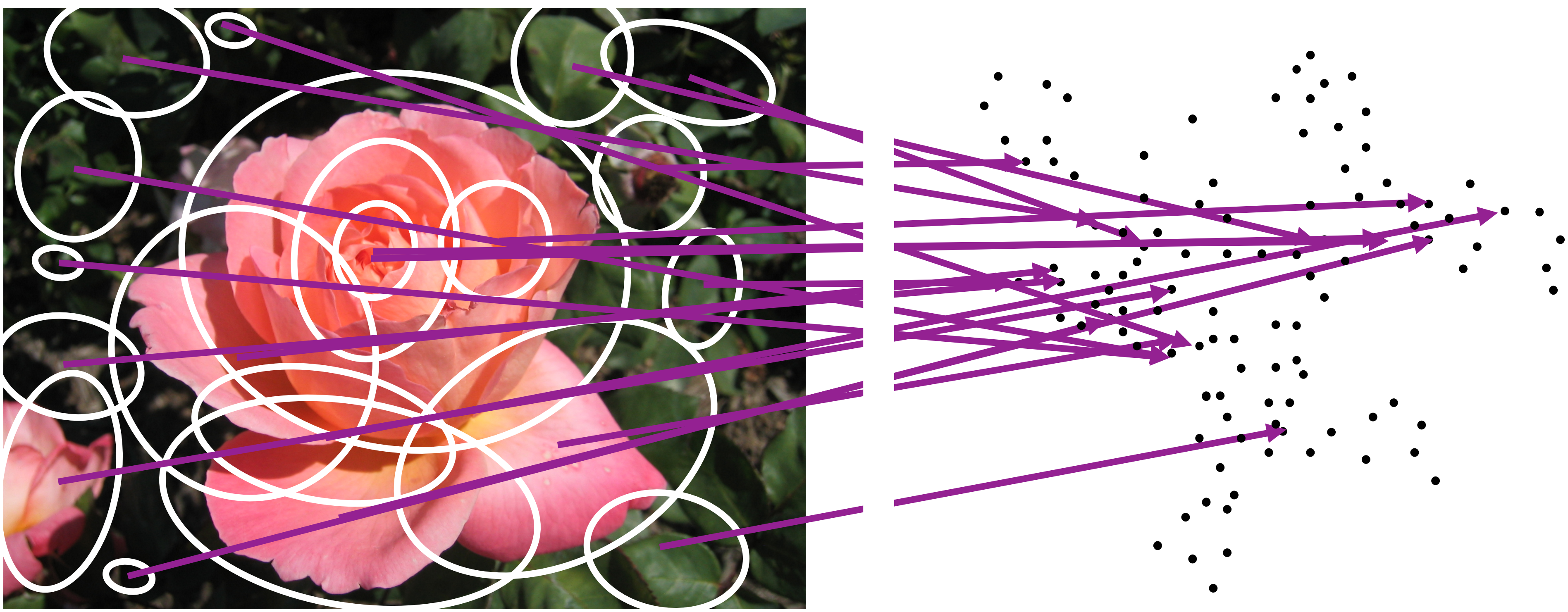
Slides from Nister & Stewenius 06

Building Visual Vocabulary Tree



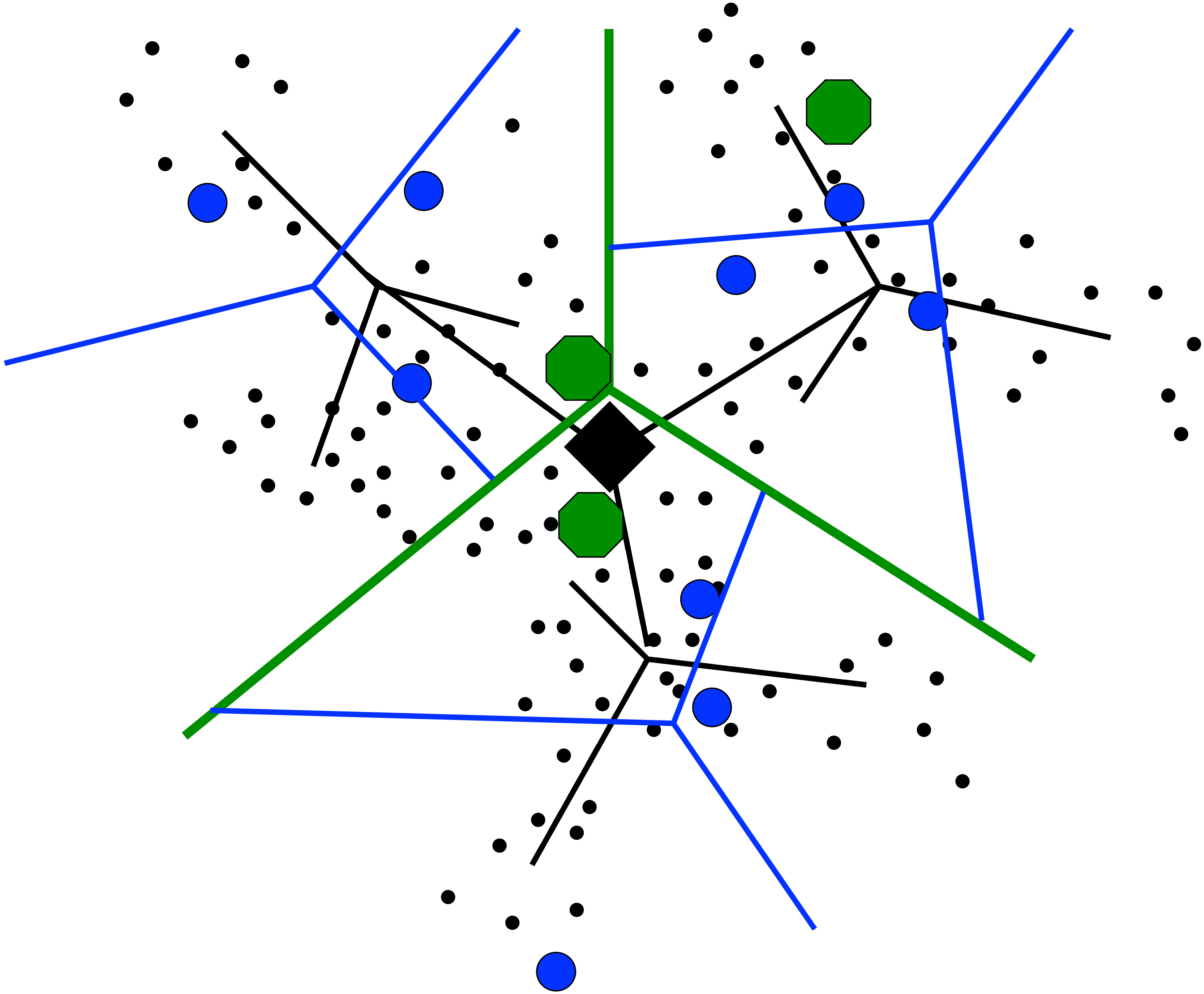
Slides from Nister & Stewenius 06

Building Visual Vocabulary Tree

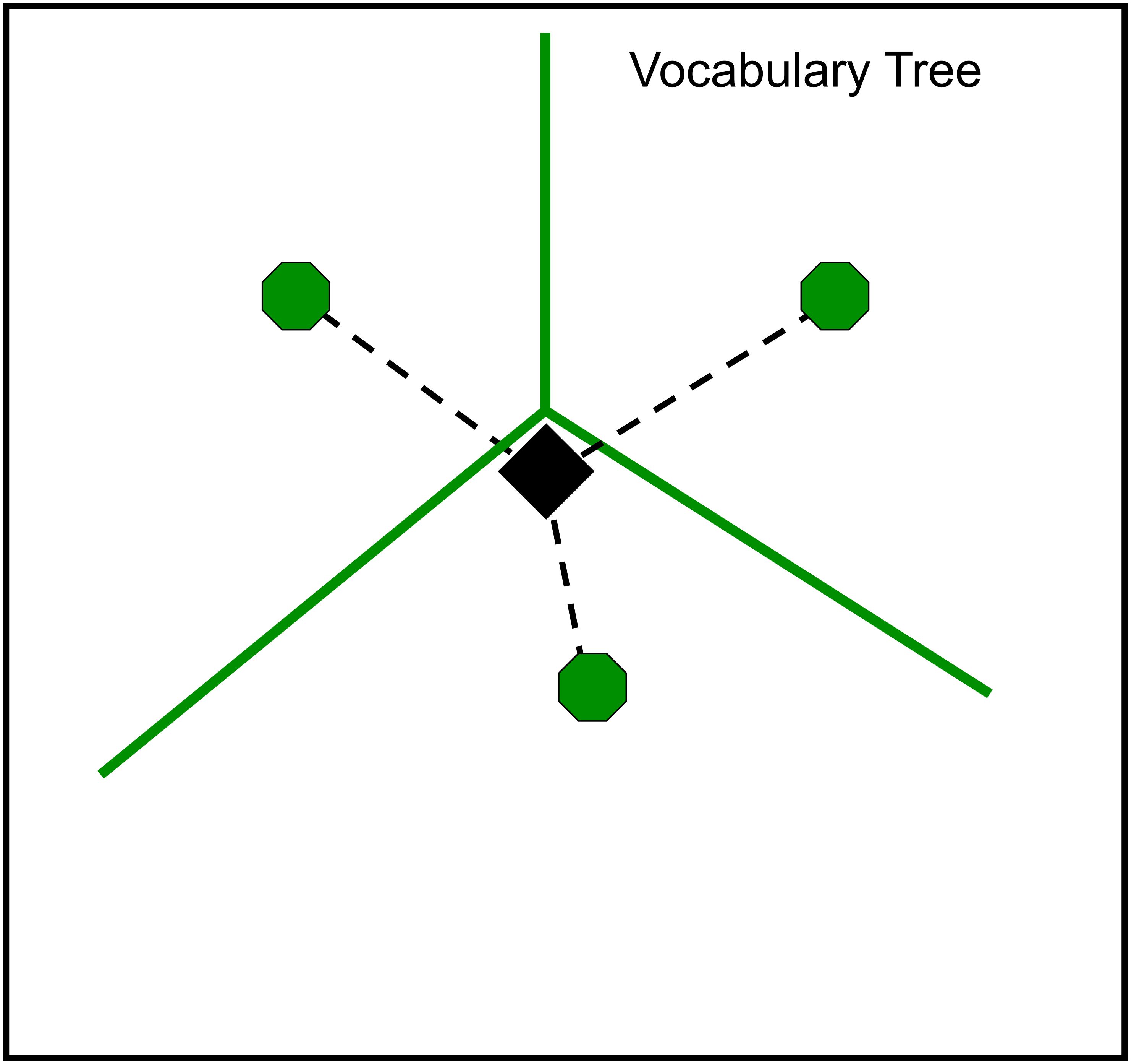


Slides from Nister & Stewenius 06

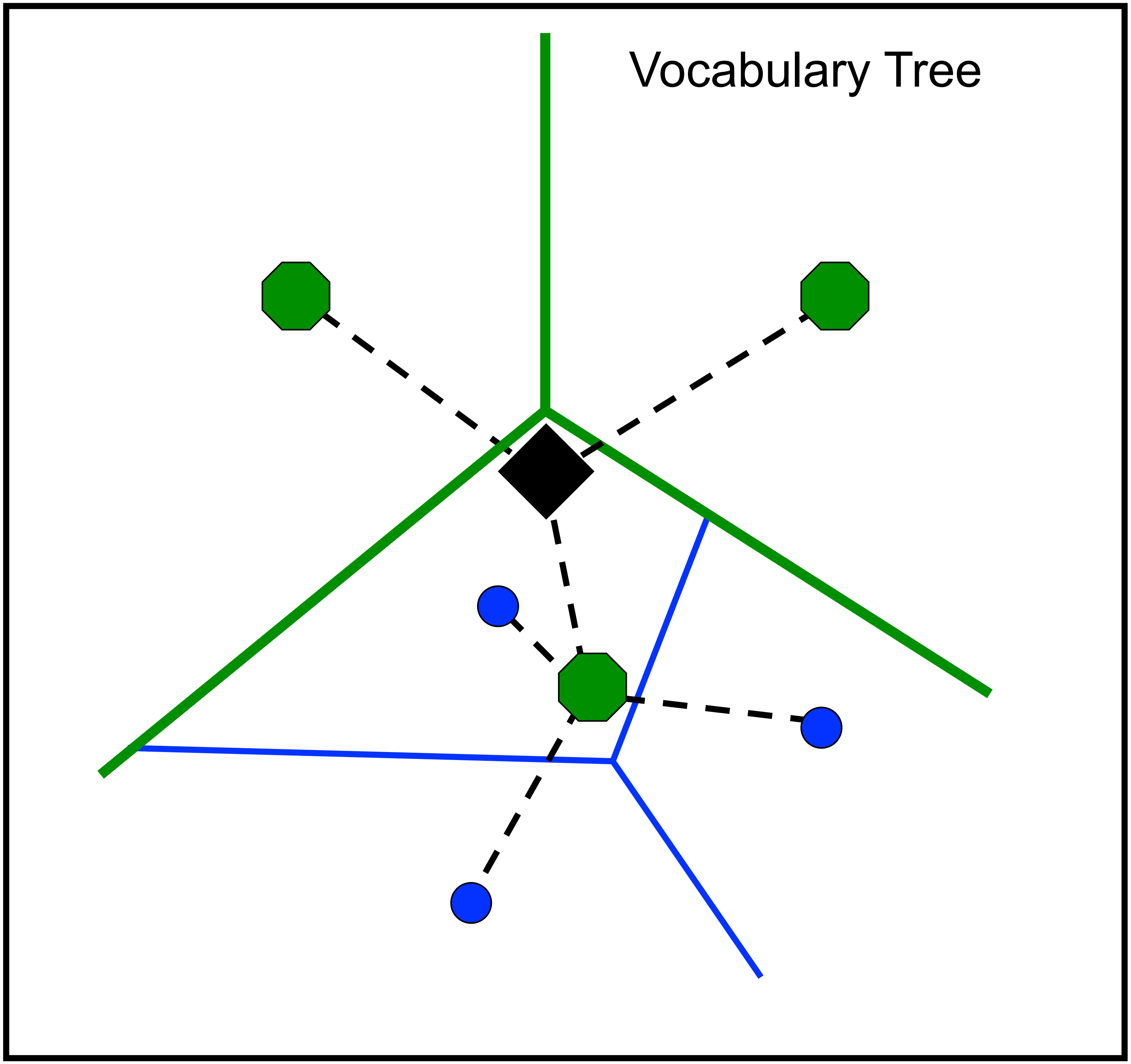
Slides from Nister & Stewenius 06



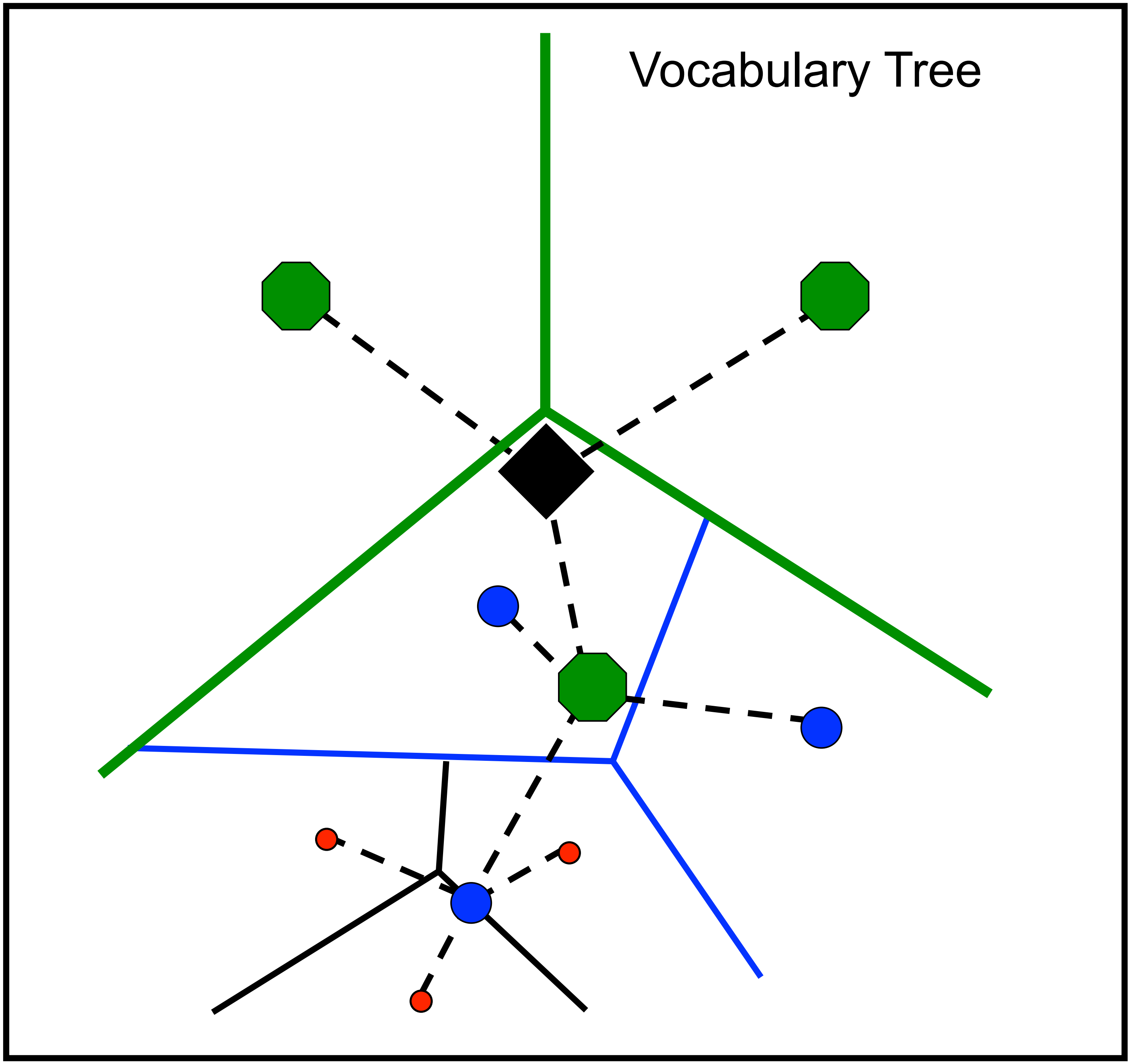
Slides from Nister & Stewenius 06



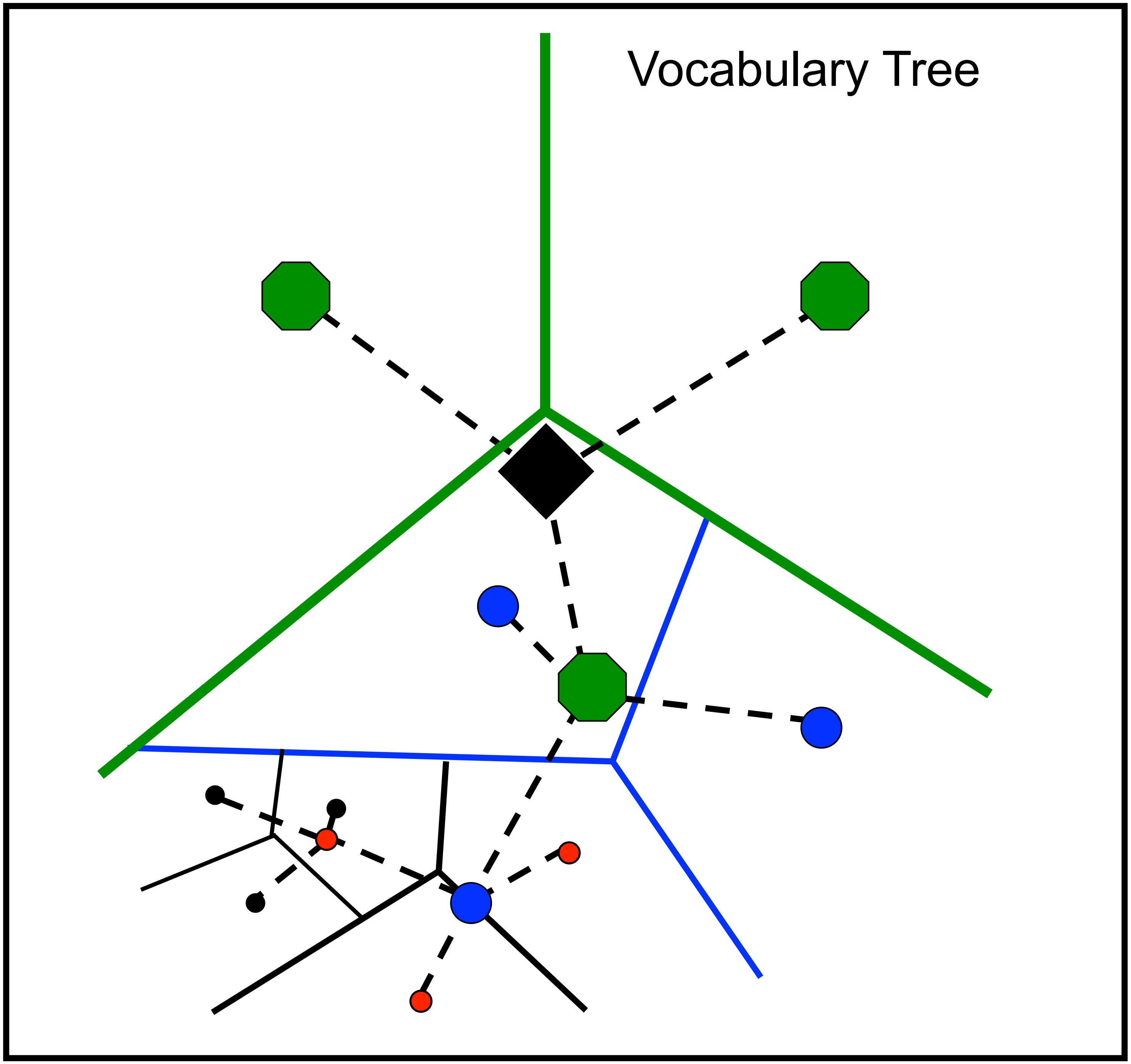
Slides from Nister & Stewenius 06



Slides from Nister & Stewenius 06



Slides from Nister & Stewenius 06



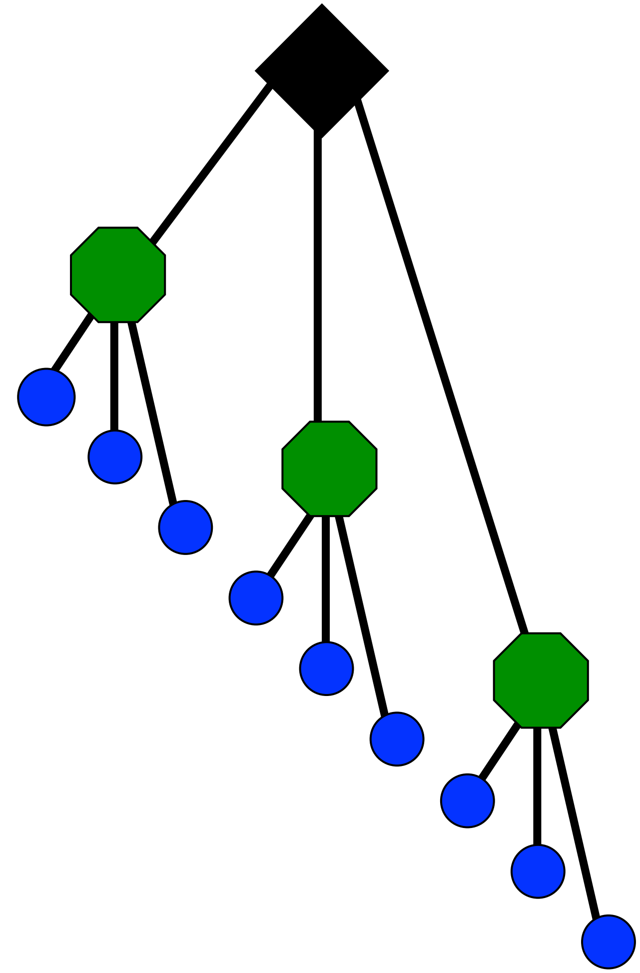
Slides from Nister & Stewenius 06

Vocabulary Trees

- Easy to add/remove images from the database
- Suitable for incremental approach
- Suitable for creating single generic vocabulary
-
- **Approach**
- Extract descriptors from many/many images
- Acquire enough statistics about the descriptor distribution
- Run k-means hierarchically k- is the branching factor of the tree
- E.g. Branching factor of 10 and 6 levels – million leaves

Slides from Nister & Stewenius 06

Vocabulary Trees

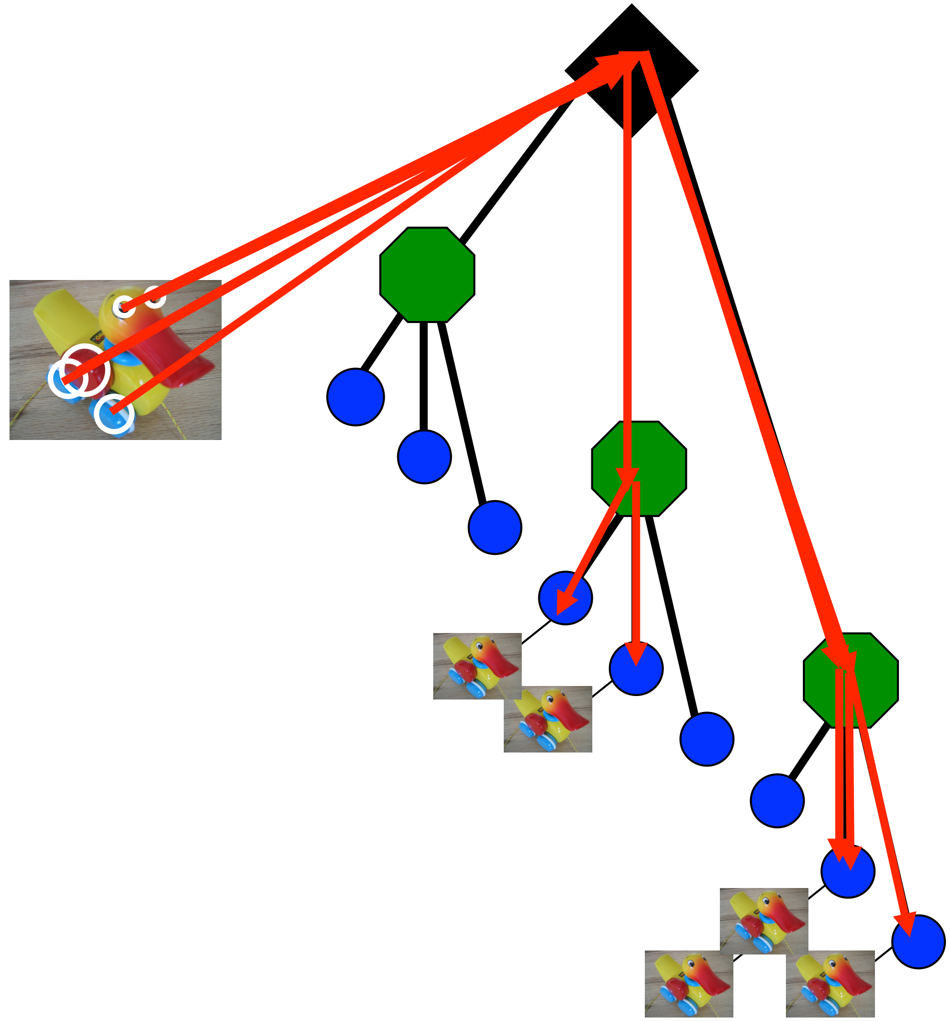


- Training phase – add images to the database
- Extract descriptors – drop it down the tree
- Each node has an inverted file index
- Index to that image is added to all inverted files

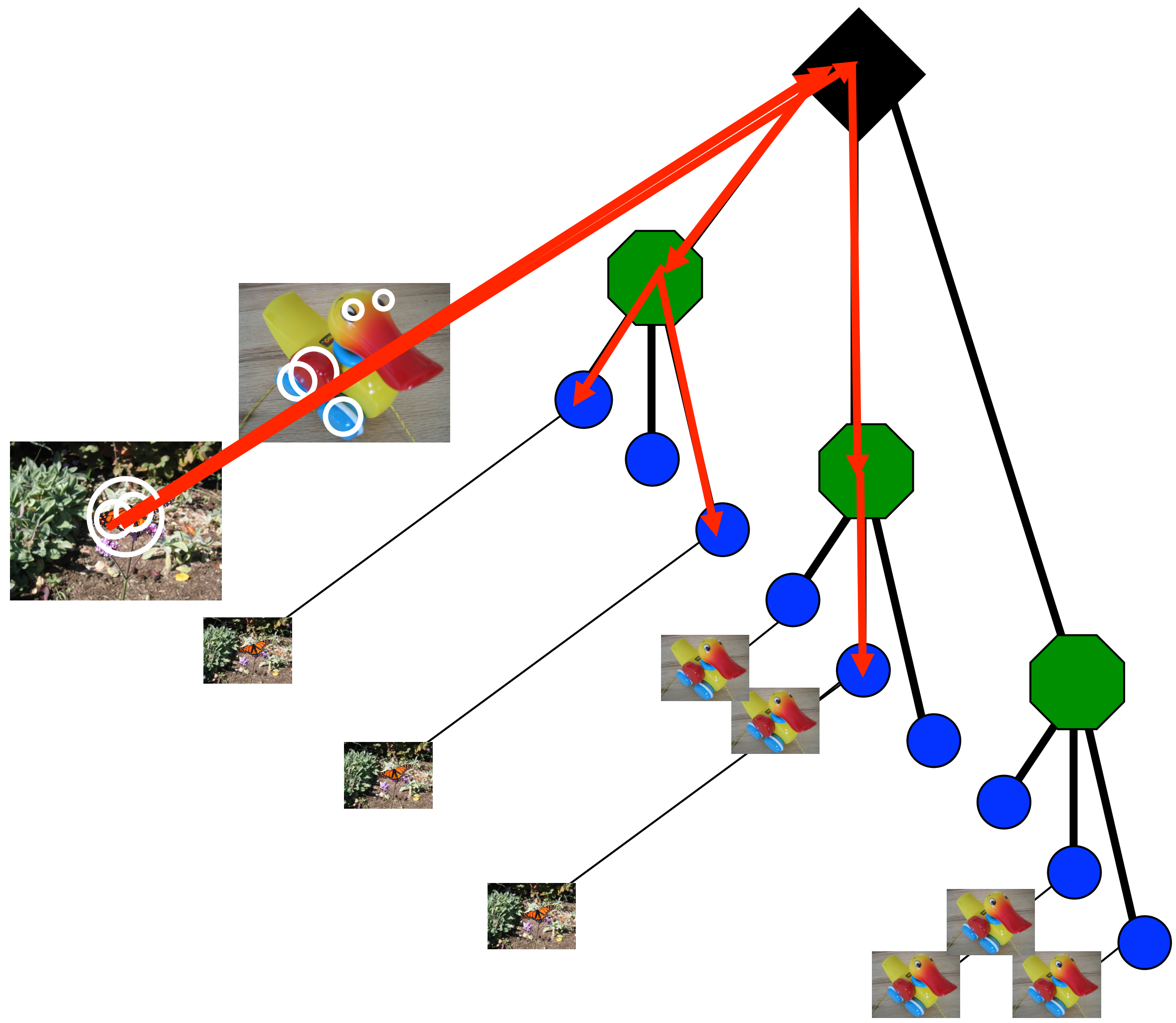
- When we want to query image
- Pass each descriptor down the tree
- Accumulate scores for each image in the database

- At each level do k dot products total of kL dot products
- For k^L leafs and integer descriptors we need Dk^L bytes for 1M leaf nodes use 143 MB of memory

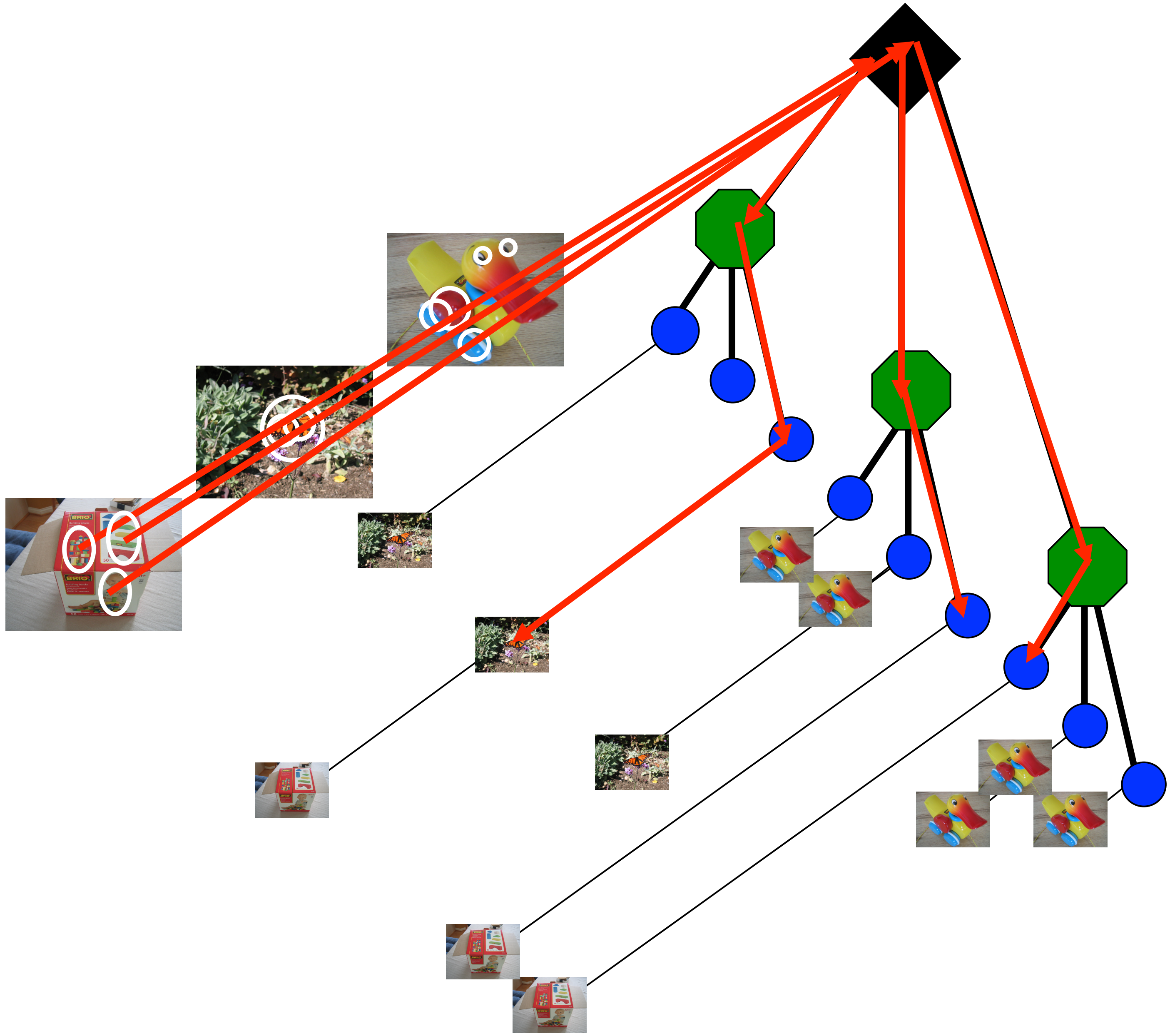
Slides from Nister & Stewenius 06



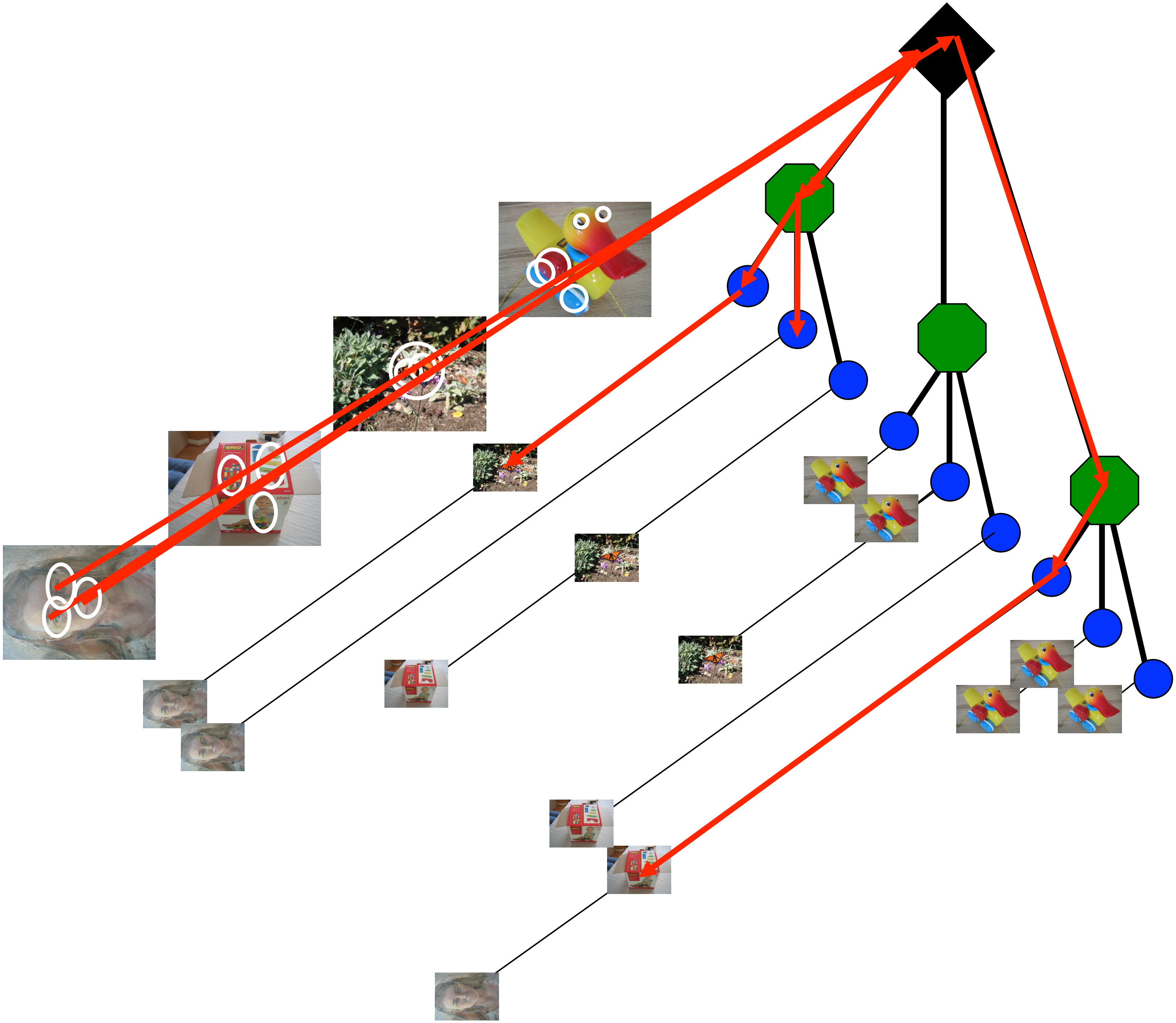
Slides from Nister & Stewenius 06



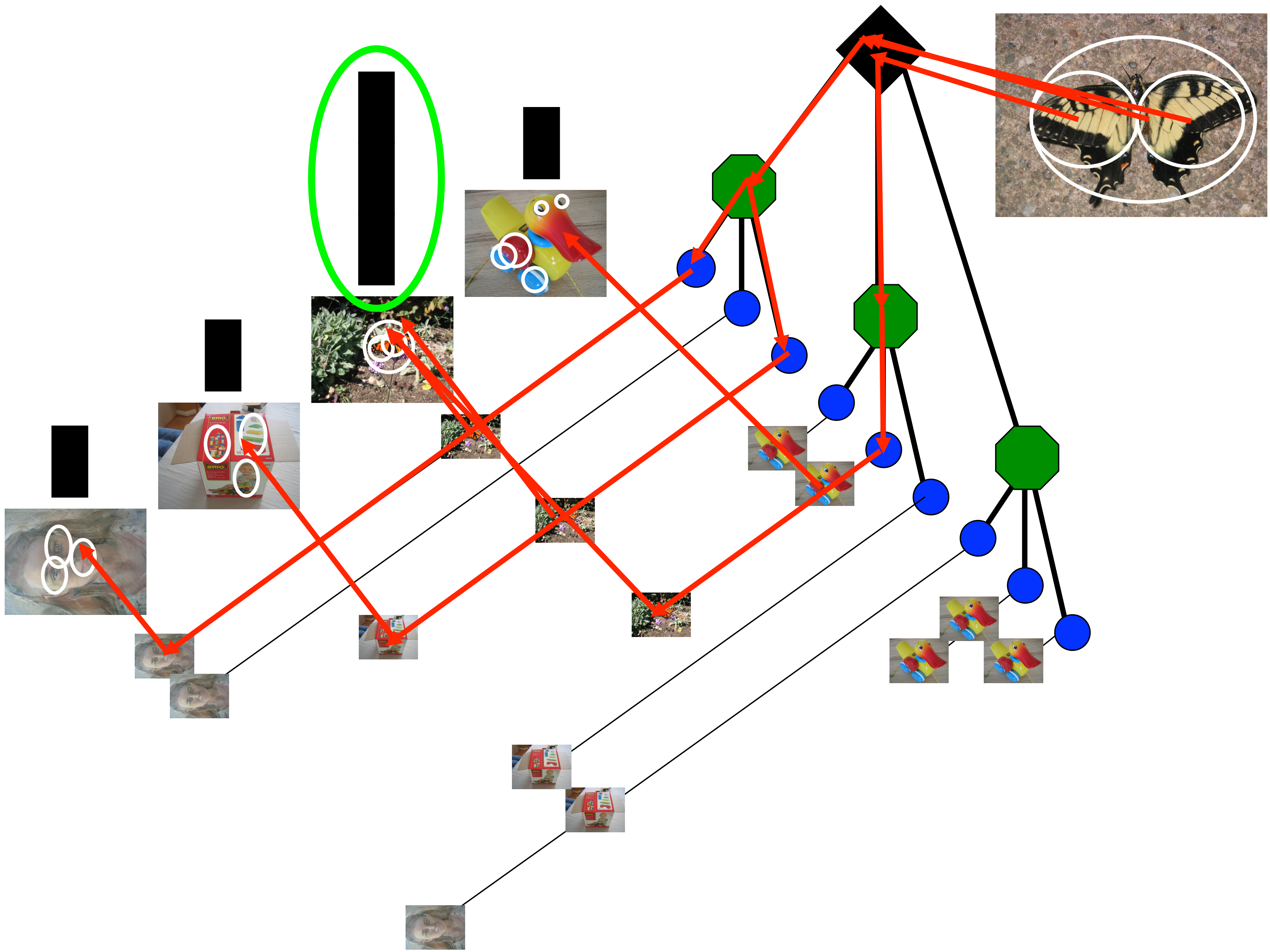
Slides from Nister & Stewenius 06



Slides from Nister & Stewenius 06



Slides from Nister & Stewenius 06



Slides from Nister & Stewenius 06

TF-IDF scoring

- TF-IDF term frequency – inverse document frequency
- Used in the information retrieval and text mining
- To evaluate how important is a word to document

- Importance depends on how many times the word appears in document – offset by number of occurrence of that word in the whole document corpus

TF-IDF scoring

- TF-IDF term frequency – inverse document frequency
- Number of occurrences of a word in a document / number of occurrences of all words in the document

$$\mathrm{tf}_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

$$|d : t_i \in d|$$

- Number of documents / number of documents where term appears

$$\mathrm{idf}_{i,j} = \log \frac{|D|}{|\{d : t_i \in d\}|}$$

$$|D|$$

- High weight of a word/term is when it has high frequency and low term document frequency

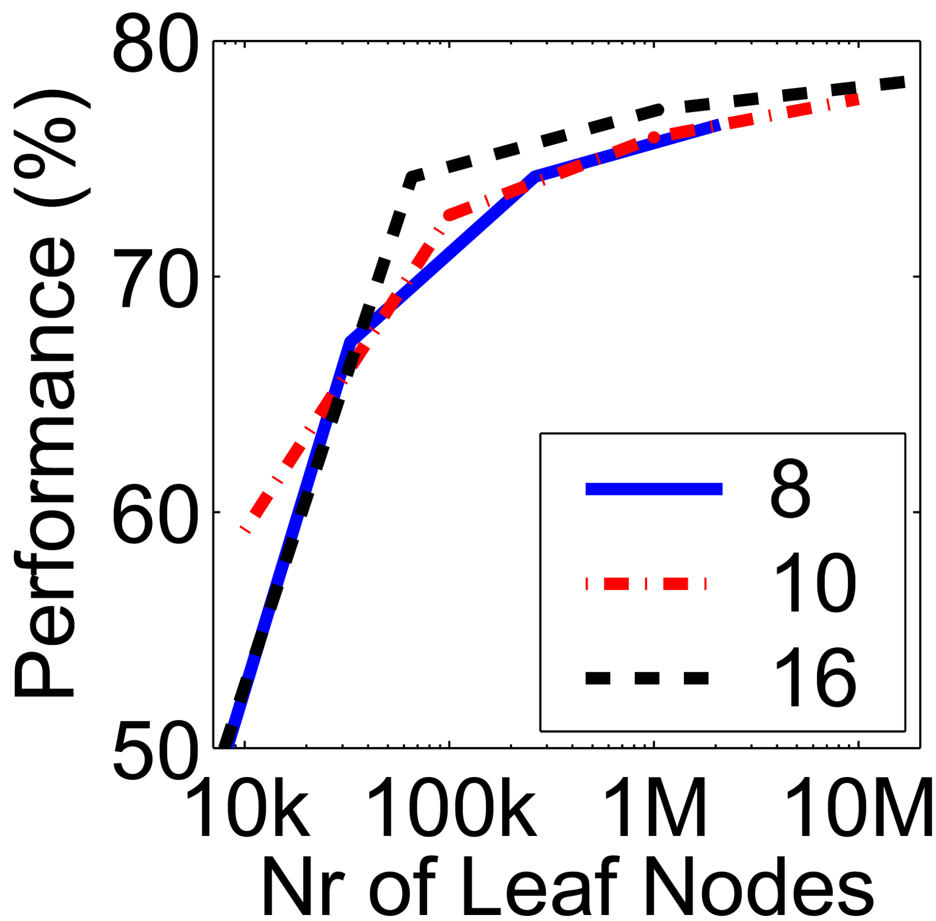
$$\mathrm{tfidf}_{i,j} = \mathrm{tf}_{i,j} \times \mathrm{idf}_i$$

Size Matters

Performance improves with the
Size of the database

Improves
Retrieval

Improves
Speed

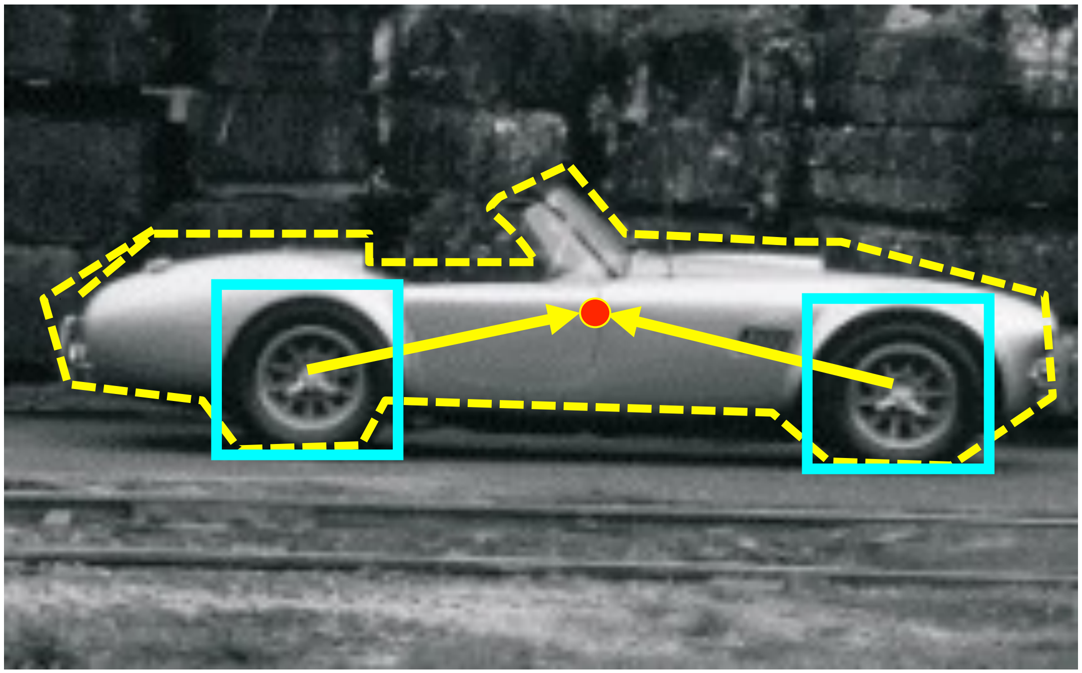


Here the results of particular object instance retrieval, database
Of ~ 40,000 objects, real-time performance

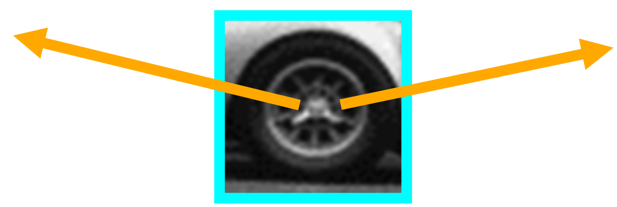
Slides from Nister & Stewenius 06

Implicit shape models

- Combining the edge based GHT style voting with appearance codebooks
- Visual codebook is used to index votes for object position



training image annotated with object localization info



visual codeword with displacement vectors

B. Leibe, A. Leonardis, and B. Schiele, Combined Object Categorization and Segmentation with an Implicit Shape Model, ECCV Workshop on Statistical Learning in Computer Vision 2004

Implicit shape models

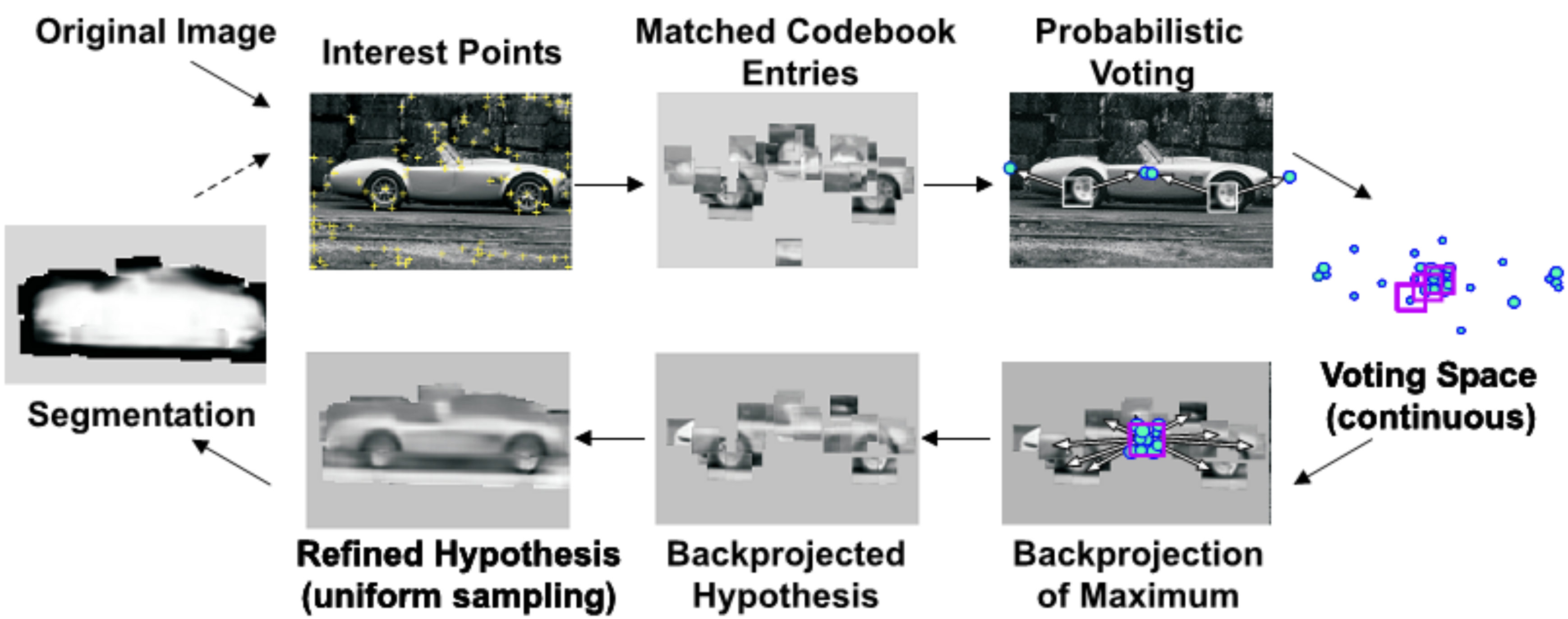
- Visual codebook is used to index votes for object position

test image

Idea Implicit Shape Model

- Faces rectangular templates – detection windows
- Does not generalize to more complex object with different

  shapes
- How to combine patch based – appearance based representations to incorporate notion of shape
- Combined Object Categorization and Segmentation with an Implicit Shape Model. Bastian Leibe, Ales Leonardis, and

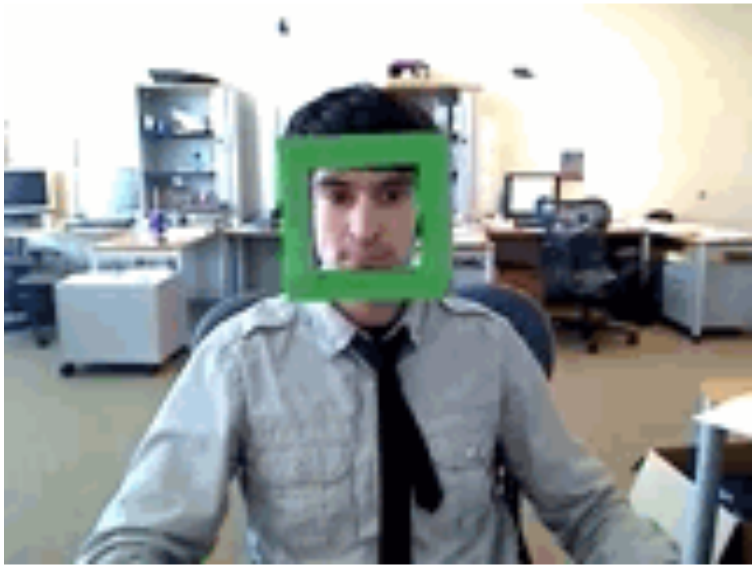


- Object Category Detection

Face detection

- Basic idea: slide a window across image and evaluate a face model at every location

Face detection



Behold a state-of-the-art face detector!
(Courtesy Boris Babenko)

Challenges of face detection

- Sliding window detector must evaluate tens of thousands of location/scale combinations
- Faces are rare: 0–10 per image
  - For computational efficiency, we should try to spend as little time as possible on the non-face windows
  - A megapixel image has ~10^6 pixels and a comparable number of candidate face locations
  - To avoid having a false positive in every image image, our false positive rate has to be less than 10^{-6}

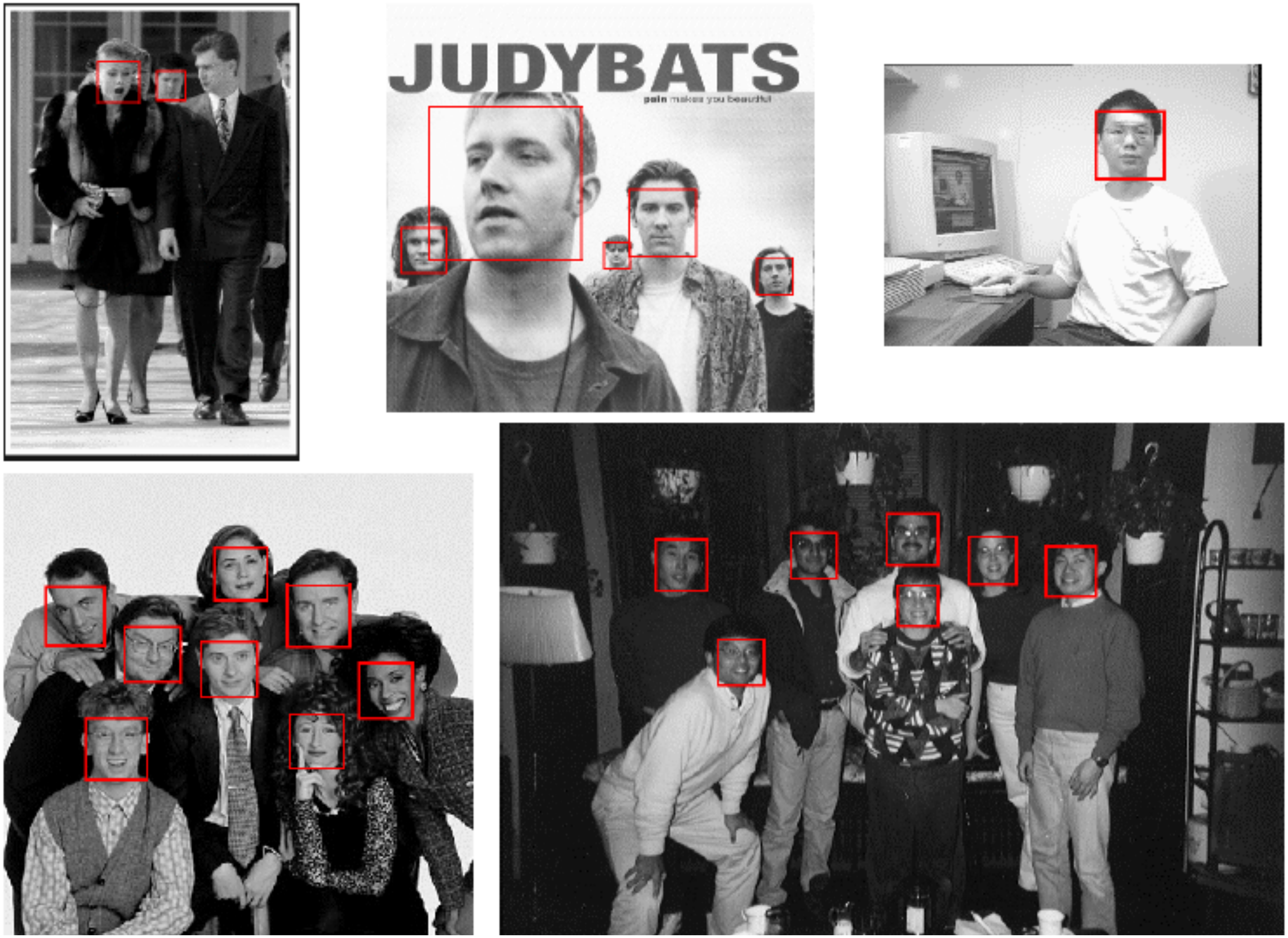
The Viola/Jones Face Detector

- A seminal approach to real-time object detection
- Training is slow, but detection is very fast
- Key ideas
  - *Integral images* for fast feature evaluation
  - *Boosting* for feature selection
  - *Attentional cascade* for fast rejection of non-face windows

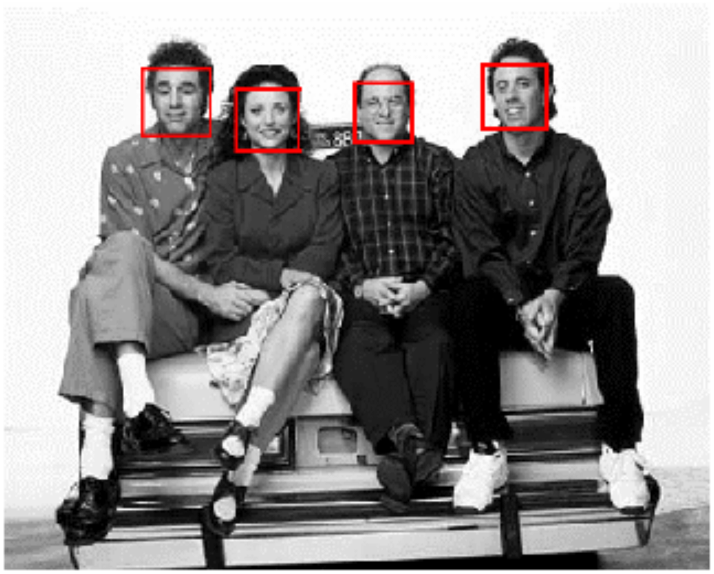
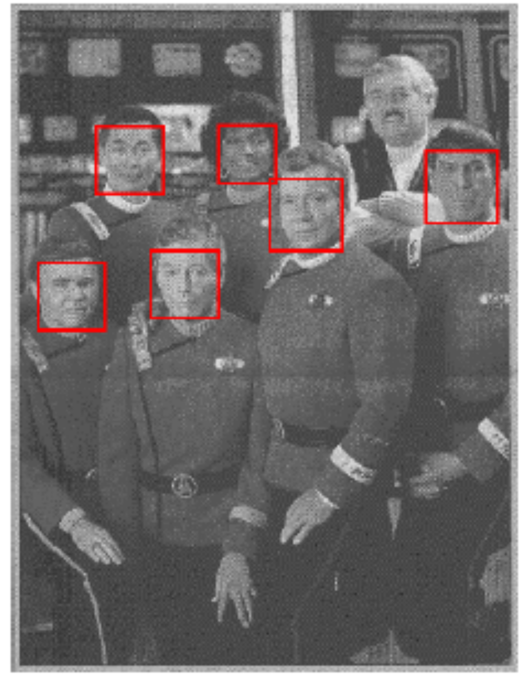
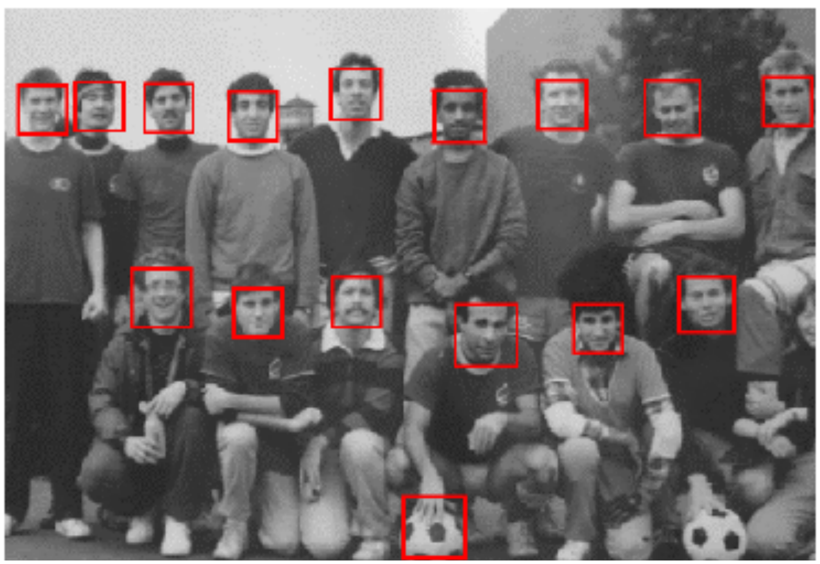
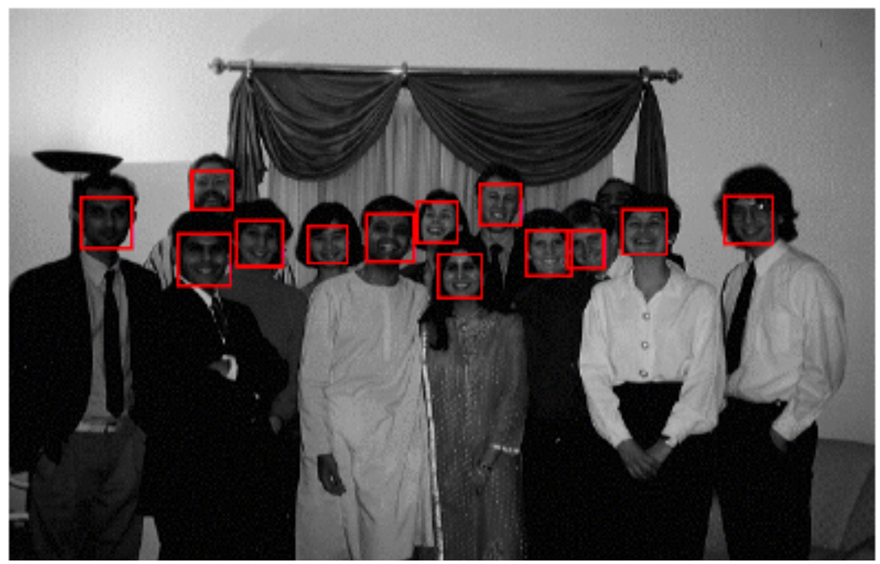
P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features.* CVPR 2001.

P. Viola and M. Jones. *Robust real-time face detection.* IJCV 57(2), 2004.

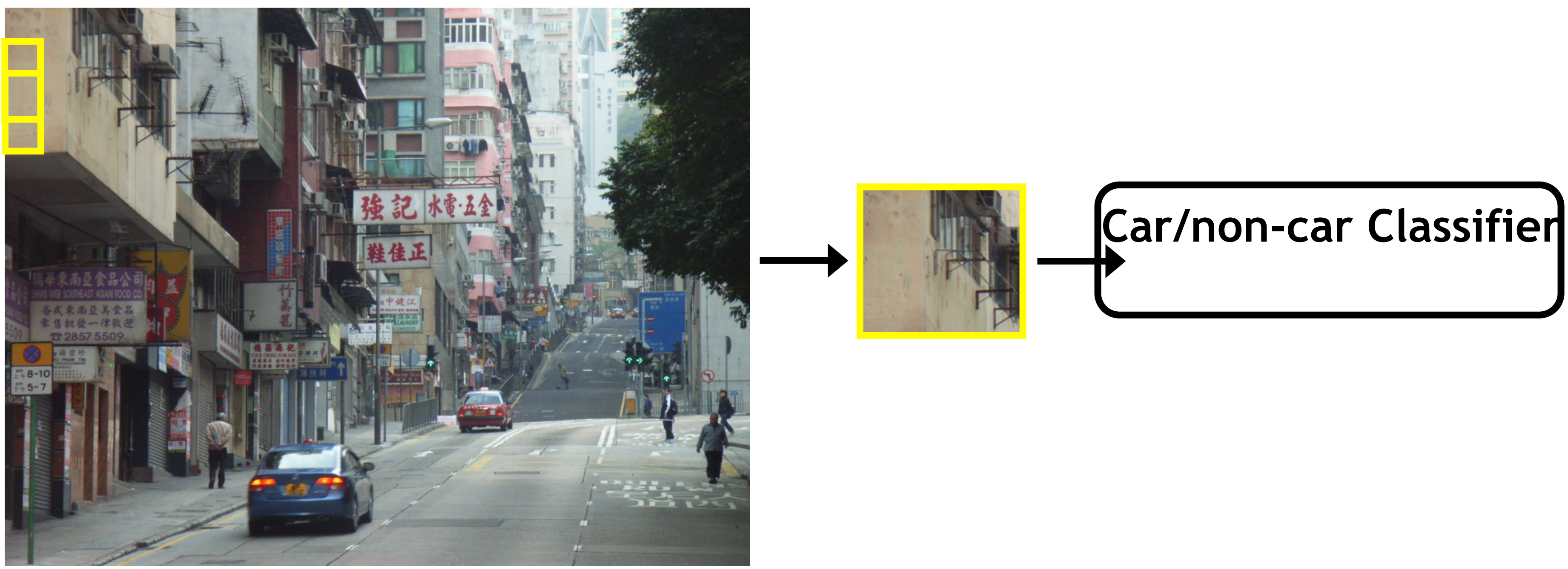
Viola-Jones Face Detector: Results



Viola-Jones Face Detector: Results

Window-based models
Generating and scoring candidates



Window-based object detection: recap

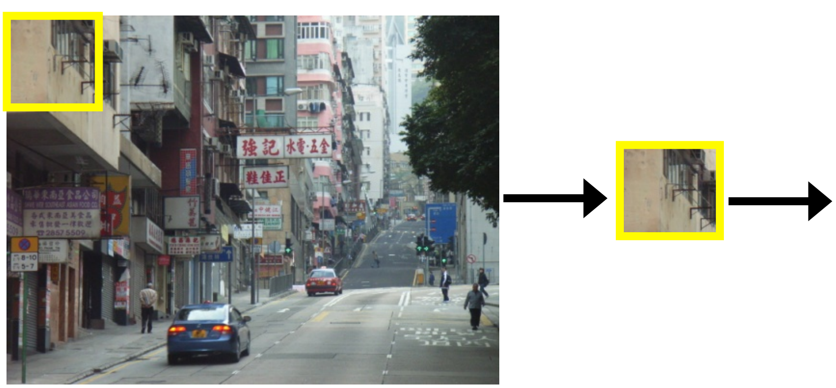
Training:

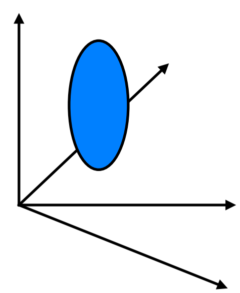
1. Obtain training data
2. Define features
3. Define classifier

Given new image:

1. Slide window
2. Score by classifier

Training examples





Feature extraction

Car/no car classifier

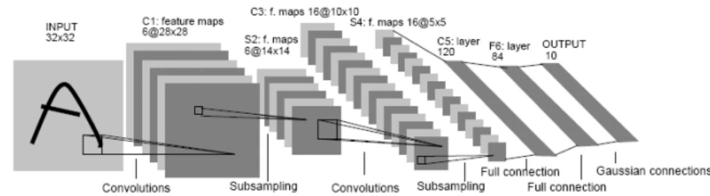
Discriminative classifier construction

Nearest neighbor

10^6 examples

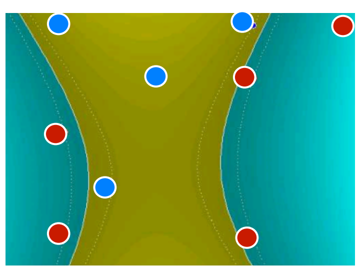
Shakhnarovich, Viola, Darrell 2003
Berg, Berg, Malik 2005...

Neural networks



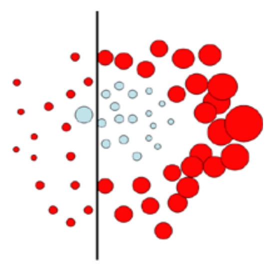
LeCun, Bottou, Bengio, Haffner 1998
Rowley, Baluja, Kanade 1998
…

Support Vector Machines



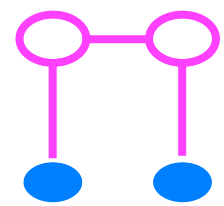
Guyon, Vapnik
Heisele, Serre, Poggio, 2001,…

Boosting



Viola, Jones 2001, Torralba et al. 2004, Opelt et al. 2006,…

Conditional Random Fields

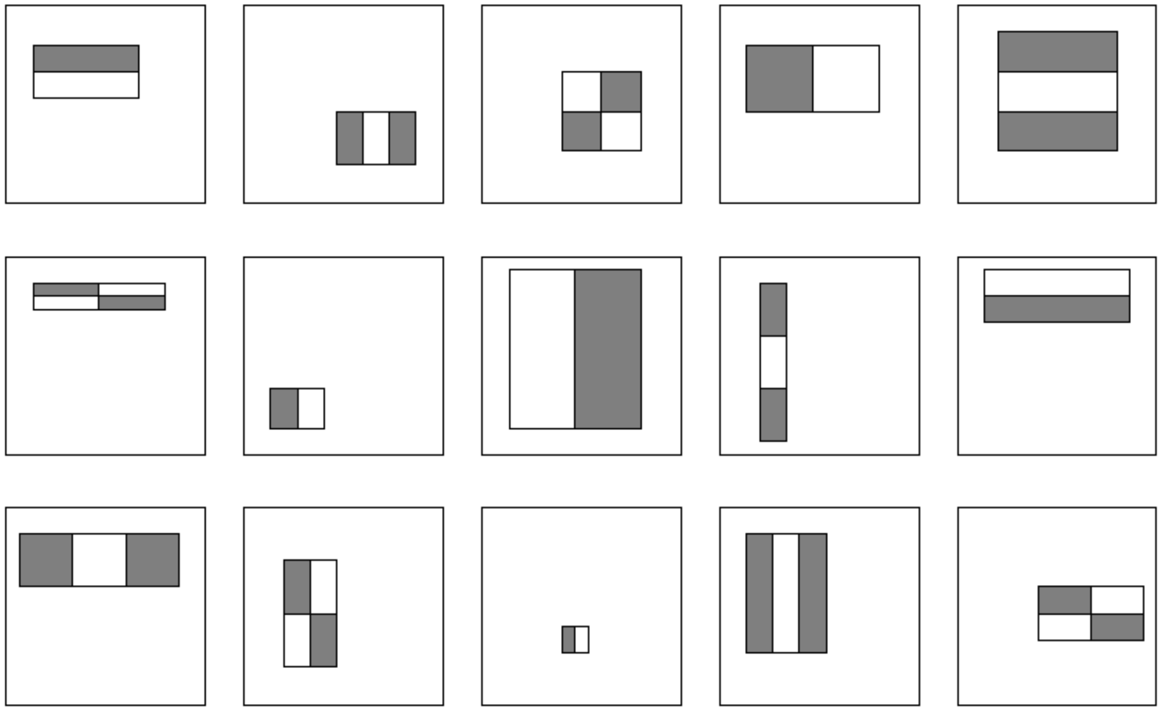


McCallum, Freitag, Pereira 2000; Kumar, Hebert 2003
…

Slide adapted from Antonio Torralba

Viola-Jones Face Detector: Results

Viola-Jones detector: features



Considering all possible filter parameters: position, scale, and type:

180,000+ possible features associated with each 24 x 24 window

Which subset of these features should we use to determine if a window has a face?

Use AdaBoost both to select the informative features and to form the classifier

Kristen Grauman

Boosting for face detection

- Define weak learners based on rectangle features

value of rectangle feature

$$h_t(x) = \begin{cases} 1 & \text{if } p_t f_t(x) > p_t \theta_t \\ 0 & \text{otherwise} \end{cases}$$

window

parity

threshold

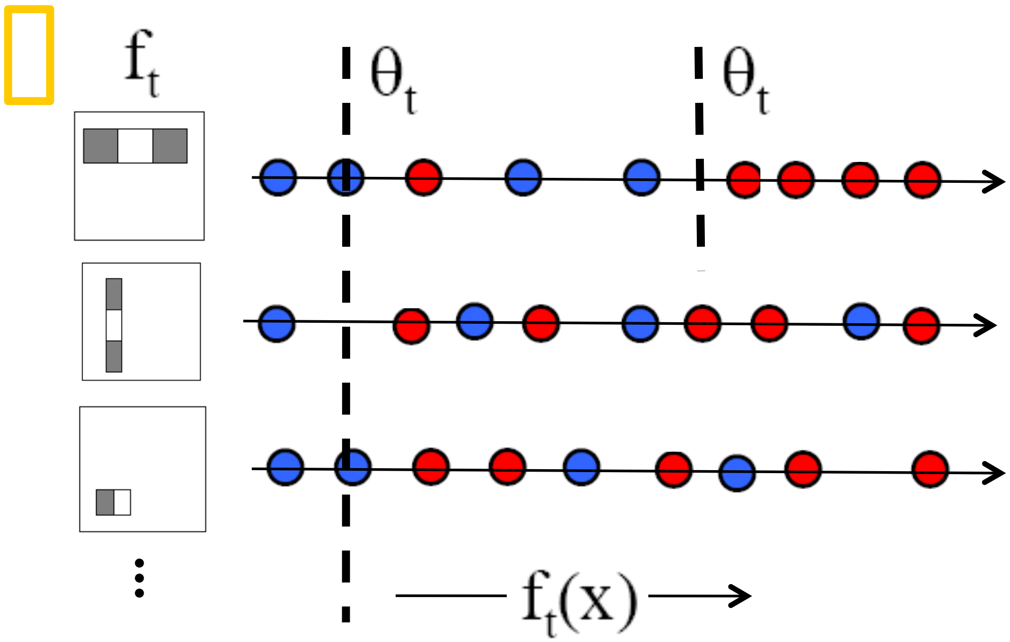
- For each round of boosting:
  Evaluate each rectangle filter on each example
  Select best filter/threshold combination based on weighted training error reweight examples

Boosting for face detection

- Define weak learners based on rectangle features
- For each round of boosting:
  - Evaluate each rectangle filter on each example
  - Select best threshold for each filter
  - Select best filter/threshold combination
  - Reweight examples
- Computational complexity of learning: $O(MNK)$
  - M rounds, N examples, K features

Viola-Jones detector: AdaBoost

- Want to select the single rectangle feature and threshold that best separates positive (faces) and negative (non-faces) training examples, in terms of *weighted* error.



Outputs of a possible rectangle feature on faces and non-faces.

Resulting weak classifier:

$$h_t(x) = \begin{cases} +1 & \text{if } f_t(x) > \theta_t \\ -1 & \text{otherwise} \end{cases}$$

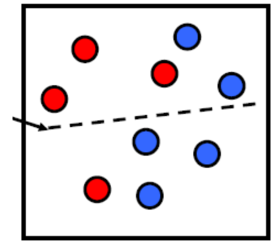
For next round, reweight the examples according to errors, choose another filter/threshold combo.

Kristen Grauman

AdaBoost Algorithm

- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For $t = 1, \ldots, T$:
  1. Normalize the weights,
     $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
     so that w_t is a probability distribution.
  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i \, |h_j(x_i) - y_i|$.
  3. Choose the classifier, h_t, with the lowest error ϵ_t.
  4. Update the weights:
     $$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$
     where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- The final strong classifier is:
  $$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$
  where $\alpha_t = \log \frac{1}{\beta_t}$

Start with uniform weights on training examples



$\{x_1, \ldots x_n\}$

For T rounds

Evaluate weighted error for each feature, pick best.

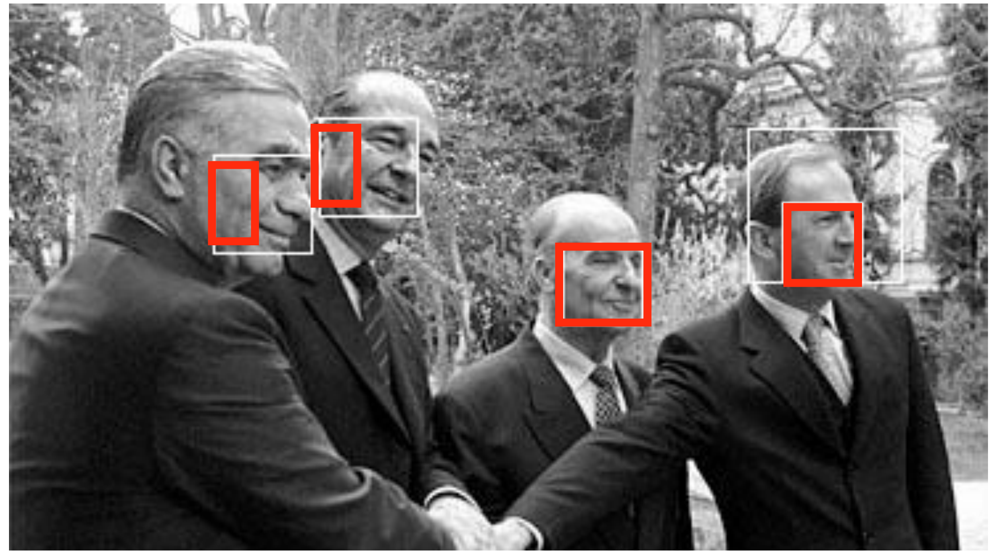
Re-weight the examples:
Incorrectly classified -> more weight
Correctly classified -> less weight

Final classifier is combination of the weak ones, weighted according to error they had.

- Even if the filters are fast to compute, each new image has a lot of possible windows to search.
- How to make the detection more efficient?

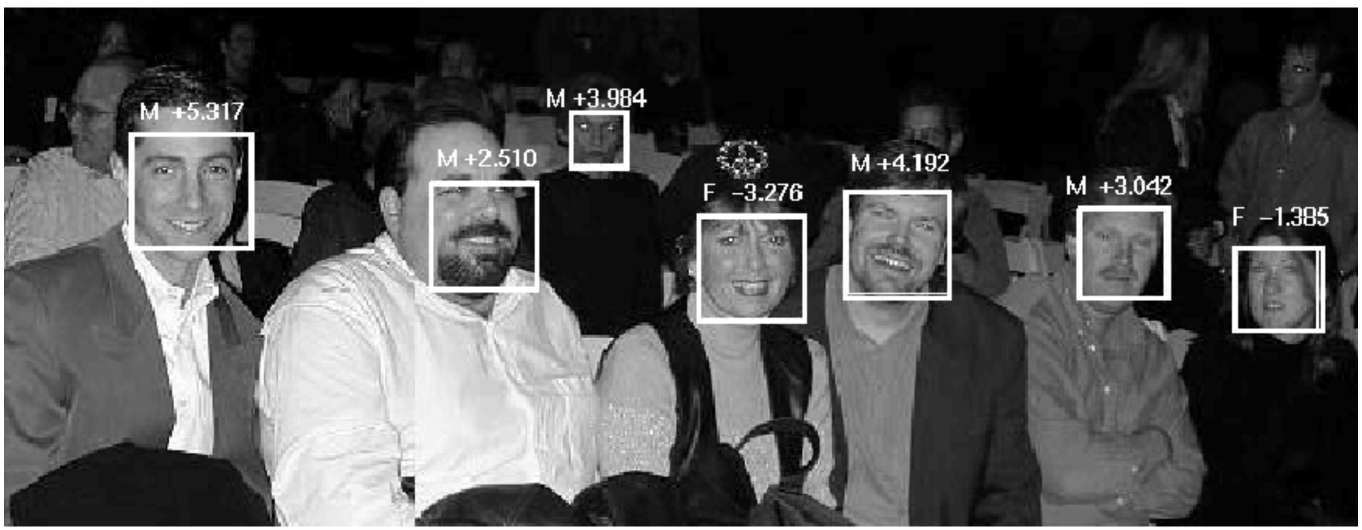
Solving other “Face” Tasks

Facial Feature Localization



Profile Detection

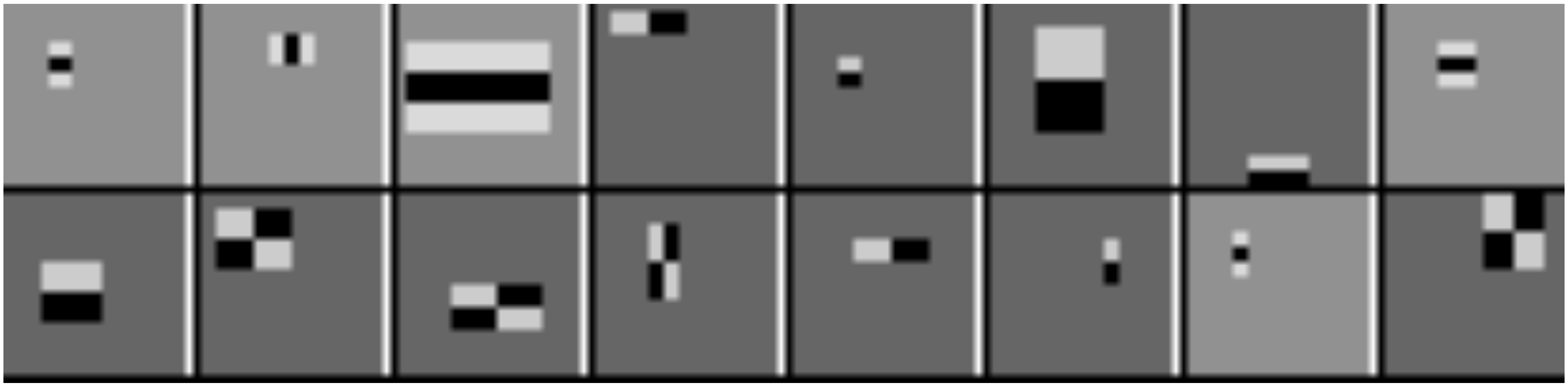
Demographic Analysis



 Slide credit: Frank Dellaert, Paul Viola, Foryth&Ponce

Face Localization Features

- Learned features reflect the task

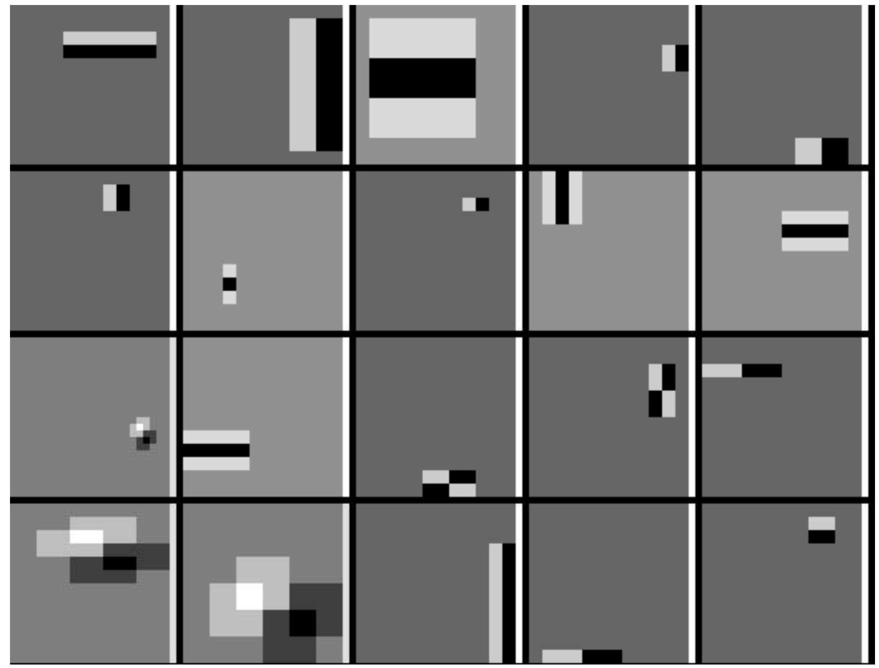


Slide credit: Frank Dellaert, Paul Viola, Forsyth&Ponce

Face Profile Detection

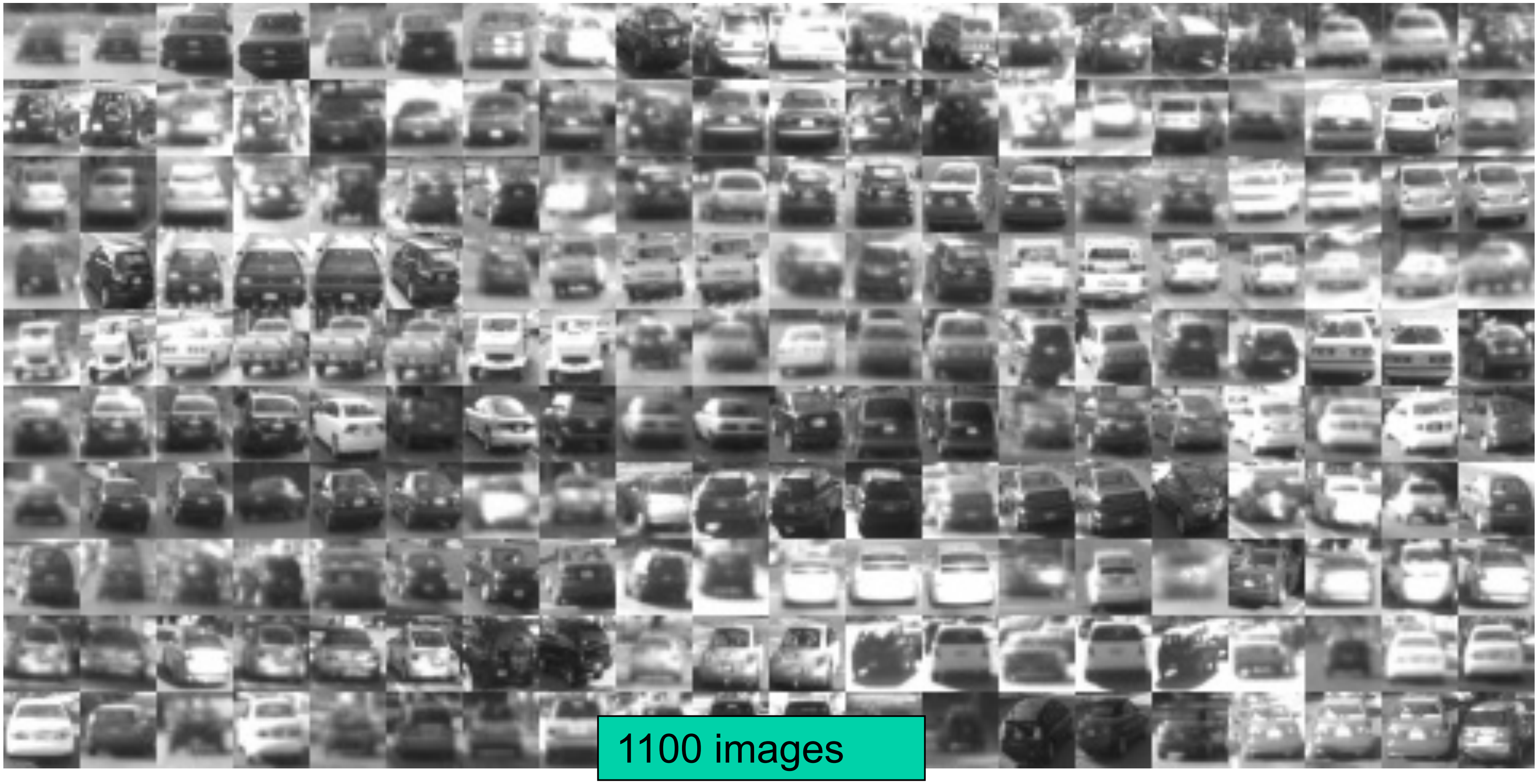
Slide credit: Frank Dellaert, Paul Viola, Foryth&Ponce

Face Profile Features

Finding Cars (DARPA Urban Challenge)

- Hand-labeled images of generic car rear-ends
- Training time: ~5 hours, offline



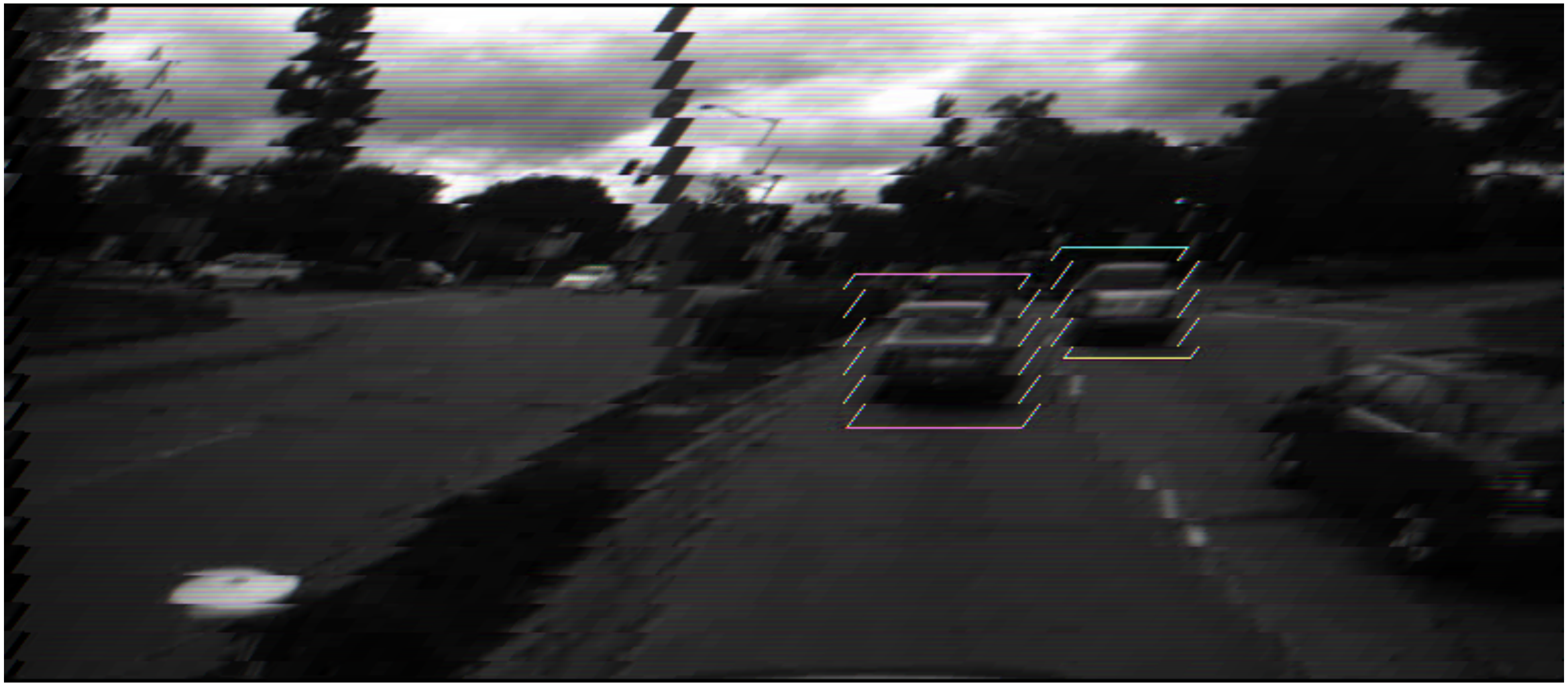
Credit: Hendrik Dahlkamp

Generating even more examples

- Generic classifier finds all cars in recorded video.
- Compute offline and store in database

Credit: Hendrik Dahlkamp

Results - Video



Pedestrian Detection: HOG Feature

- Positive data – 1208 positive window examples

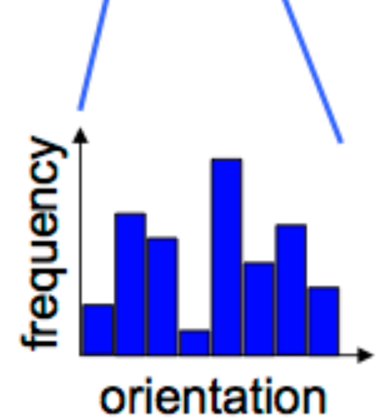
- Negative data – 1218 negative window examples (initially)

Slides from Andrew Zisserman

Pedestrian Detection: HOG Feature

HOG: Histogram of Gradients

- tile window into 8 x 8 pixel cells
- each cell represented by HOG



Feature vector dimension = 16 x 8 (for tiling) x 8 (orientations) = 1024

Dalal & Triggs, CVPR 2005

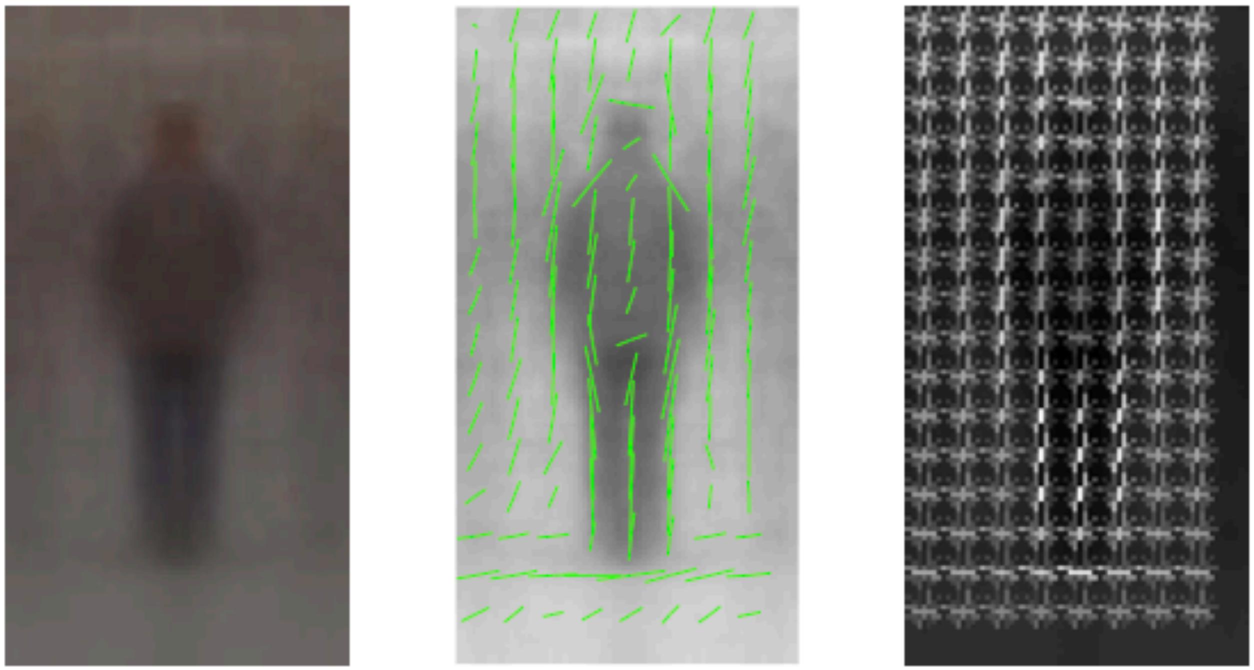
Slides from Andrew Zisserman

Pedestrian Detection: HOG Feature

Slides from Andrew Zisserman

Pedestrian Detection: HOG Feature

Averaged examples

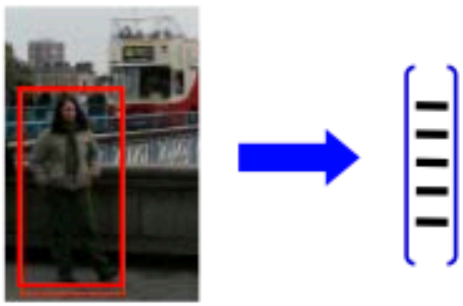


Slides from Andrew Zisserman

Algorithm

Training (Learning)

- Represent each example window by a HOG feature vector



$$\mathbf{x}_i \in \mathbb{R}^d, \text{ with } d = 1024$$

- Train a SVM classifier

Testing (Detection)

- Sliding window classifier

$$f(x) = \mathbf{w}^\top \mathbf{x} + b$$

Slides from Andrew Zisserman

Model training using SVM

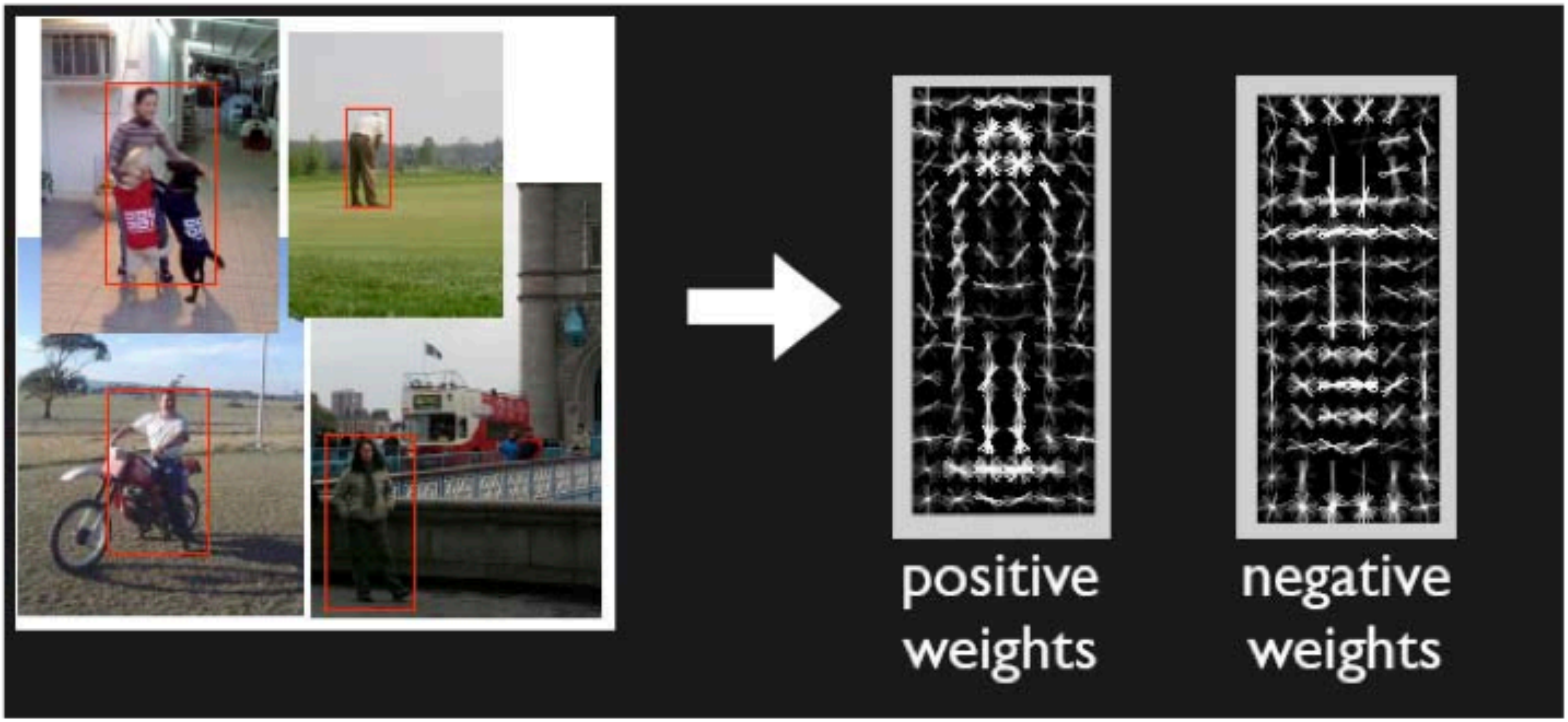
- Given $\left\{ \mathbf{x}_i \in \mathrm{R}^d, y_i \in \{0,1\} \right\}$
- Find
$$f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b$$
- To minimize
$$\min_{\mathbf{w},b} \|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \mathrm{error}\left(y_i f(\mathbf{x}_i)\right)$$
$$\mathrm{error}(z) = \max(0, 1 - z)$$

Result

Dalal and Triggs, CVPR 2005

Learned model

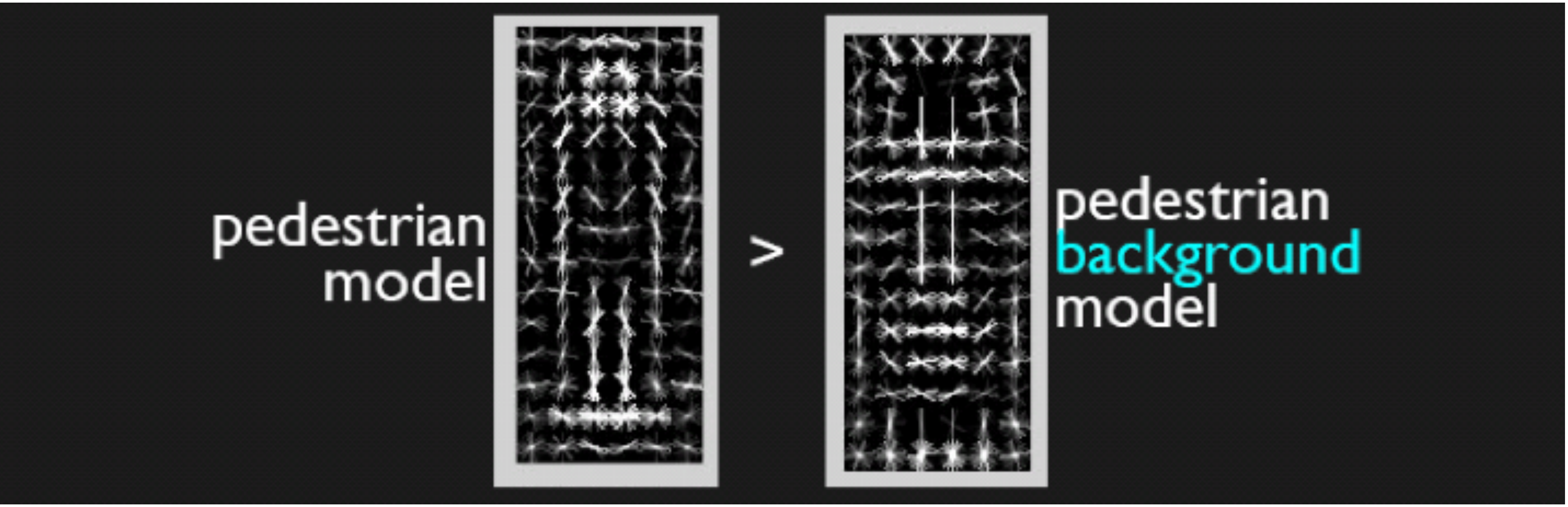
$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$



Slides from Deva Ramanan

Meaning of negative weights

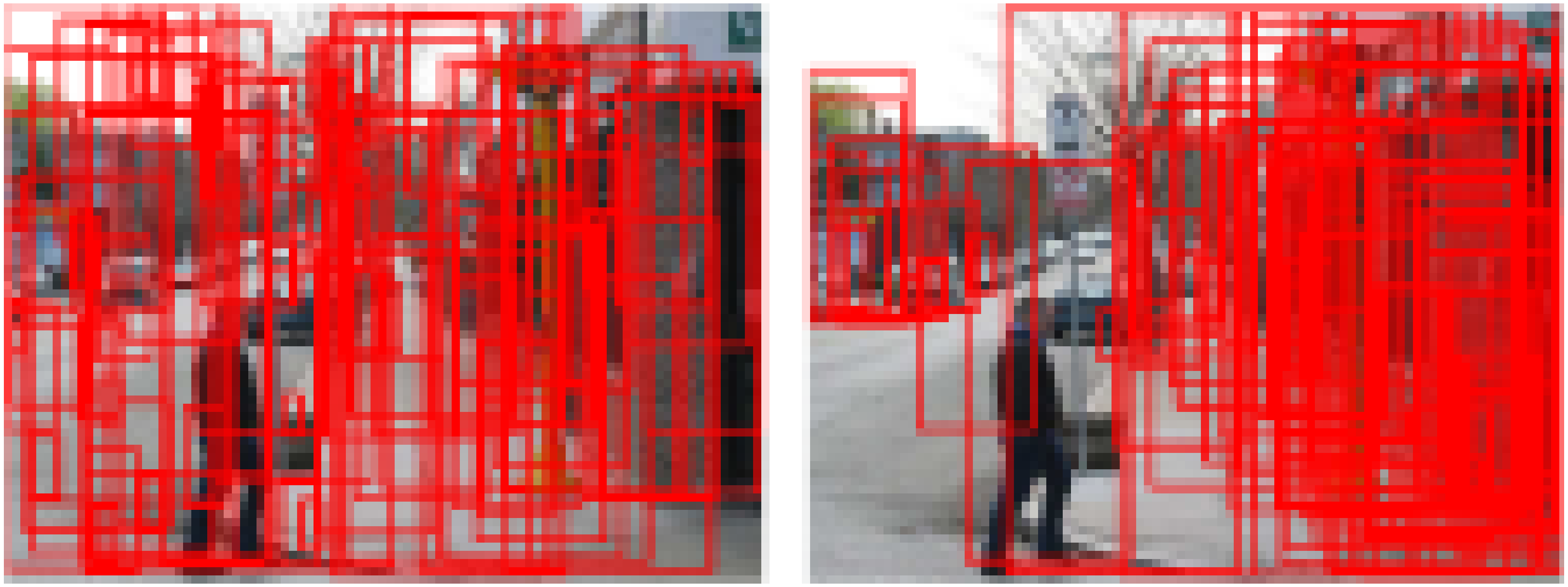
$wx > -b$
$(w_+ - w_-)x > -b$
$w_+x - w_-x > -b$



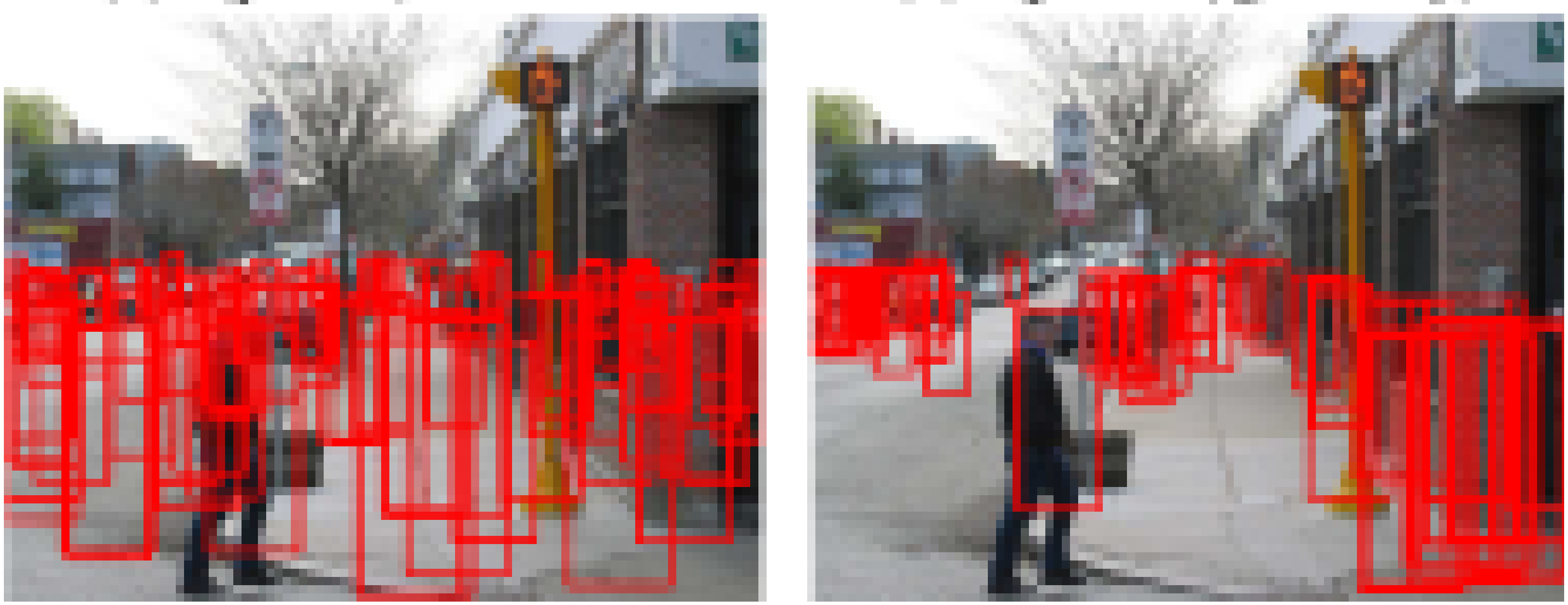
Complete model should compete pedestrian/pillar/doorway

Slides from Deva Ramanan

Context



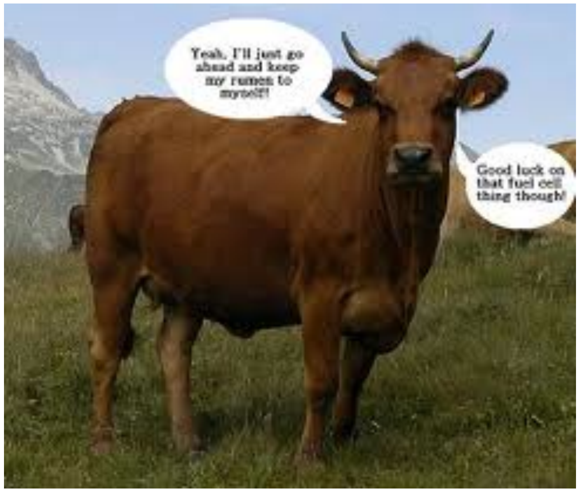
(b) P(person) = uniform
(d) P(person | geometry)



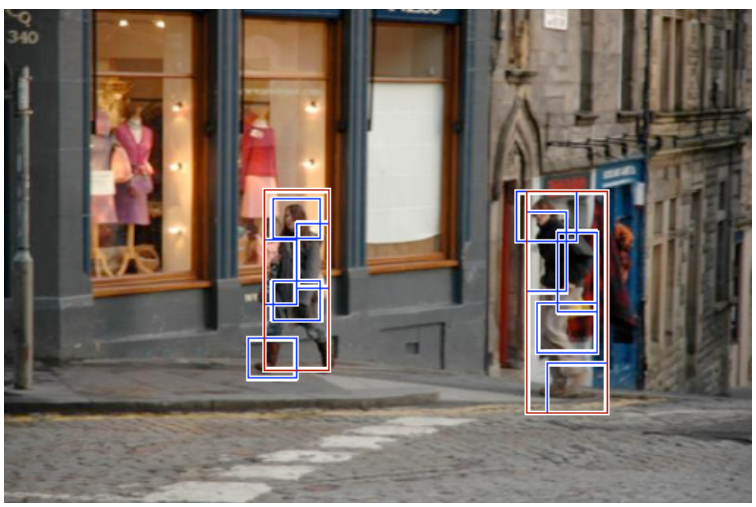
(f) P(person | viewpoint)
(g) P(person|viewpoint,geometry)

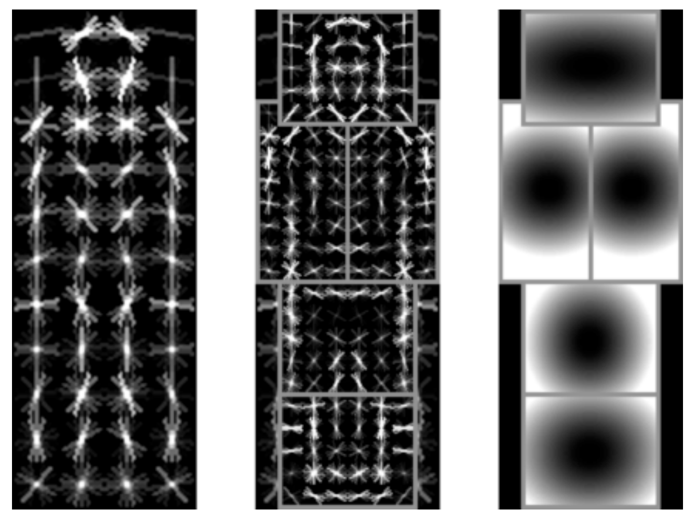
Hoiem, Efros, Herbert, 2006

More difficult cases



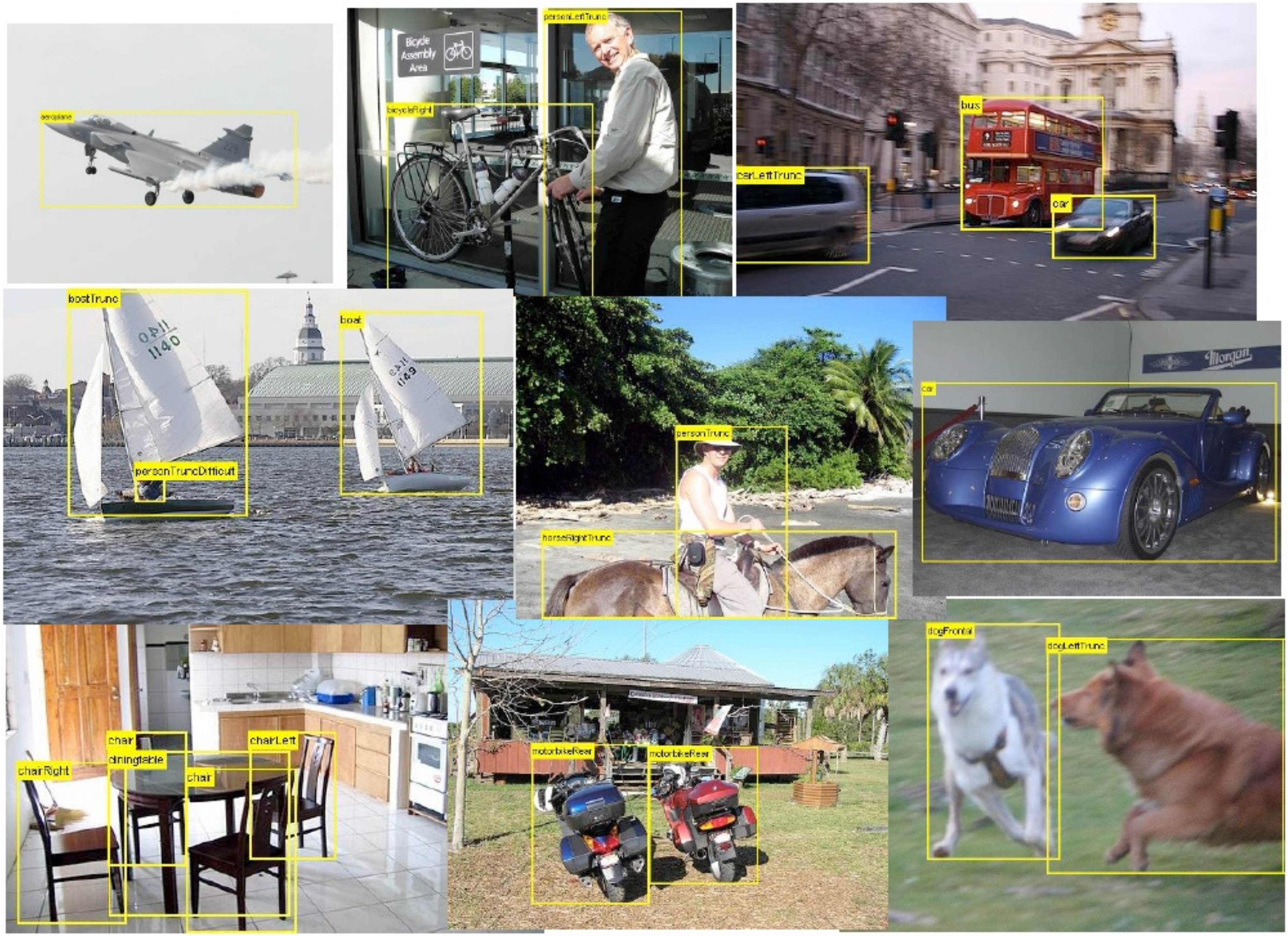
More sliding window detection: Discriminative part-based models



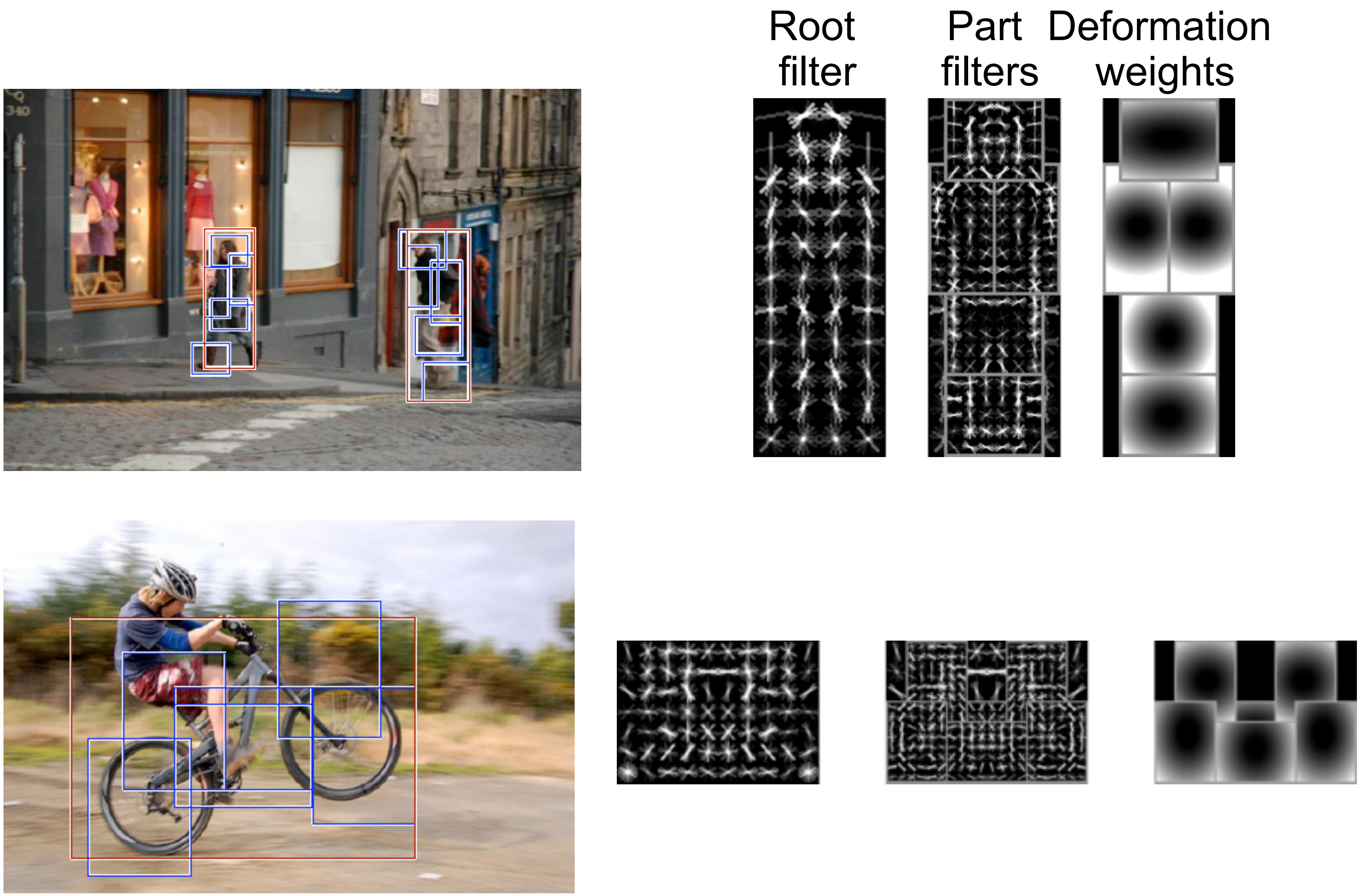


Many slides based on P. Felzenszwalb

Challenge: Generic object detection



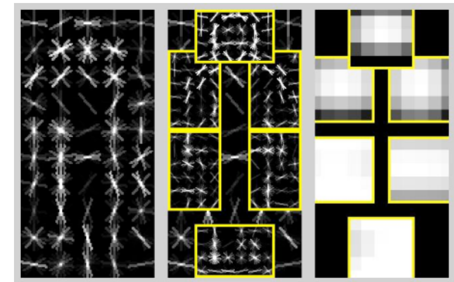
Discriminative part-based models



P. Felzenszwalb, R. Girshick, D. McAllester, D. Ramanan, [Object Detection with Discriminatively Trained Part Based Models](), PAMI 32(9), 2010

Object hypothesis

- Multiscale model: the resolution of part filters is twice the resolution of the root



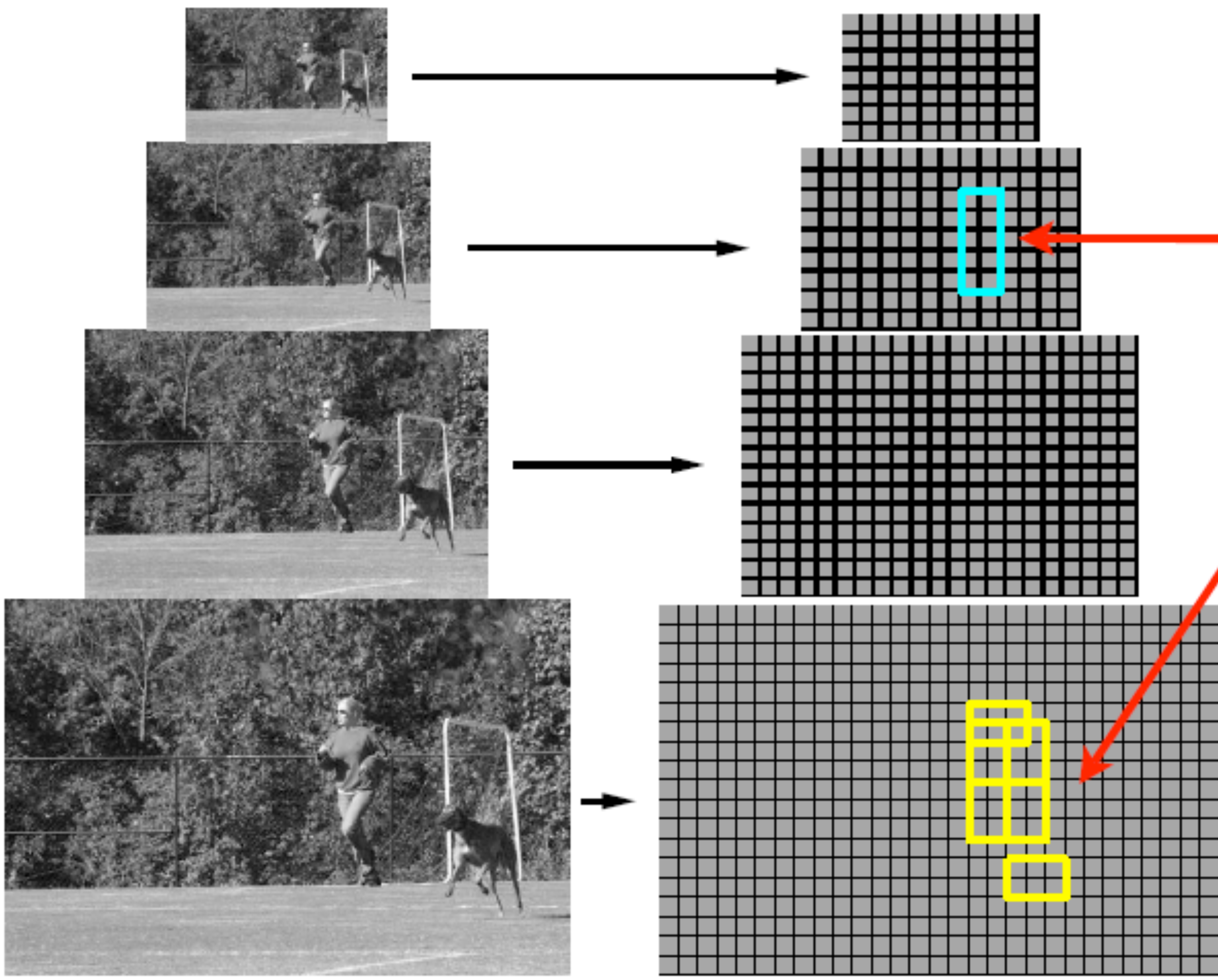


Image pyramid

HOG feature pyramid

$z = (p_0, \ldots, p_n)$

p_0 : location of root

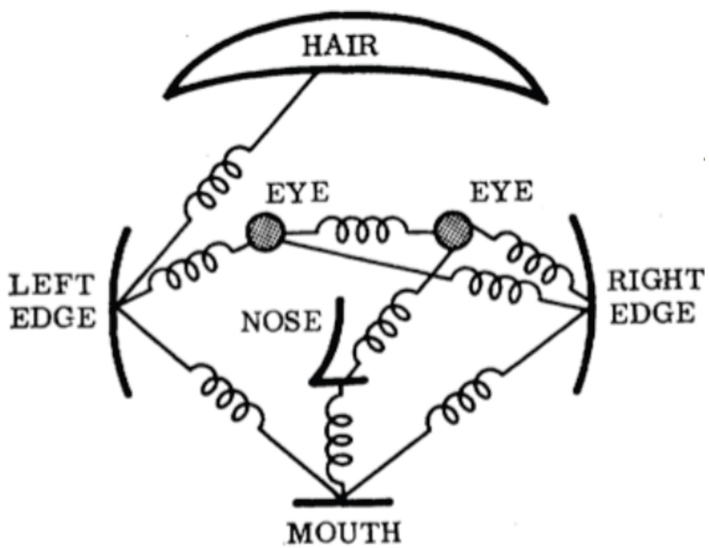
$p_1, \ldots, p_n$: location of parts

Score is sum of filter scores minus deformation costs

Score of the filter : inner products between the filter and features

Part-based representation

- Objects are decomposed into parts and spatial relations among parts
- E.g. Face model by Fischler and Elschlager ‘73



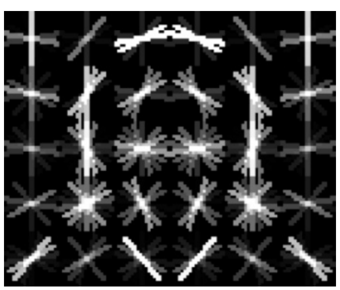
Detection

- Define the score of each root filter location as the score given the best part placements:

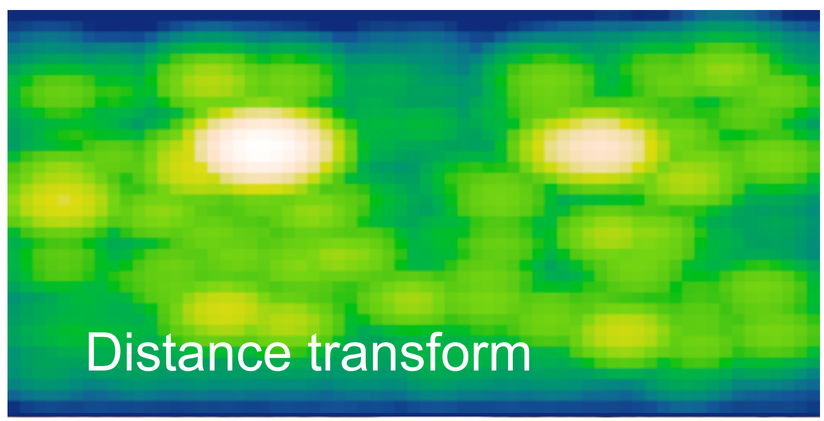
$$score(p_0) = \max_{p_1,\dots,p_n} score(p_0,\dots,p_n)$$

- Efficient computation: *generalized distance transforms*
  - For each “default” part location, find the best-scoring displacement

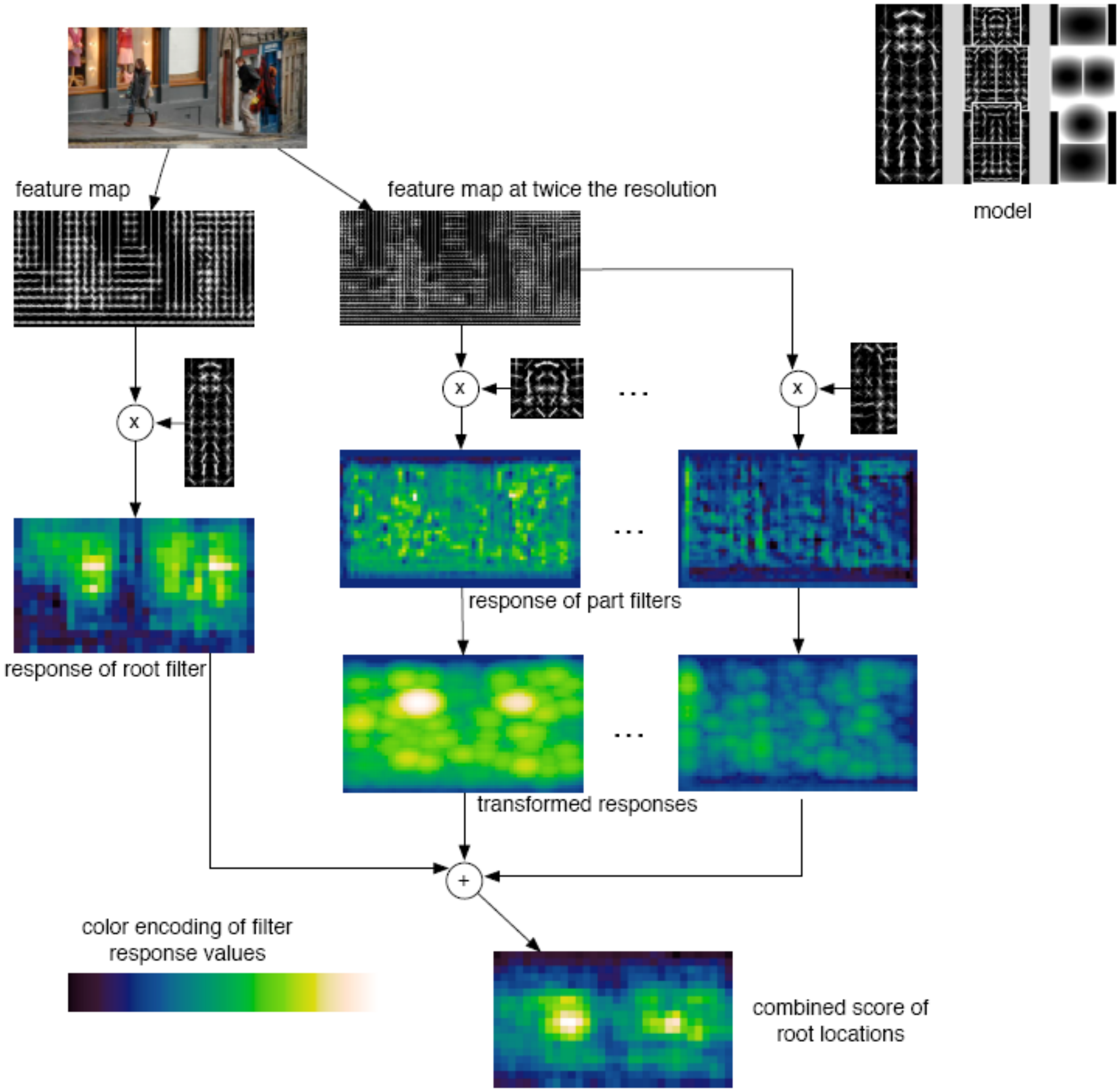
$$R_i(x,y) = \max_{dx,dy}\Big(F_i \cdot H(x+dx, y+dy) - D_i \cdot (dx, dy, dx^2, dy^2)\Big)$$



Head filter



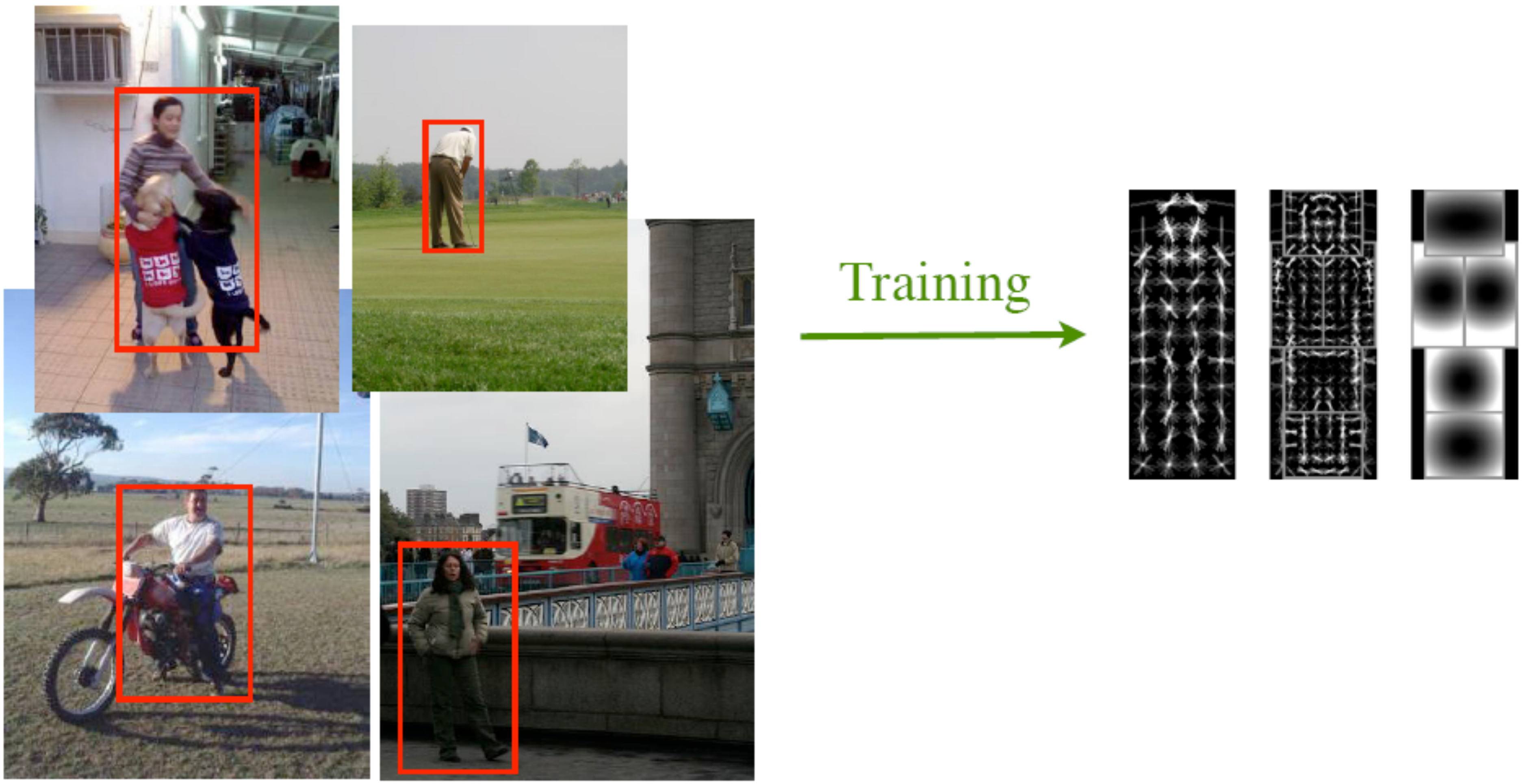
Detection



feature map
feature map at twice the resolution
model
x
x
x
...
...
response of part filters
response of root filter
...
transformed responses
+
color encoding of filter
response values
combined score of
root locations

Training

- Training data consists of images with labeled bounding boxes



Training

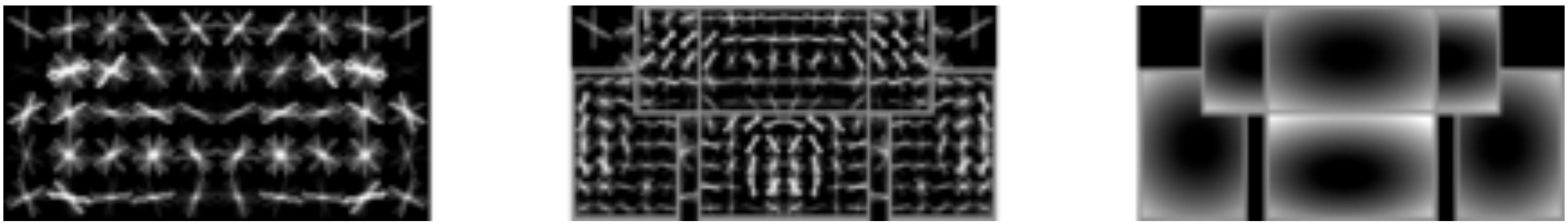
- The classifier has the form

$$f(x) = \max_z w \cdot H(x, z)$$

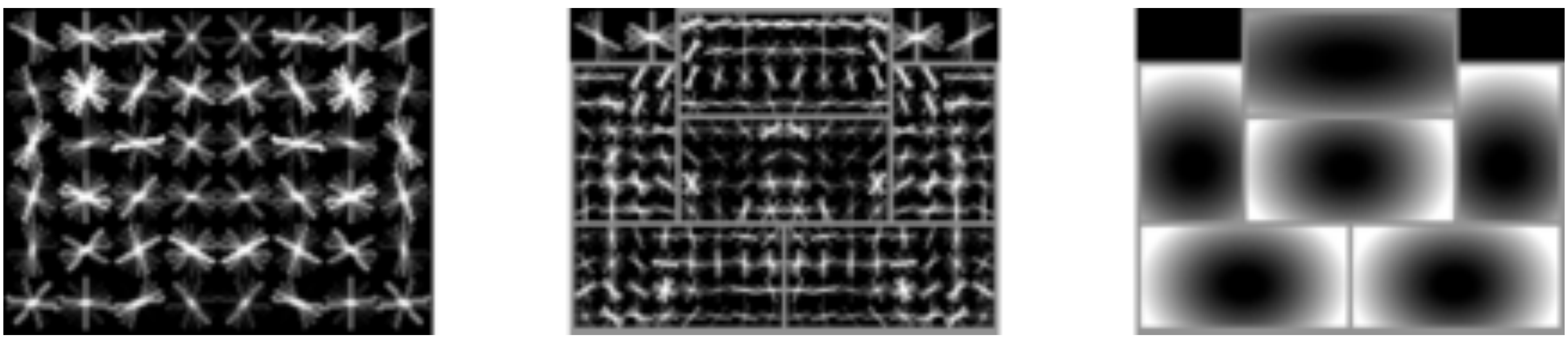
- *w* are model parameters (filters and deformation parameters, *z* are *latent* hypotheses)
- *x* is detection window, *z* are features and filter placements
- **Latent SVM** training:
  - Initialize *w* and iterate:
    - Fix *w* and find the best *z* for each training example (detection)
    - Fix *z* and solve for *w* (standard SVM training)
- Issue: too many negative examples
  - Do “data mining” to find “hard” negatives

Car model

Component 1

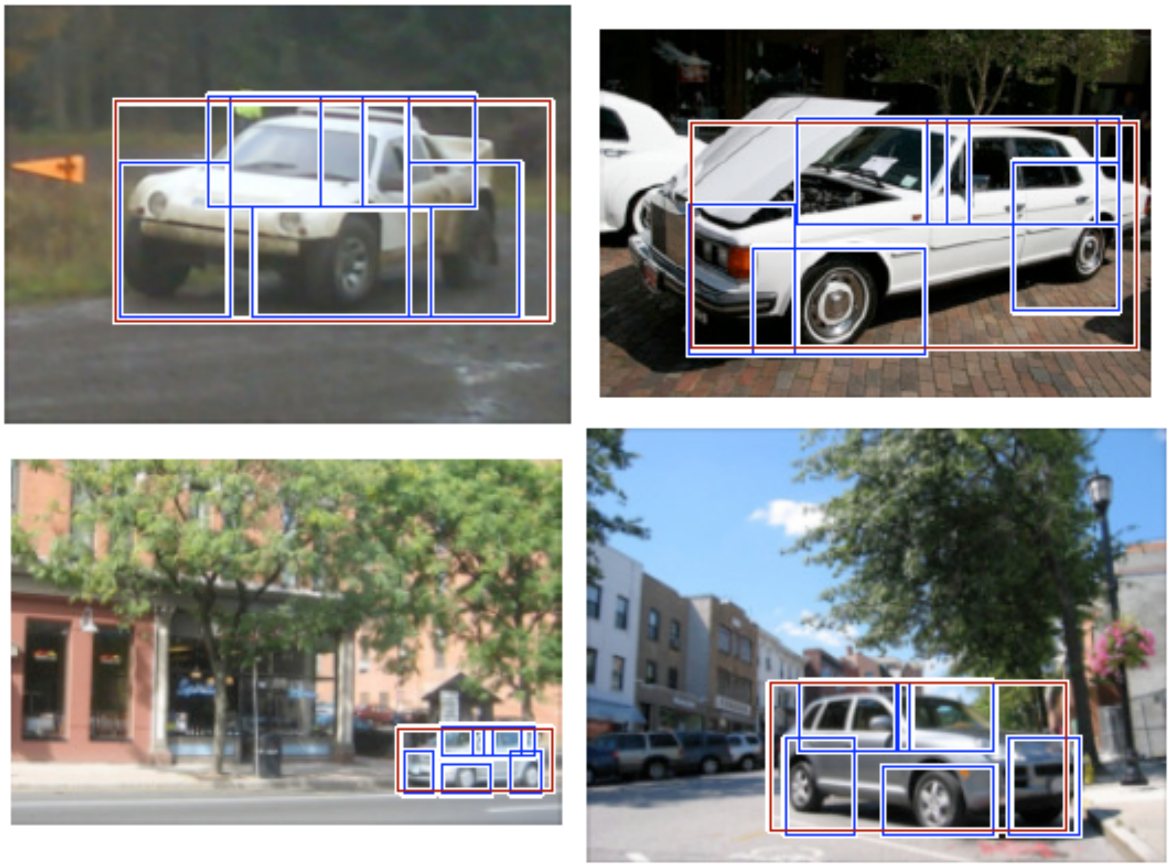


Component 2

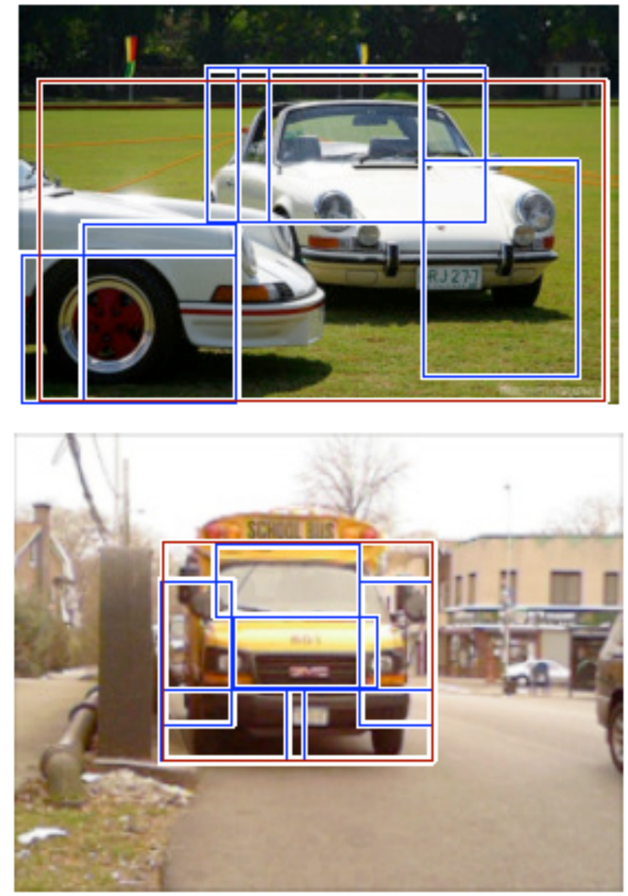


Car detections

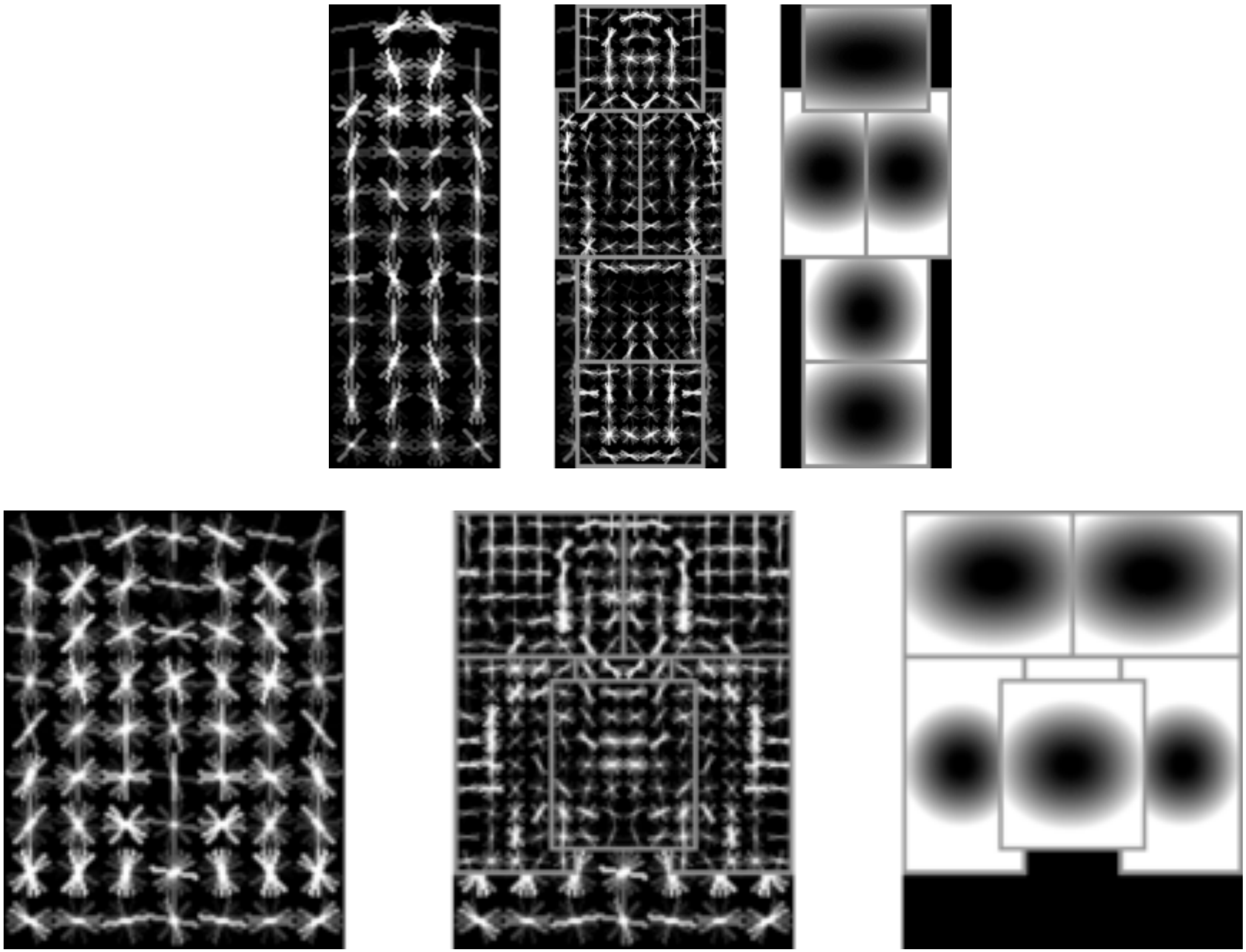
high scoring true positives



high scoring false positives

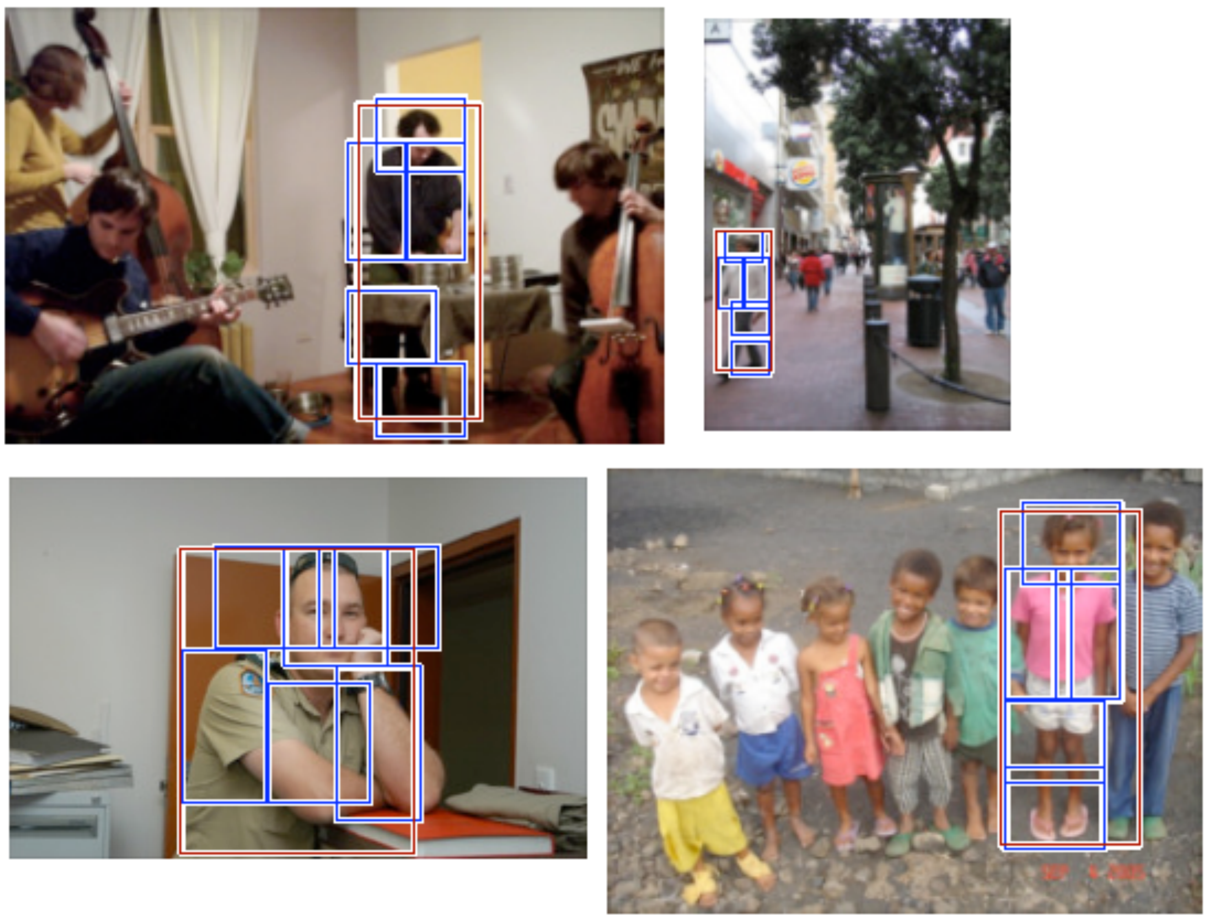


Person model

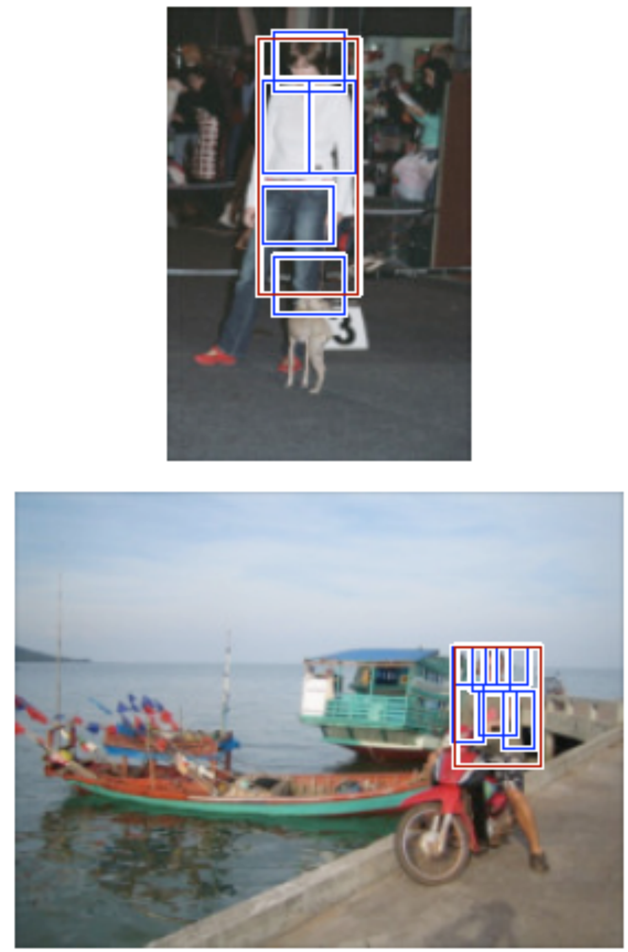


Person detections

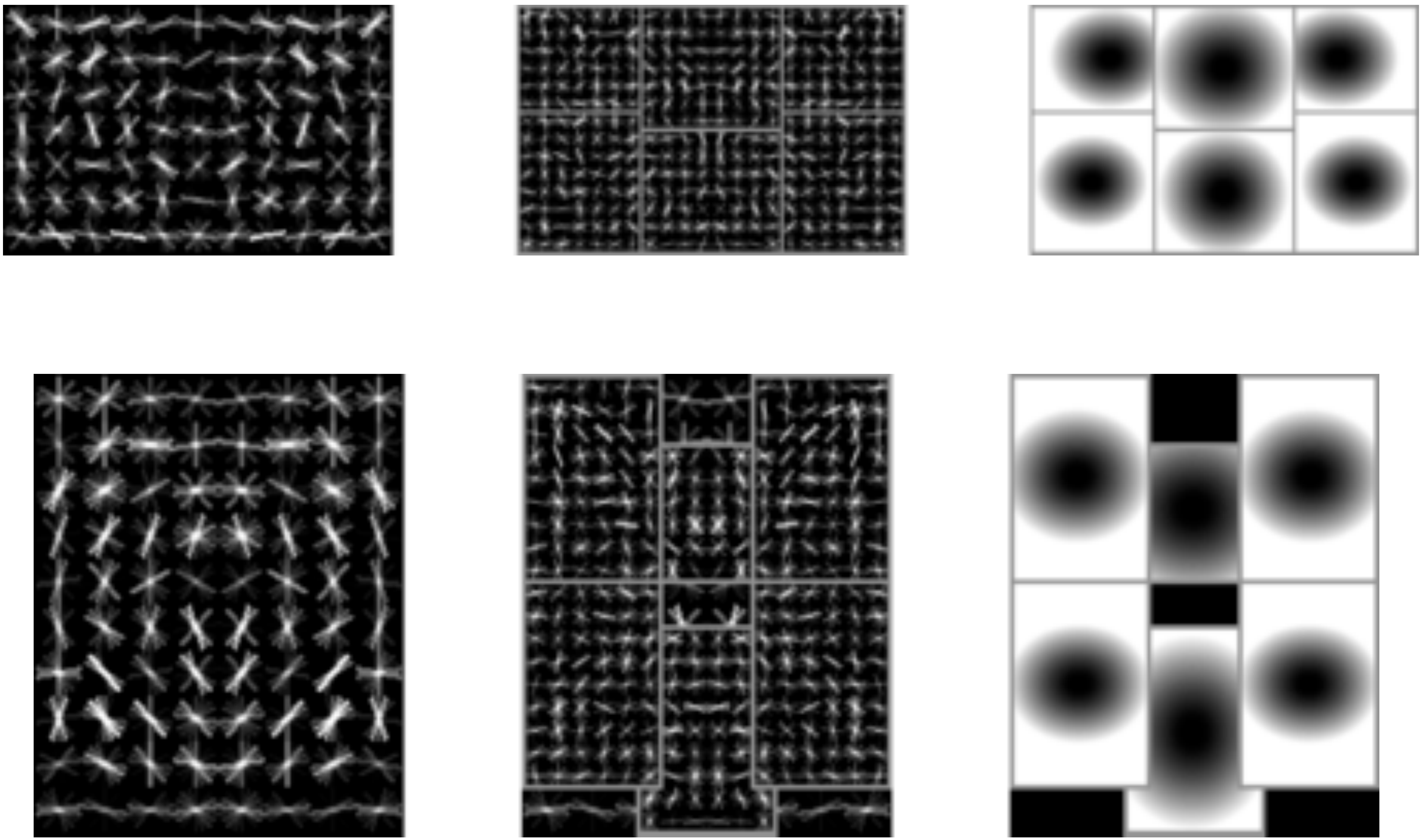
high scoring true positives



high scoring false positives
(not enough overlap)

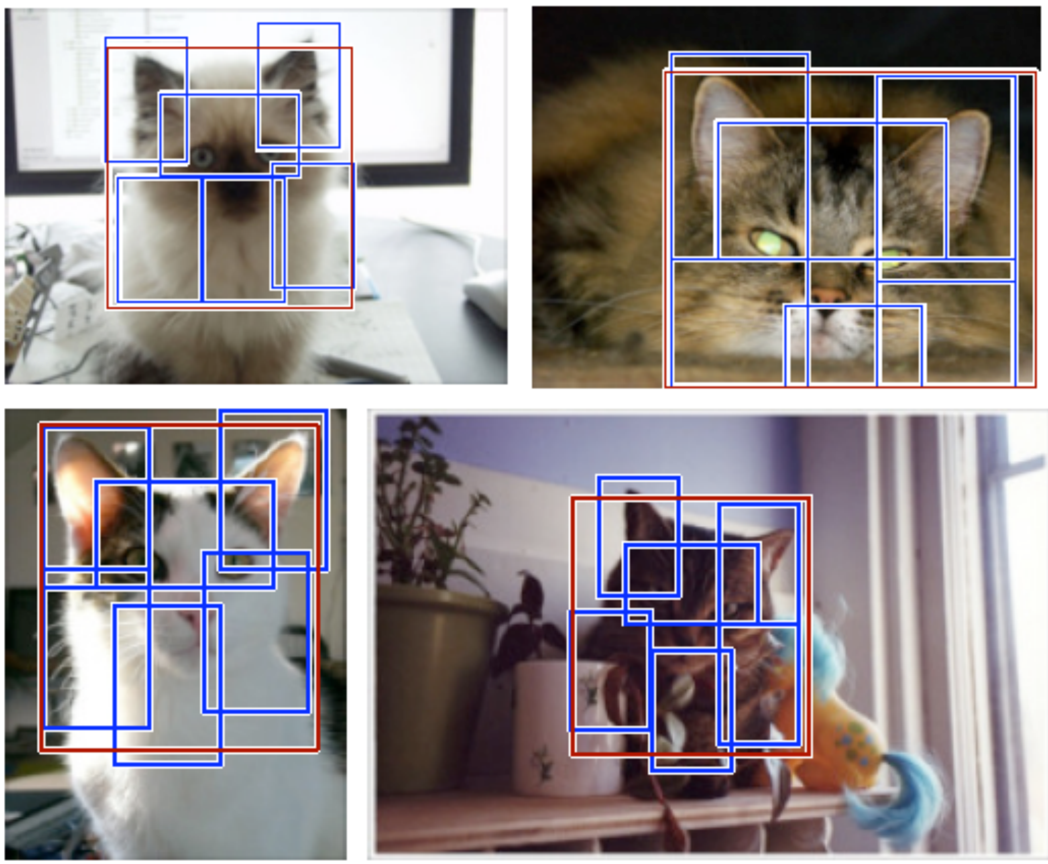


Cat model

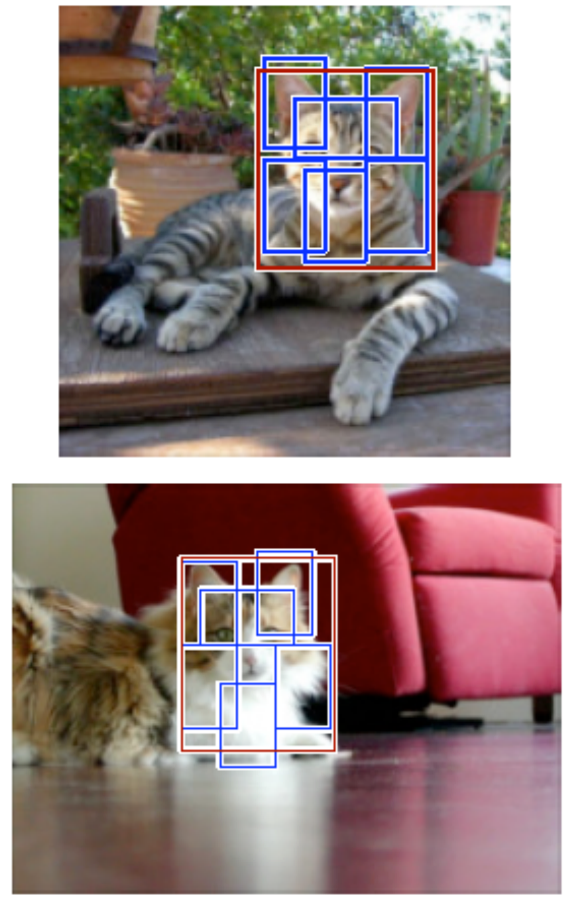


Cat detections

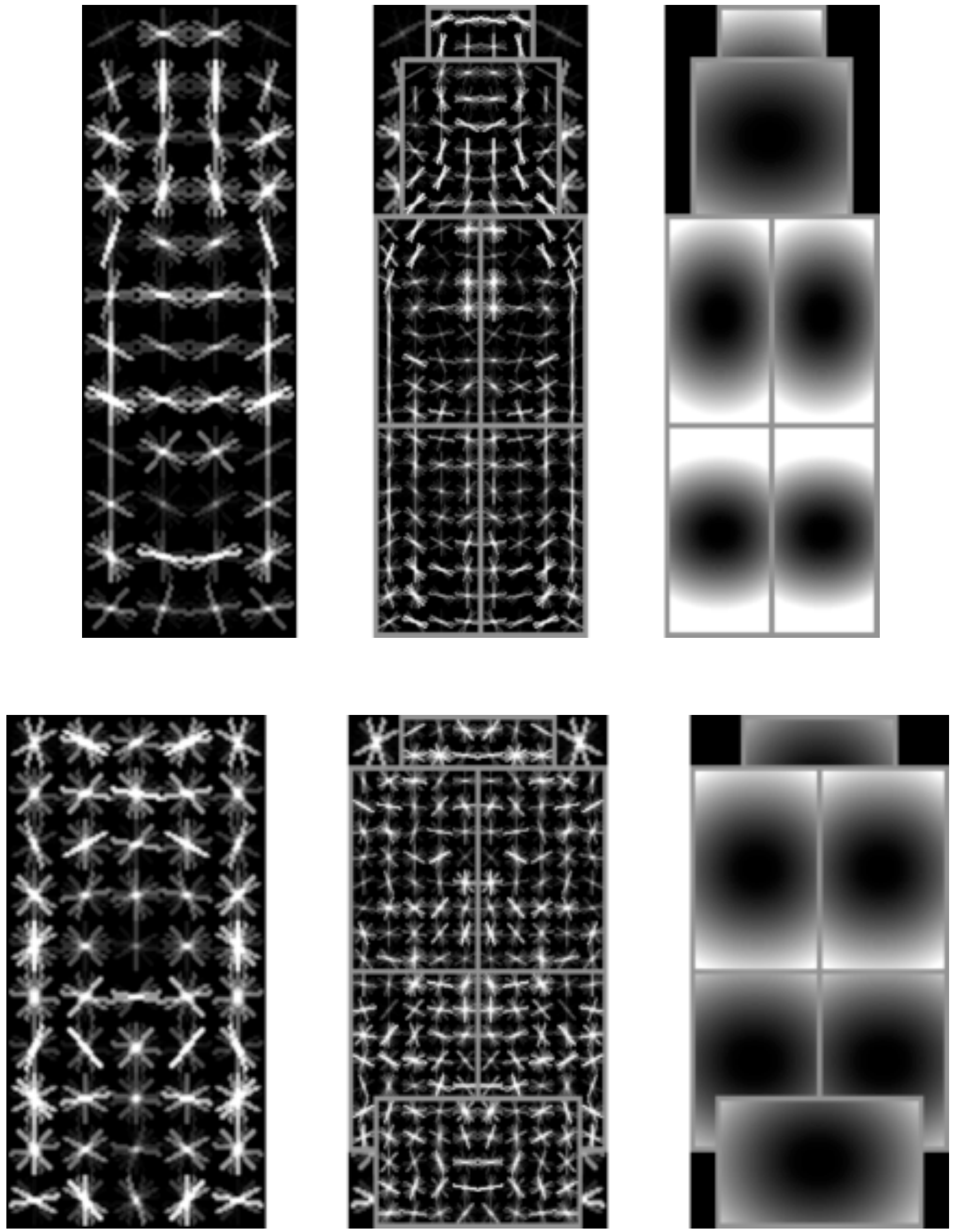
high scoring true positives



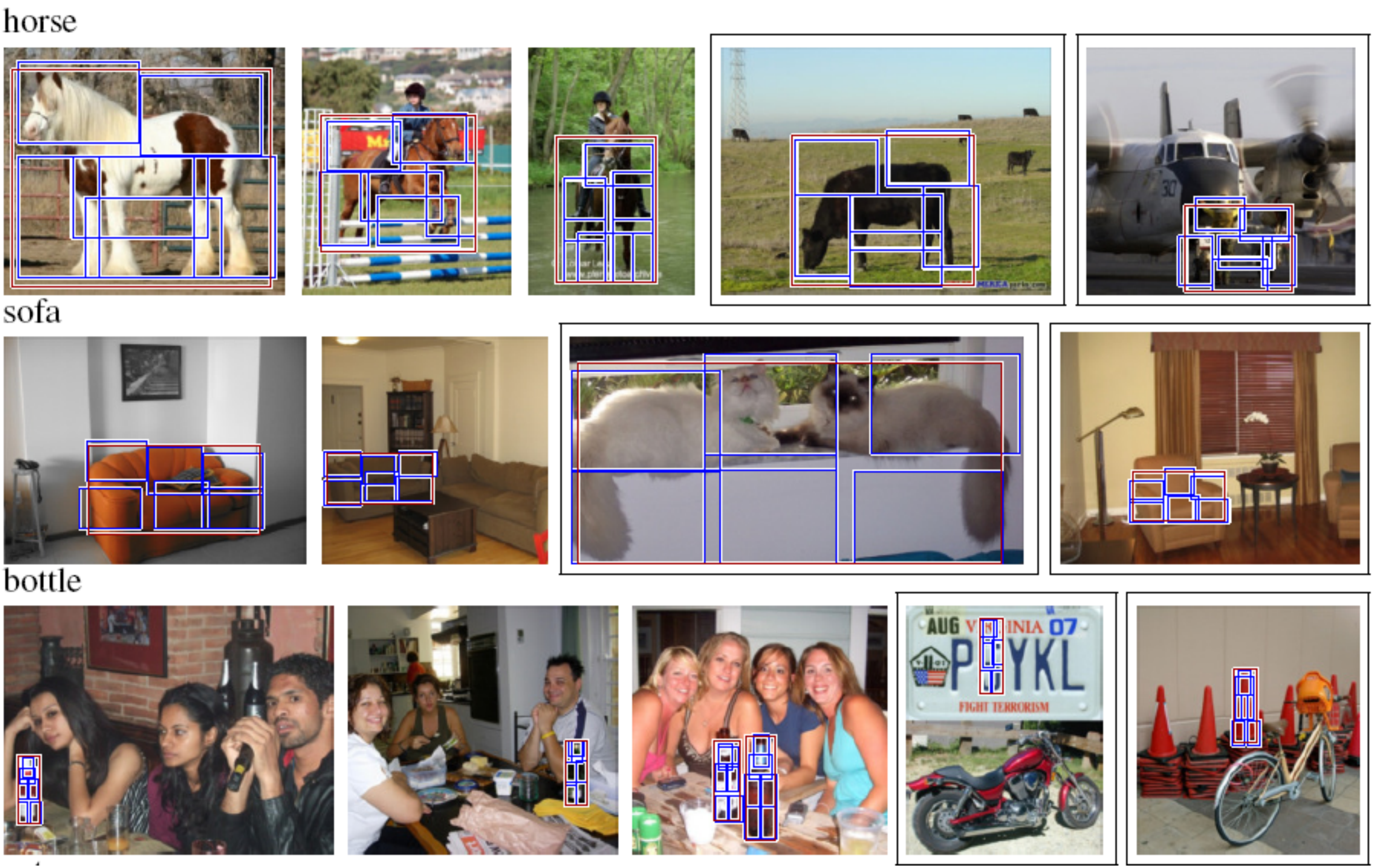
high scoring false positives
(not enough overlap)



Bottle model



More detections

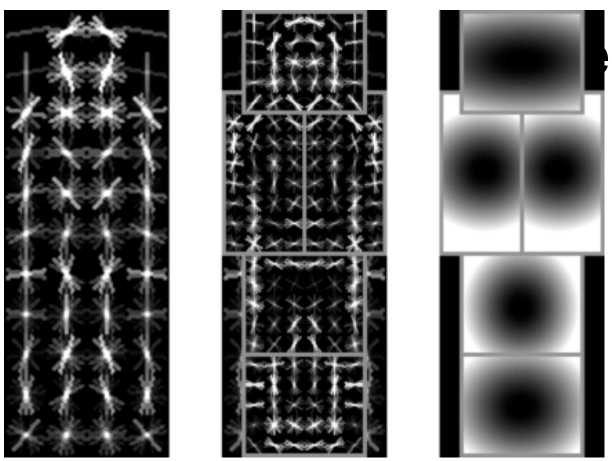


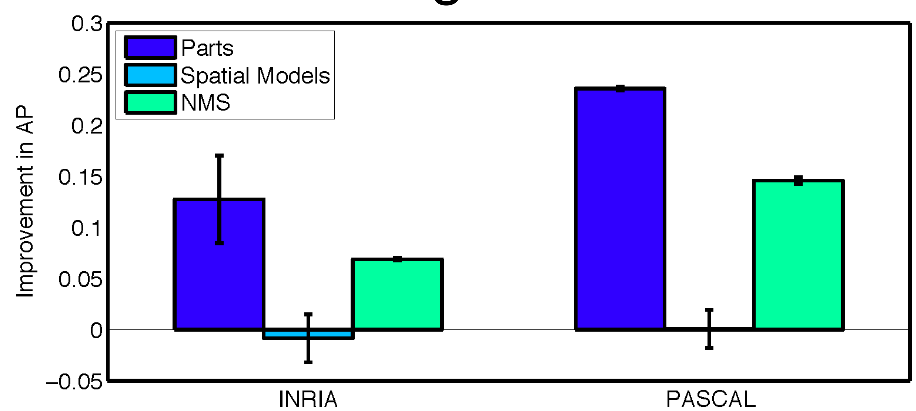
Background Selective Search

van de Sande et al ICCV 2011
(ILSVRC 2011)

State of the art

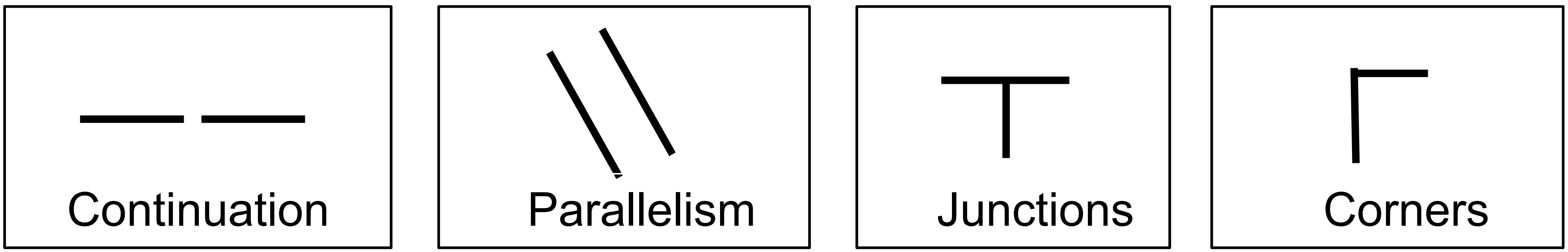
- Previous approaches
  - Hand Designed Features (SIFT, HOG, GIST …)
  - What is next ? Better Features ? More Training data ? Better classifiers ?
  - Main factor compared to humans is better features (Parikh & Zitnick’10) – study look at little patches and recognize





Mid-Level Representations

- **Mid-level cues**

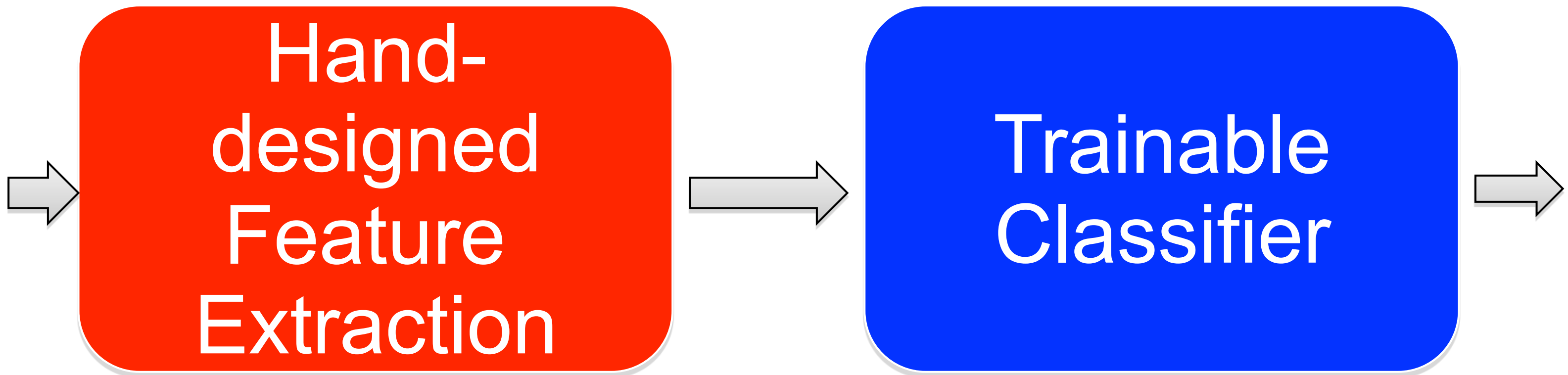


"Tokens" from Vision by D.Marr: 

- Object parts:

- Difficult to engineer, What about learning them?

Traditional Recognition Approach

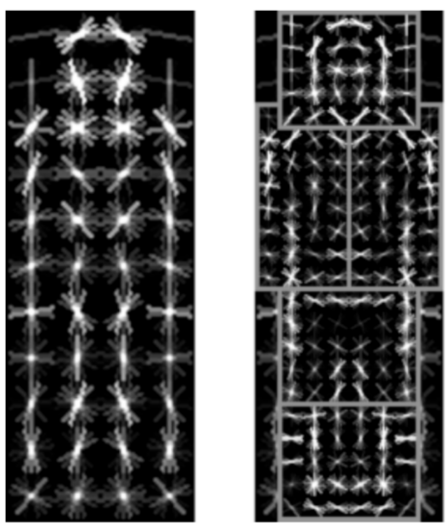


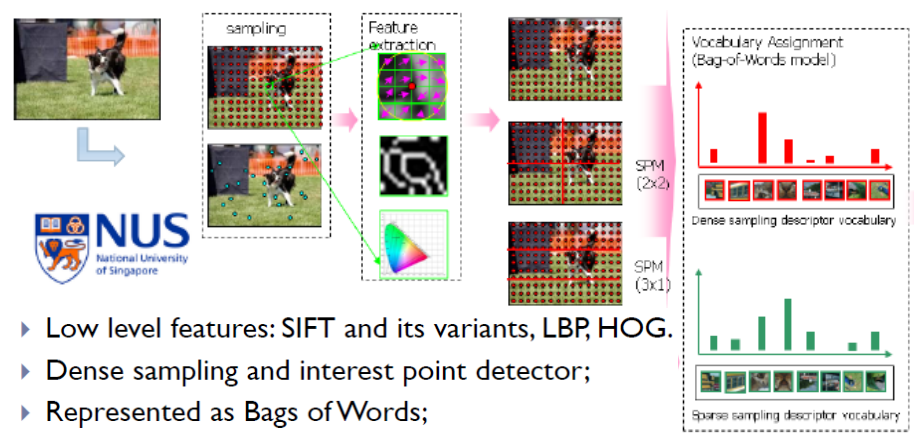
•

•

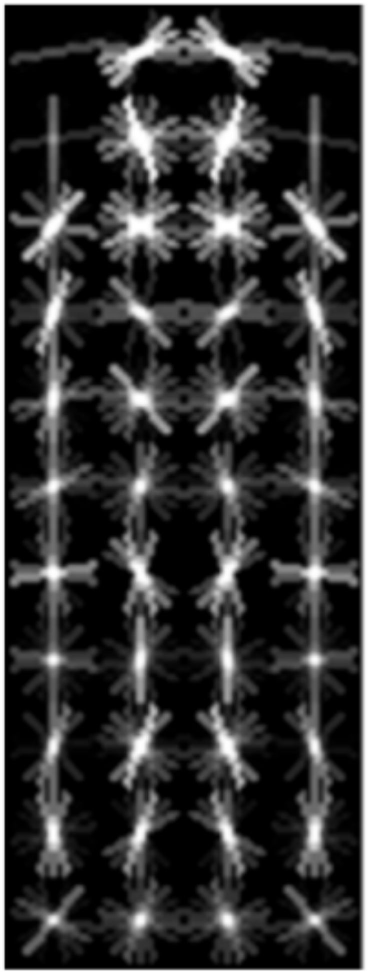
Motivation

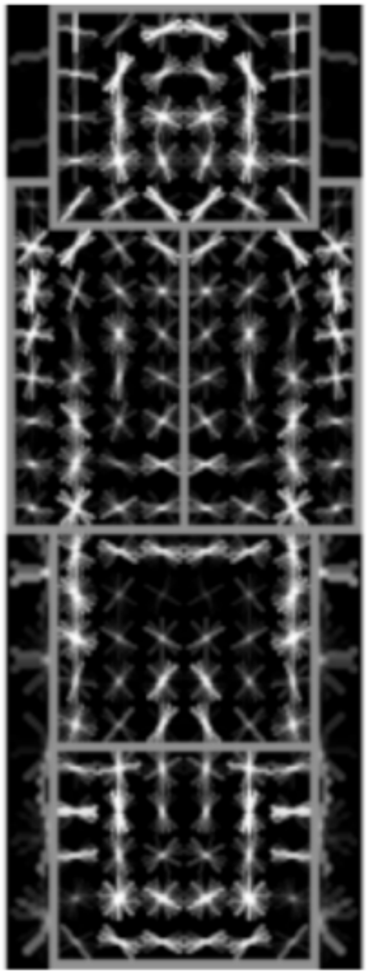
- Features are key to recent progress in recognition
- Multitude of hand-designed features currently in use
  - SIFT, HOG, LBP, MSER, Color-SIFT…………
- Where next? Better classifiers? Or keep building more features





Existing Methods





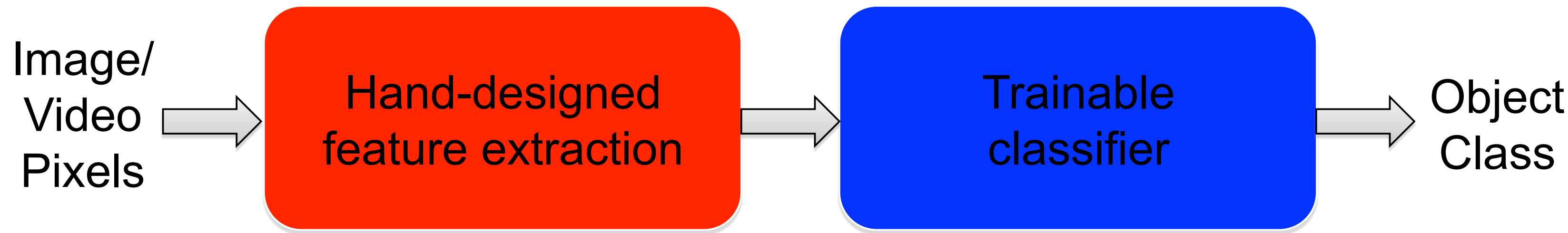
- Histogram of Gradient (HOG) features extracted at multiple scales
- Series of templates for model “parts”
- Springs between them to ensure geometric consistency

What about learning the features?

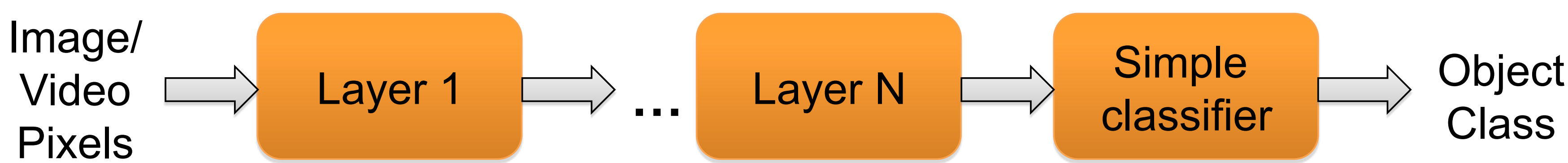
- Learn a *feature hierarchy* all the way from pixels to classifier
- Each layer extracts features from the output of previous layer
- Train all layers jointly

“Shallow” vs. “deep” architectures

Traditional recognition: “Shallow” architecture

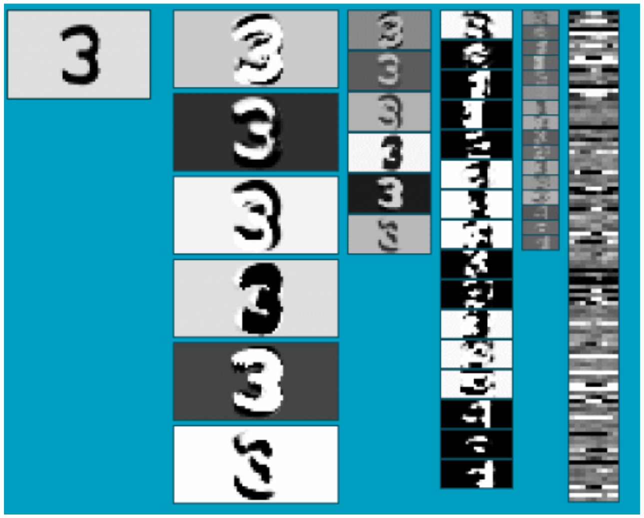


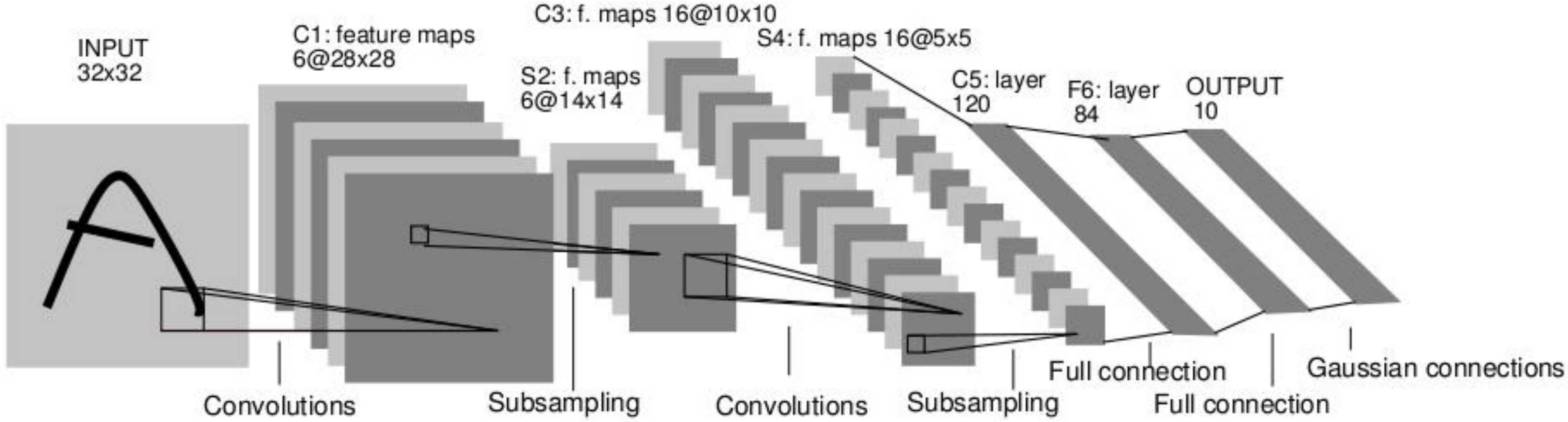
Deep learning: “Deep” architecture



Convolutional Neural Networks (CNN, Convnet)

- Neural network with specialized connectivity structure
- Stack multiple stages of feature extractors
- Higher stages compute more global, more invariant features
- Classification layer at the end

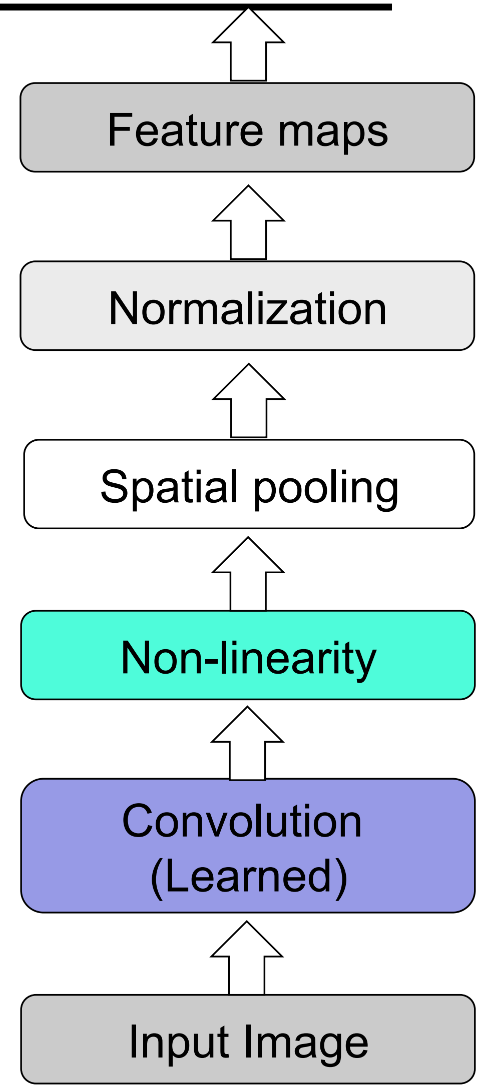




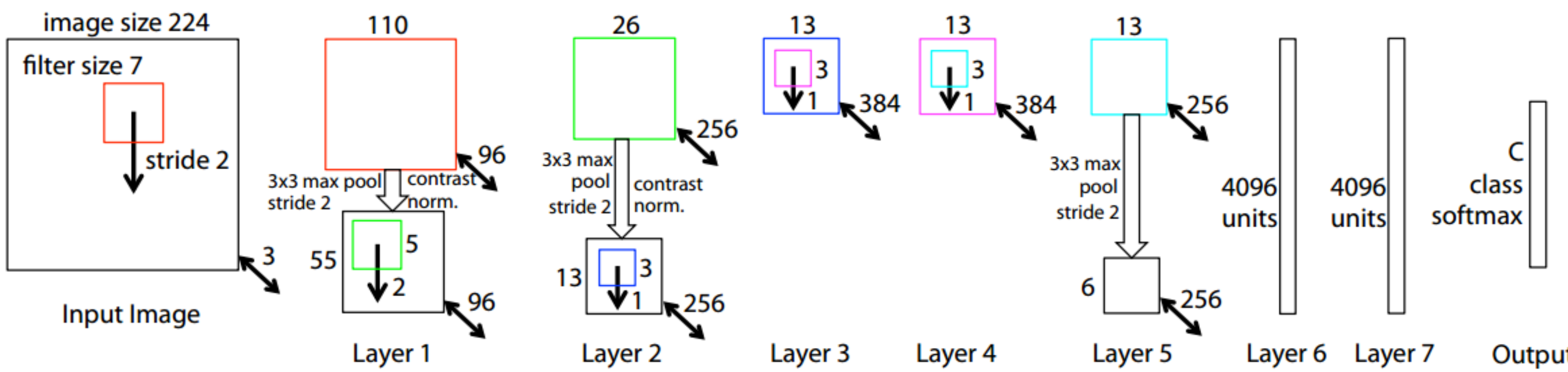
Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner,
Gradient-based learning applied to document recognition, Proceedings of the IEEE

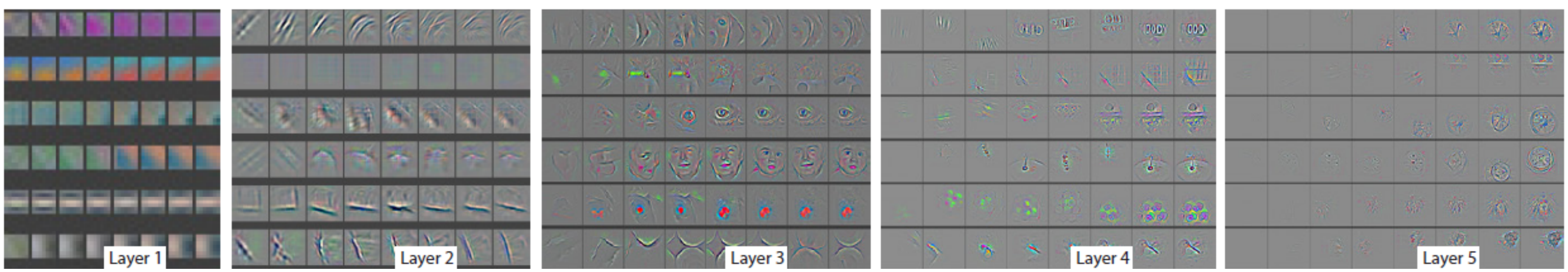
Convolutional Neural Networks (CNN, Convnet)

- Feed-forward feature extraction:
  1. Convolve input with learned filters
  2. Non-linearity
  3. Spatial pooling
  4. Normalization
- Supervised training of convolutional filters by back-propagating classification error



Deep Convolutional Neural Networks for Image Classification



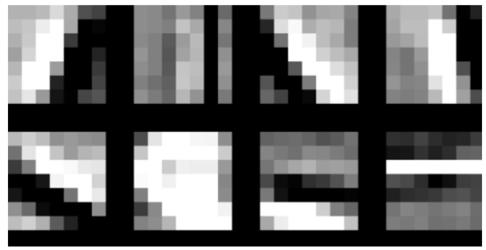


Many slides from Rob Fergus (NYU and Facebook)

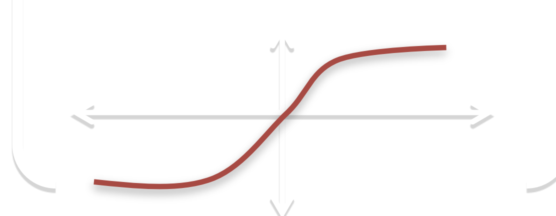
Example Feature Learning Architectures

Pixels / Features

Filter with Dictionary (patch/tiled/ convolutional)



+ Non-linearity



Normalization between feature responses

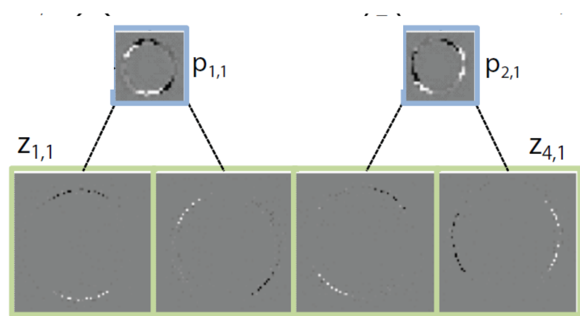
(Group) Sparsity

Max / Softmax

Local Contrast Normalization (Subtractive & Divisive)

Not an exact separation

Spatial/Feature (Sum or Max)

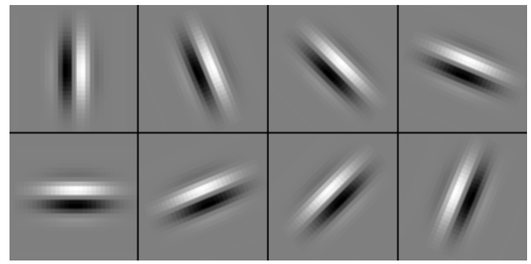


Features

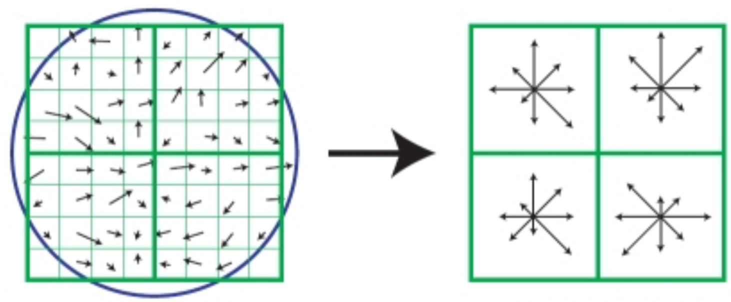
SIFT Descriptor

- Image Pixels

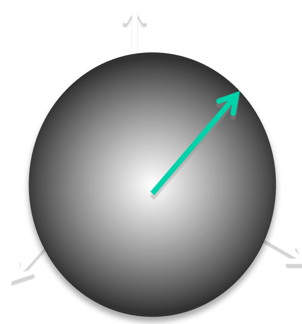
Apply Gabor filters



Spatial pool (Sum)

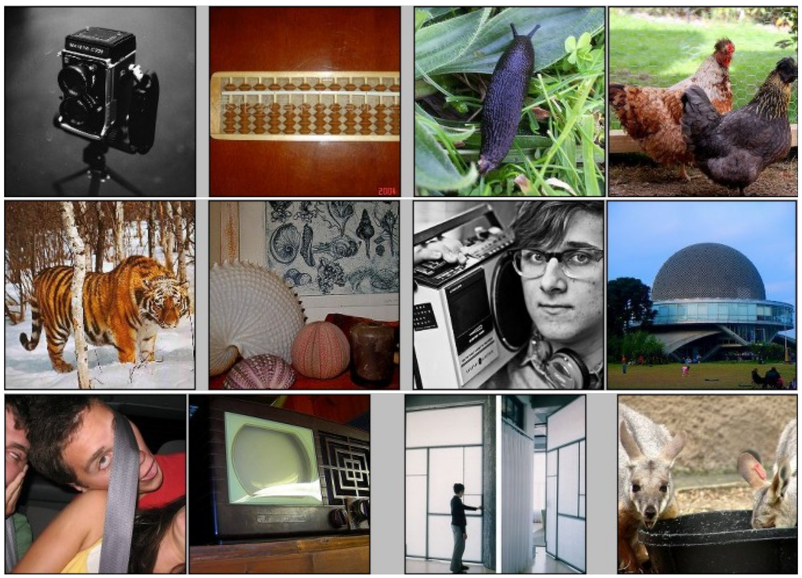


Normalize to unit length



Application to ImageNet

IMAGENET



- ~14 million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon Turk

ImageNet Classification with Deep Convolutional Neural Networks [NIPS 2012]

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

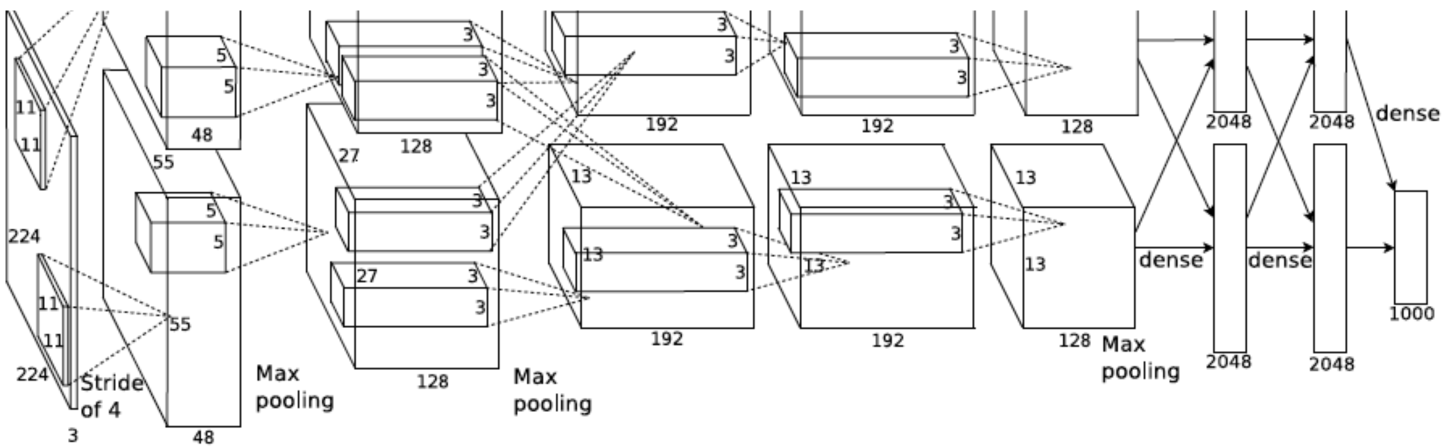
Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

Krizhevsky et al. [NIPS2012]

ImageNet Challenge 2012

- Similar framework to LeCun'98 but:
  - Bigger model (7 hidden layers, 650,000 units, 60,000,000 params)
  - More data (10^6 vs. 10^3 images)
  - GPU implementation (50x speedup over CPU)
    - Trained on two GPUs for a week
  - Better regularization for training (DropOut)

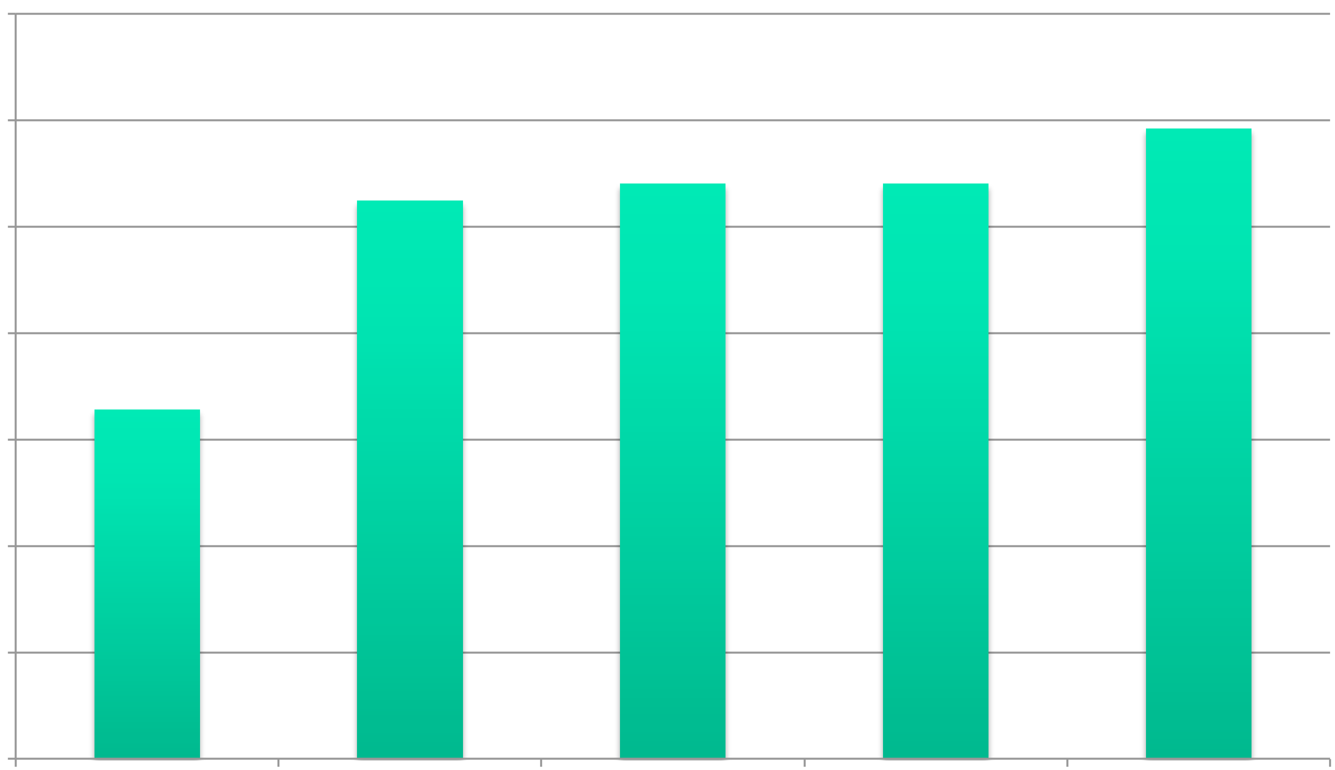


A. Krizhevsky, I. Sutskever, and G. Hinton,
[ImageNet Classification with Deep Convolutional Neural Networks](), NIPS 2012

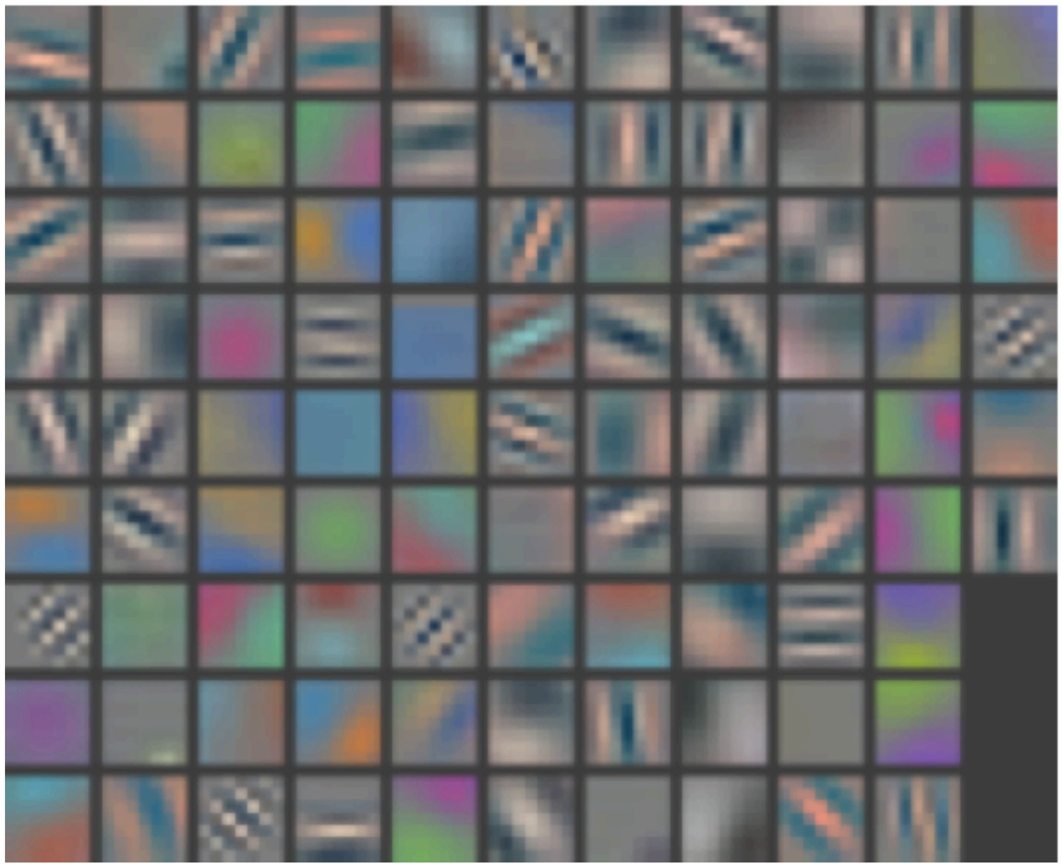
Tr

ImageNet Classification 2012

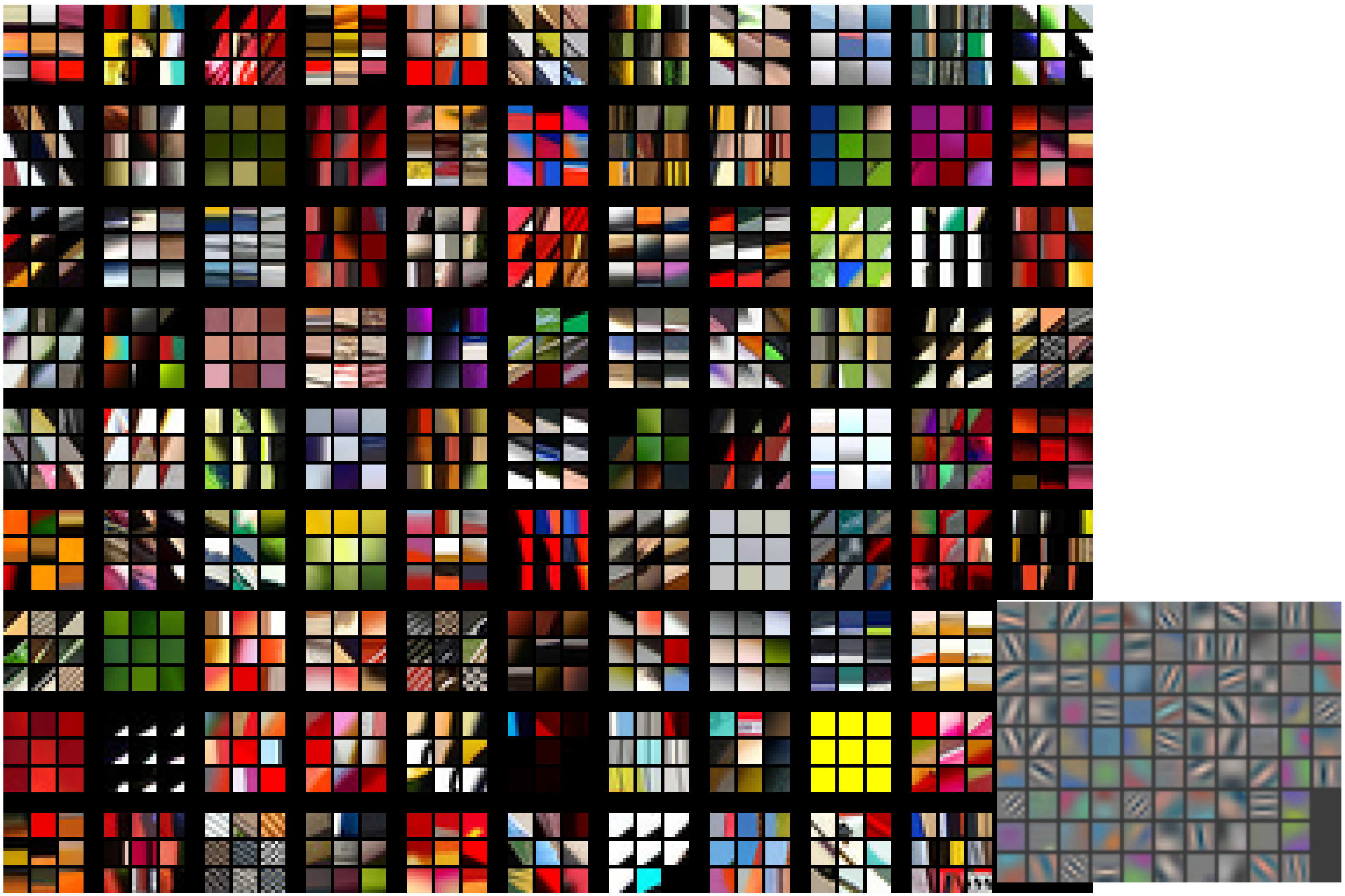
- Krizhevsky et al. -- 16.4% error (top-5)
- Next best (non-convnet) – 26.2% error



Layer 1 Filters



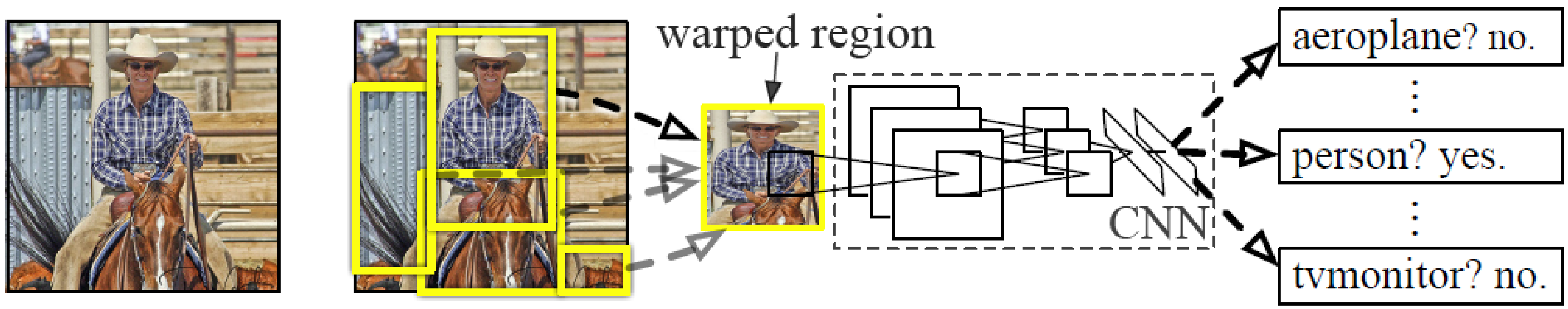
Layer 1: Top-9 Patches



Layer 3: Top-9 Patches

Background, R-CNN

R-CNN: Region proposals + CNN

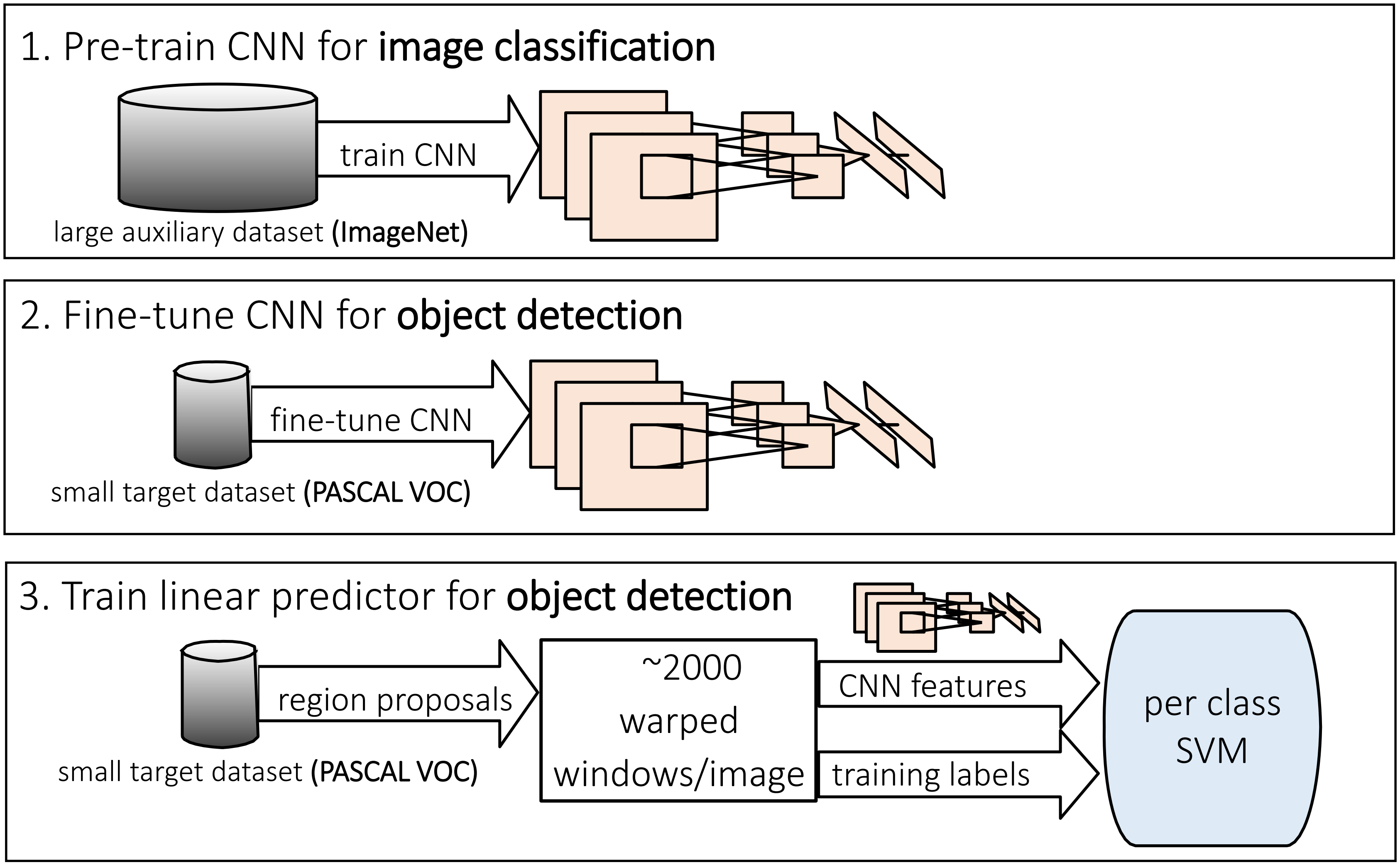


	localization	feature extraction	classification
this paper:	selective search	deep learning CNN	binary linear SVM
alternatives:	objectness, constrained parametric min-cuts, sliding window …	HOG, SIFT, LBP, BoW, DPM …	SVM, Neural networks, Logistic regression …

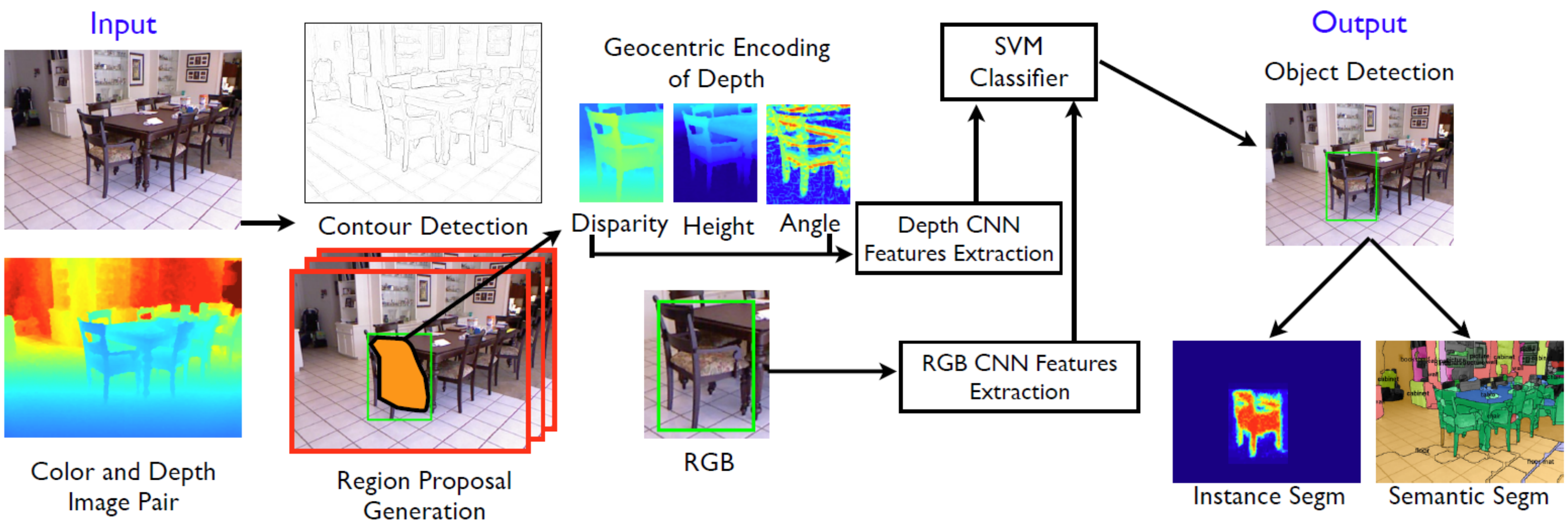
Girshick et al 2013

Results summary

R-CNN: Training



R-CNNs on RGB-D
for Object Detection and Segmentation



Pre-trained on Image-Net using RGB images.
Fine-tuned on NYUD2 (400 images) and synthetic data.
SVM training on pool5, **fc6** and fc7.