

Image Search & Retrieval





P J Narayanan

CS5765. Computer Vision. Spring 2013

CVIT, IIT, Hyderabad

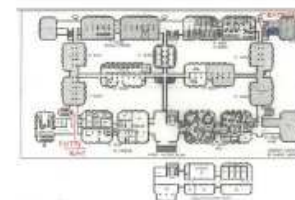


Images and Videos on the Net

- Rapid expansion in visual content on the net.
-  ,  Picasa ,  ,  , ...
- Statistics: 59% or above of internet traffic is videos.
Mostly streaming TV programs like Netflix, TiVo.
About 15% is YouTube.
- Videos and images are bulky, so that is easy
- Do we have tools to organize/search visual content?
- Text search engines are excellent. Shouldn't we be making similar for the bulk content?
- Google image search is there

About 68,800 results (0.24 seconds)

SafeSearch

Related searches: [iiit hyderabad logo](#)
[gallery seaview99 2.jpg](#)
[icon2010.in](#)

 400 x 300 - NLP Association, India - IIIT
 Hyderabad - LDC-IL, CIIL - IIT Kharagpur
 Similar - More sizes


Google Image Search

- Is really simple text search!!
- Images are annotated using keywords given by individuals.
- Or using nearby words from documents that contain the images.
- The image and its contents are not looked at.
- How do we build a search engine that looks inside?
- Before that, how does a text search engine work?



Text Search Engines: How do they work?

Documents, Words, and More

- Webpages or **documents** are the basic units search engines handle
- Documents contain **words**, which are the atomic elements
- A document is an **unordered collection** of words.
- First two steps of processing:
 - **Stop word removal:** Eliminate very common words like *a, an, the, at, or, on, etc..* They appear frequently and play no role
 - **Stemming:** convert the words to its root. *Run, runner, running, runs, etc.* become just “**run**”

A Document

Words: center, vision, inform, technology, research, international, active, involve, impart, train, today, faculty, work, talk,

Center for Visual Information Technology



About CVIT:

Centre for Visual Information Technology (CVIT) is a research centre at International Institute of Information Technology at Hyderabad. The centre is actively involved in research and imparting advanced training through seminars, workshops and semester long courses in the fields of Computer Vision, Image Processing, Computer Graphics, Pattern Recognition and Machine Learning.

CVIT today has 4 faculty members, over 40 research students, and 20-25 UG honours students working on different problems. In addition, several top researchers in the area visit, give talks, and interact with CVIT's researchers.

CVIT has several externally funded projects, funded by agencies including **Ministry of Communications and Information Technology (MCIT)**, **Department of Science and Technology (DST)**, different labs of the **Defence Research and Development Organization (DRDO)**, **Naval Research Board (NRB)**, **Microsoft Research**, **Philips Research**, and **Lions International**.

- Home
- Focus
- People
- Theses
- Projects
- Publications
- Visitors
- Major Events
- Resources
- Contact Us

Document Vector

- The vocabulary (dictionary) of all words used is known
- Each word is converted to its index from the dictionary
- A document is represented by the histogram of words of the vocabulary in it, called a **document vector**
- What is the dimensionality of the document vector?
- Work with the document vectors for the entire collection of documents

Crawl the web, get many documents and represent each using a document vector

abacus	0
active	3
⋮	⋮
center	8
⋮	⋮
impart	1
inform	7
⋮	⋮
today	2
⋮	⋮
work	3
⋮	⋮
zebra	0

Inverted Index

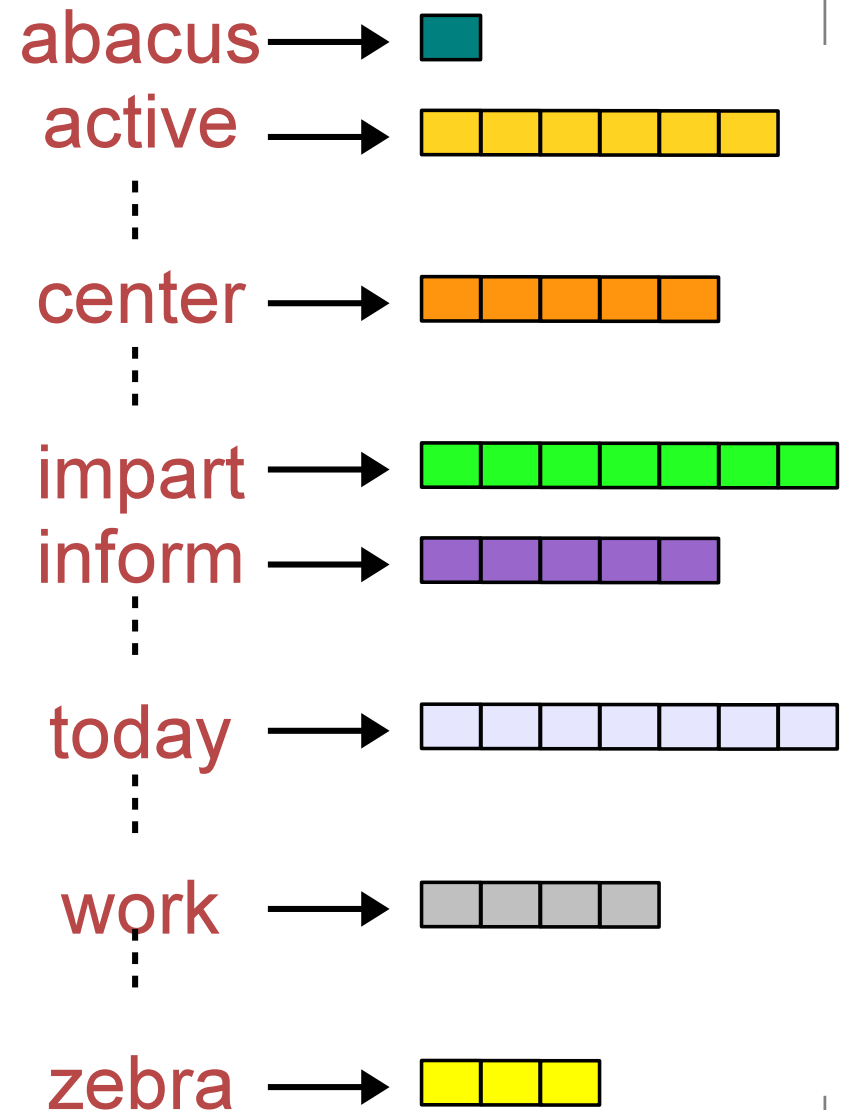
Each document is represented by a high-dimensional vector

Inverted Index: For each word, the list of documents in which it appears, like the index at the end of a book

tf-idf: Term frequency and Inverse document frequency

Weight each word using its tf-idf

Crawl the web and build such an index for all documents!!



tf-idf: Are all words equal?

- Intuition 1: A word that occurs frequently represents it well. **Boost it!**
- Intuition 2: A word that occurs in all documents carry no discriminative information. **Suppress it!**

tf-idf: Are all words equal?

- Intuition 1: A word that occurs frequently represents it well. **Boost it!**
- **term frequency:** Frequency of the word in a document. High **tf** preferred. $tf_{id} = \frac{n_{id}}{n_d}$ or $tf_{id} = \log \frac{n_{id}}{n_d}$
- Intuition 2: A word that occurs in all documents carry no discriminative information. **Suppress it!**
- **inverse document frequency:** The ratio of the total number of documents to the number of documents in which a word occurs. $idf_i = \frac{N}{N_i}$ or $idf_i = \log \frac{N}{N_i}$
- Weight of word i of document d : $w_{id} = tf_{id} \times idf_i$
That's what is stored in the document vector for d

What does a search company do?

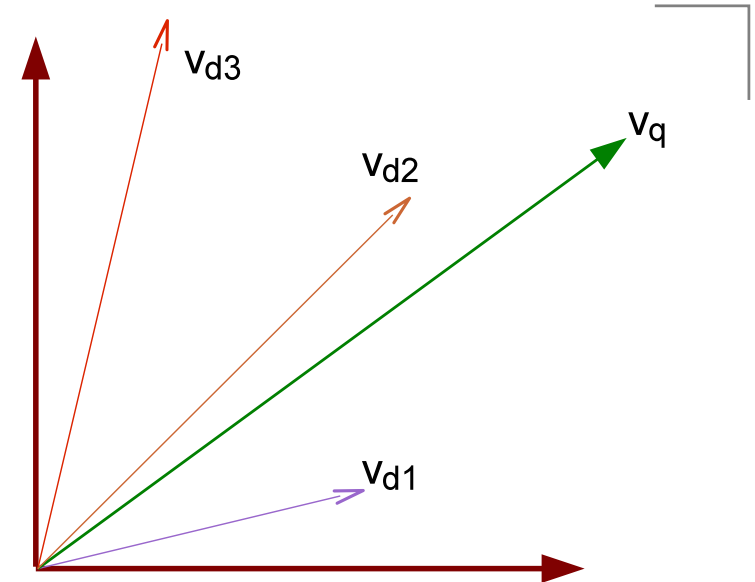
- Crawl the web frequently and collect pages to be indexed
- Convert each document to its document vector with tf-idf weights
- Build the inverted index of all documents
- Store the index in a database/file. Hashing is used to represent the inverted file usually
- How many pages in Google today?
- Very expensive computationally

Query Process

- Stem and form a document vector of the query keywords (or the query document)
- Identify the documents with the keywords from the inverted index. Needs special handling for conjunction (AND), disjunction (OR), etc.
 - Recover documents for each keyword and find their intersection or union as the case may be
- Find the similarity of query vector v_q and each document vector v_d
- Cosine distance or dot product of each with the query vector $s(d, q) = \frac{v_q v_d}{||v_q|| ||v_d||}$
- Present top k documents in similarity order

Cosine Distance for Similarity

- Distance between document vectors is not reliable
- Dot product or cosine distance is much better.
- Use the dot product if vectors are normalized
- Computing cosine similarity to all N documents and finding top k ones is wasteful. How else can we do it?
- Rank documents in the inverted index on the basis of importance. Evaluate similarity in that order. Keep the top k results in a **priority max heap**.
- Google's **PageRank** algorithm uses the number of links pointing to a document as its importance measure



Bag of Words Model

- This representation for a document is called the **bag of words (BoW)** model
- Ordering is unimportant; only word frequencies are.
Rama killed Ravana and **Ravana killed Rama** same!??
- Plus points of the model
 - Unified, fixed-length representation of documents
 - Efficient way to match documents
- Problems and limitations of such a model:
 - **Synonymy:** “Woman” and “female” are totally different. No way to know they are the same.
 - **Polysemy:** “Chip” means different things in everyday sense and in electronics hardware
 - **Independence:** “United”, “States” and “America” are independent even if they occur together often

Bag of Words Framework

- General representation structure:
 - Represent variations of a theme as a “word”
 - Document is a histogram of vocabulary words
 - Fixed size vectors represent documents of all sizes
- General usage structure:
 - Retrieval: Find similarity and rank documents
 - Recognition: Train a classifier using many training samples. Use the classifier to assign a category to query documents

Video Google Retrieval System

by Josef Sivic and Andrew Zisserman

ICCV 2003

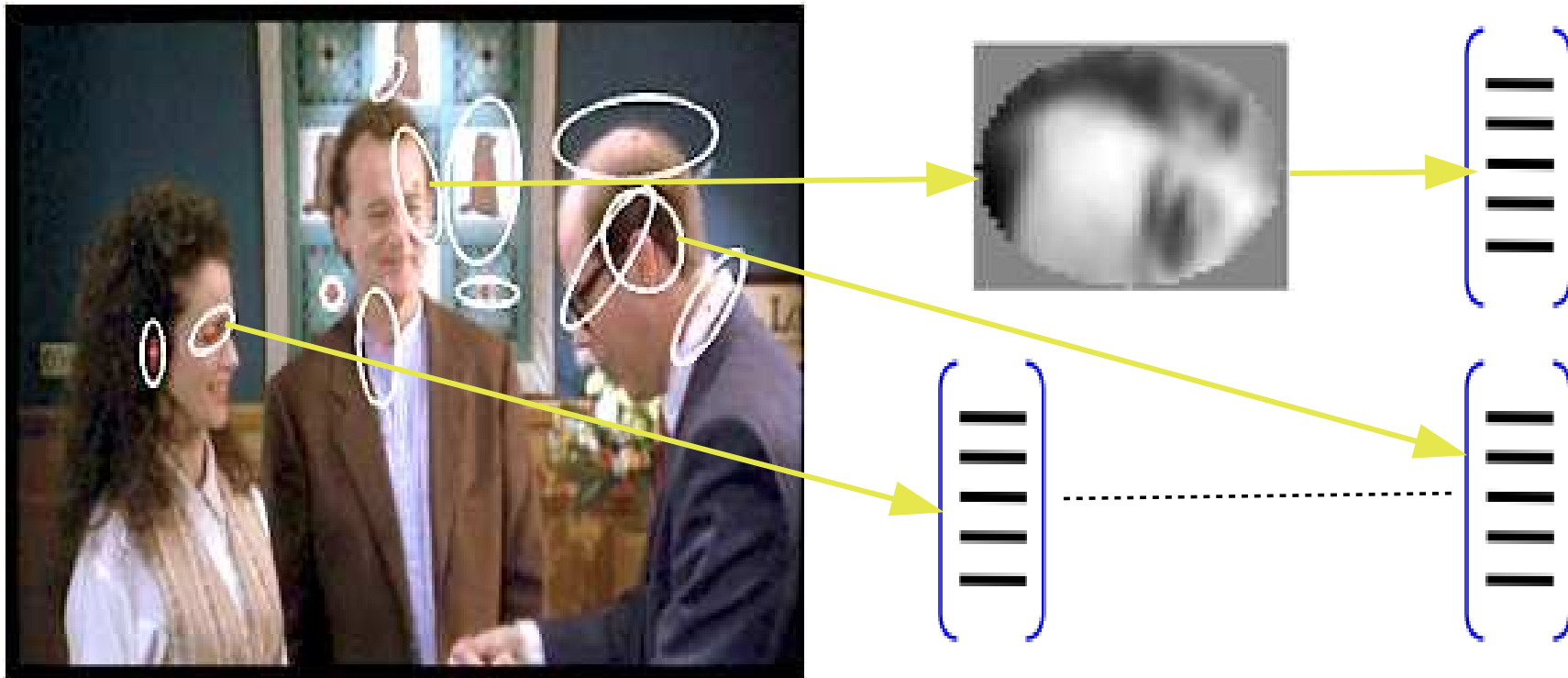
Demo at:

<http://www.robots.ox.ac.uk/~vgg/research/vgoogle/>

Main Idea

- Adapt the text search procedure for searching images and videos
- Representation and Indexing:
 - Start with local image features like SIFT
 - *Quantize* the feature descriptors into **visual words**
 - Represent each image or video frame using a document vector of histograms of visual words
 - Build an inverted index of visual words to images
- Searching using an image window:
 - Build the document vector for the query region
 - Retrieve documents for query words from the indexed collection
 - Rank them using cosine similarity

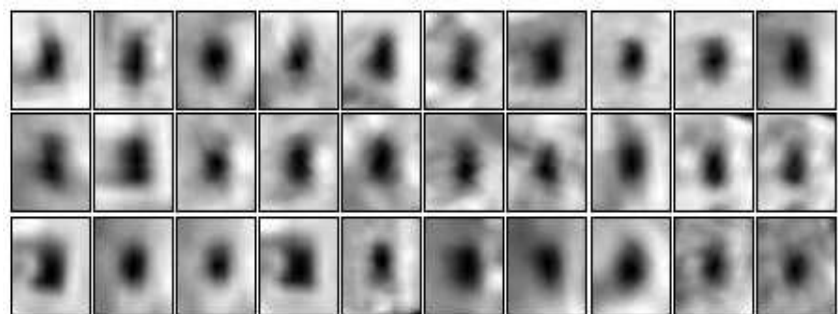
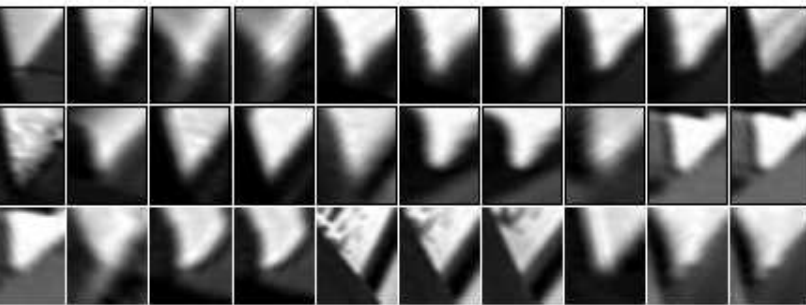
Detection and Description



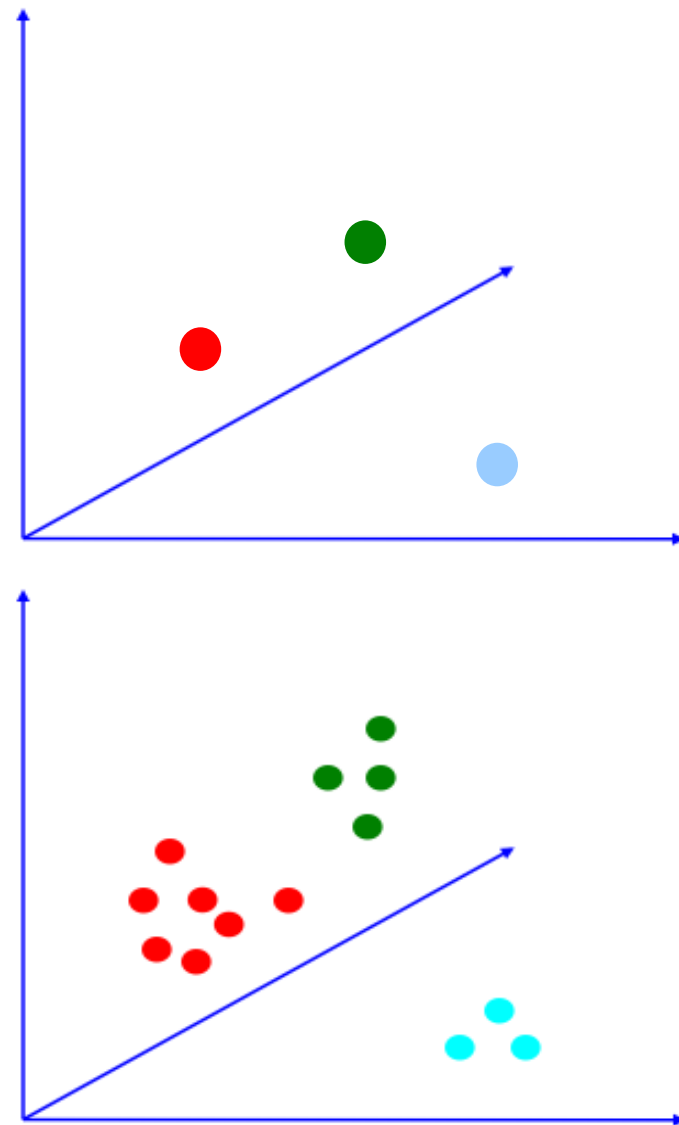
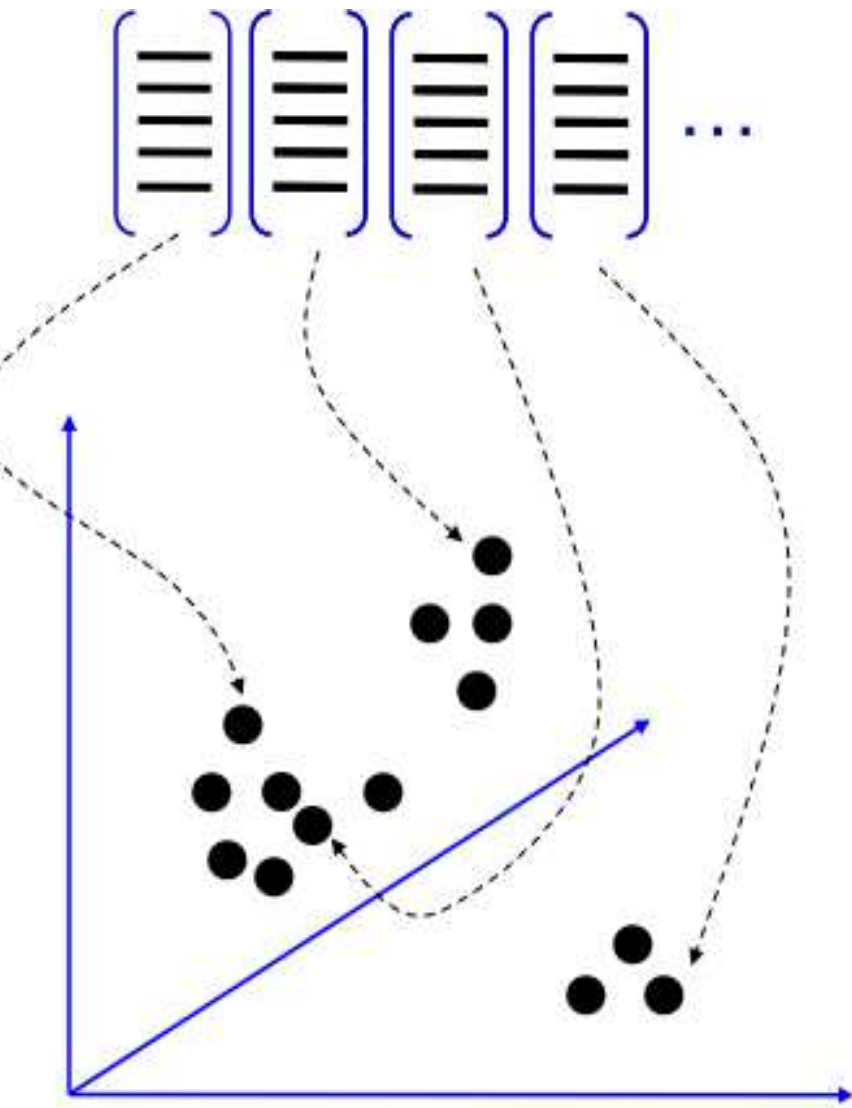
- Two types of keypoints: Harris corners (Shape Adapted) and Maximally Stable for blobs with sharp contrasts.
- Represent each using a SIFT descriptor
- Far too many SIFT vectors. So, use a codebook!

Creating Visual Vocabulary

- K-Means clustering of a large number of descriptors.
- Mahalanobis distance: $\sqrt{(\mathbf{x}_1 - \mathbf{x}_2)^T \Sigma^{-1} (\mathbf{x}_1 - \mathbf{x}_2)}$
- 6K Shape Adapted and 10K Maximally Stable clusters, in same proportion as interest points
- SA and MS regions are kept in separate vocabularies.



Clustering



Inverted Index

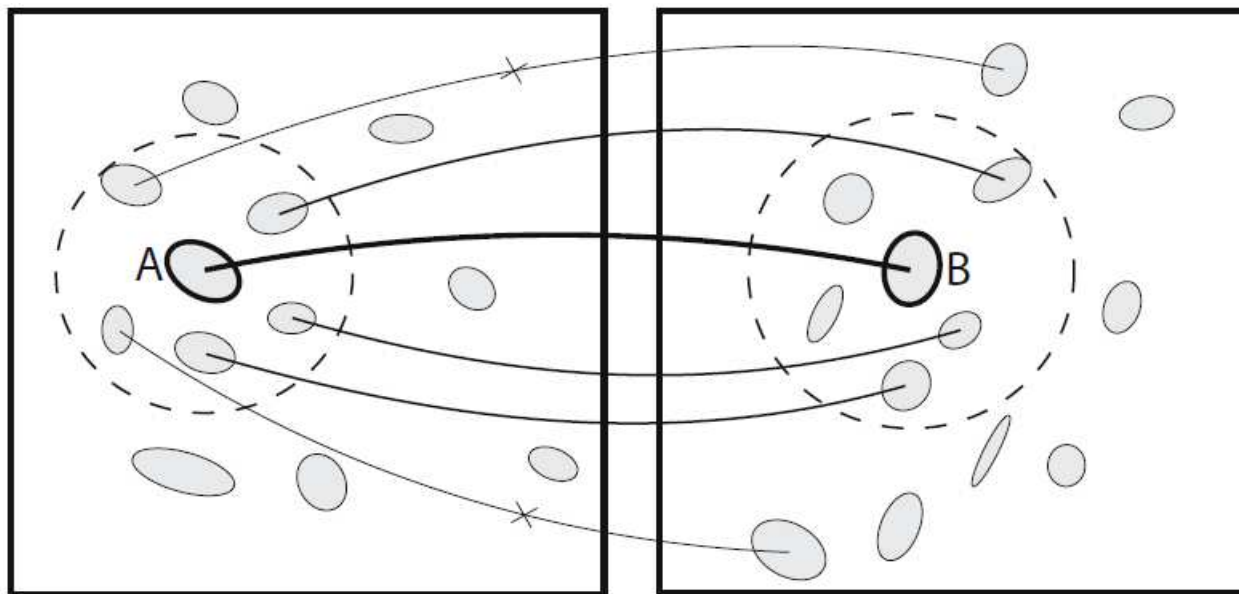
- Represent each image using a histogram of visual words
- Exactly in the same way text search does it
- Rank the visual words based on their frequency of occurring in the images of the collection.
- **Stop List:** Remove top 5% and bottom 5% of the VWs. Correspond to large clusters due to small specularities, etc.

Query Process

- User outlines a rectangle around an object being searched
- Find the interest points and descriptors in the rectangle
- Form a query vector after mapping each point to a VW
- Select possible images from the index and compute the cosine similarity measure with each
- Rank them using cosine similarity
- **Spatial consistency:** Rerank them using a spatial consistency measure

Spatial Consistency

- Prefer matches between images that occur nearby
- Matches within a window of each match will vote positively
- These votes are added to the cosine score for final ranking

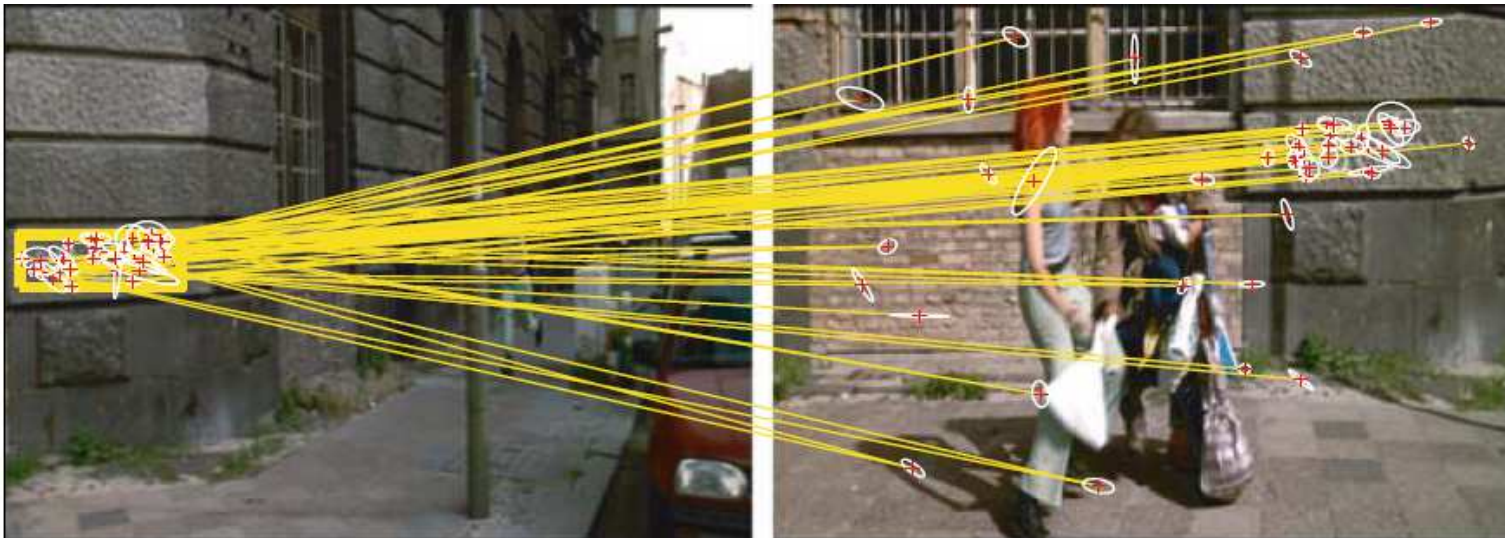


Matching Example

Query rectangle and closeup:



Raw visual word matches:

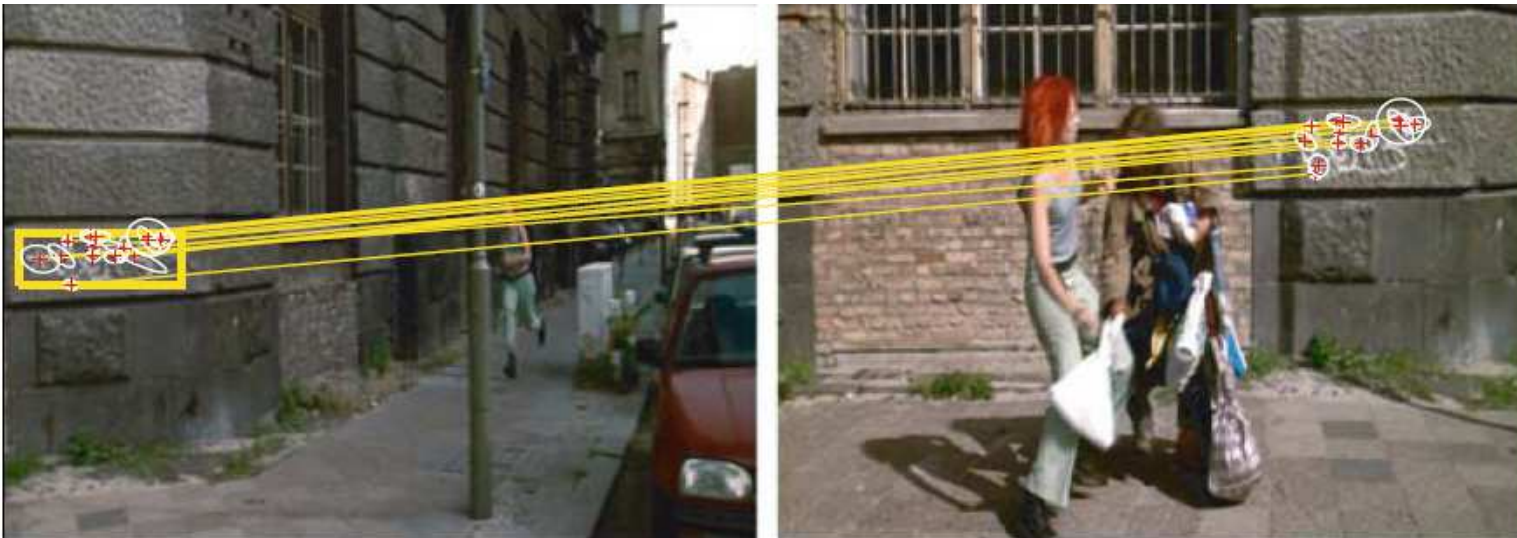


Matching Example ...

After applying stop list:



After spatial consistency:



Video Google: Algorithm

Offline Pre-Processing (for videos)

- Detect interest points/regions and represent as SIFT
- Remove unstable regions by tracking in video
- Build the vocabulary by clustering vectors from a subset of the video
- Remove words from the stop list
- Compute tf-idf weighted document vector ($= \frac{n_{id}}{n_d} \log \frac{N}{n_i}$)
- Build the inverted index structure

Live search given a query rectangle

- Find the visual words in the query region
- Retrieve keyframes using index and cosine distances
- Re-rank the top 500 matches using spatial consistency

Results

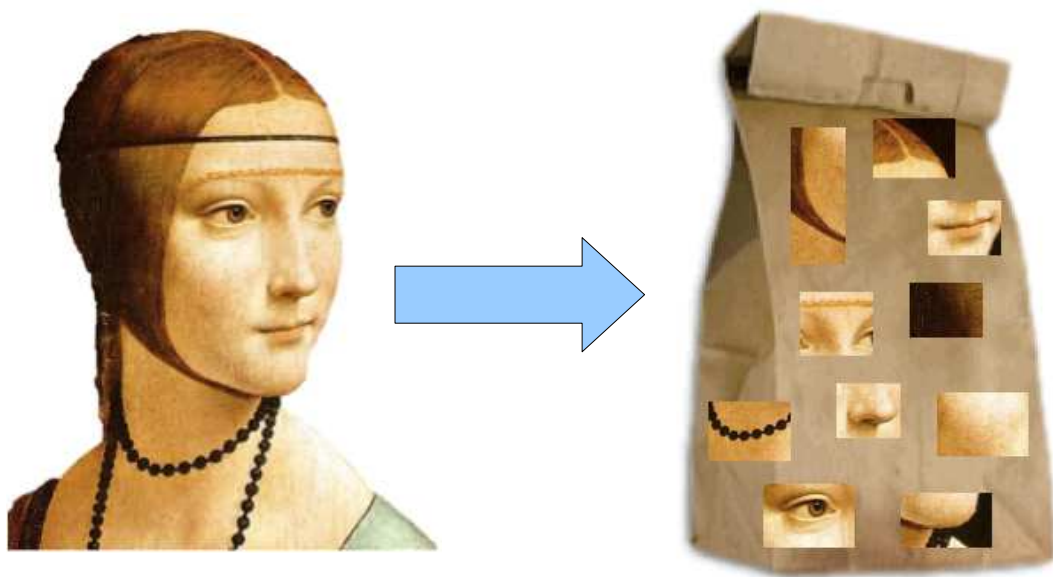


Bag of Words Model and Beyond

Bag of Words Model

- Unordered collection of feature descriptors
- Ordering is unimportant; only word frequencies are.
Rama killed Ravana and **Ravana killed Rama** same!??
- Plus points of the model
 - Flexible to wide range of variations
 - Ability to handle occlusions
 - Unified, fixed-length representation of documents
 - Efficient way to match documents

Bag of Words Framework



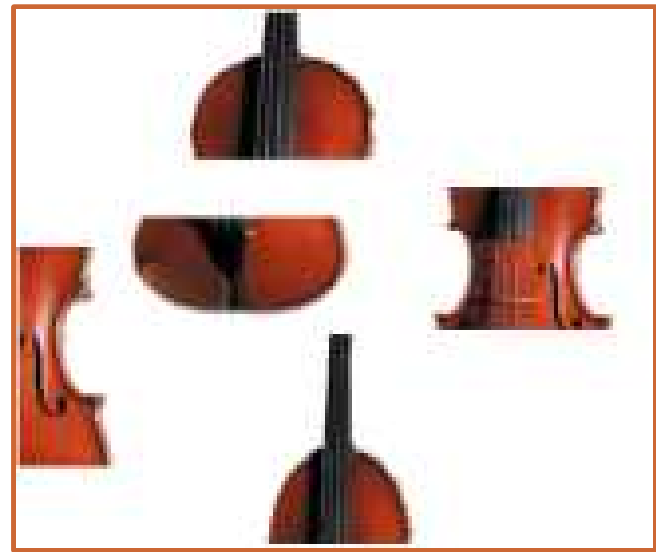
- Represent variations of a theme as a “word”
- Document is a histogram of vocabulary words
- Fixed size vectors represent documents of all sizes
- Retrieval: Find similarity and rank documents
- Recognition: Train a classifier using many training samples.

Independent Features

Features are some local, basic description.

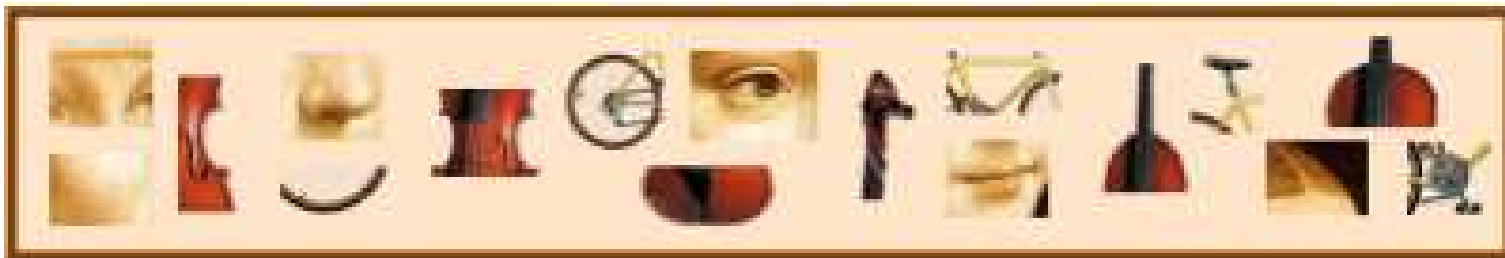
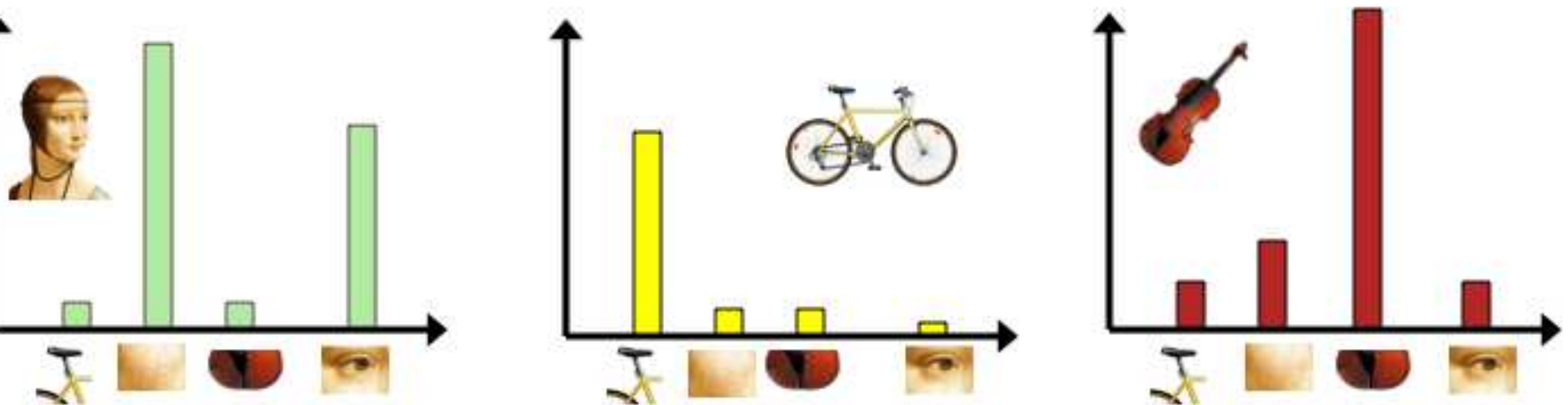
Independent features.

Described in a repeatable manner



Histogram Representation

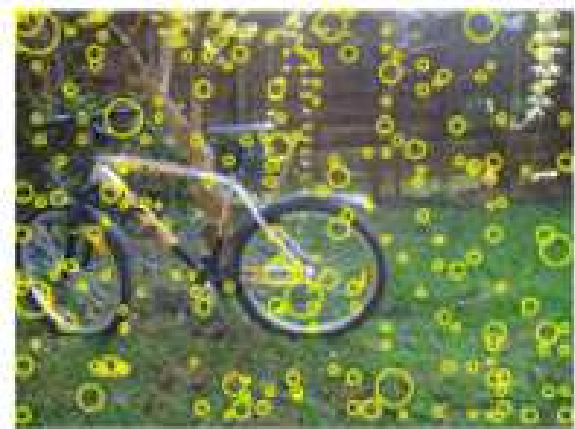
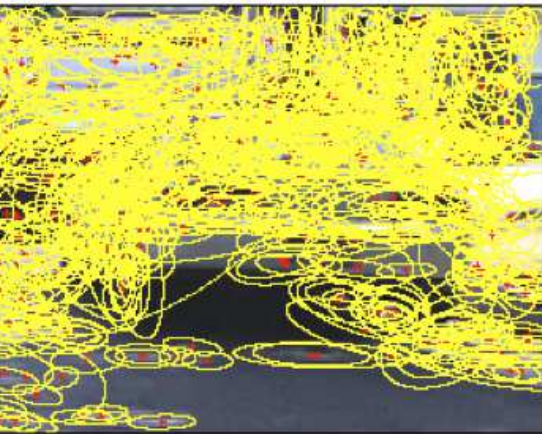
Represent each object as a histogram of such features.



Feature Sampling Strategies

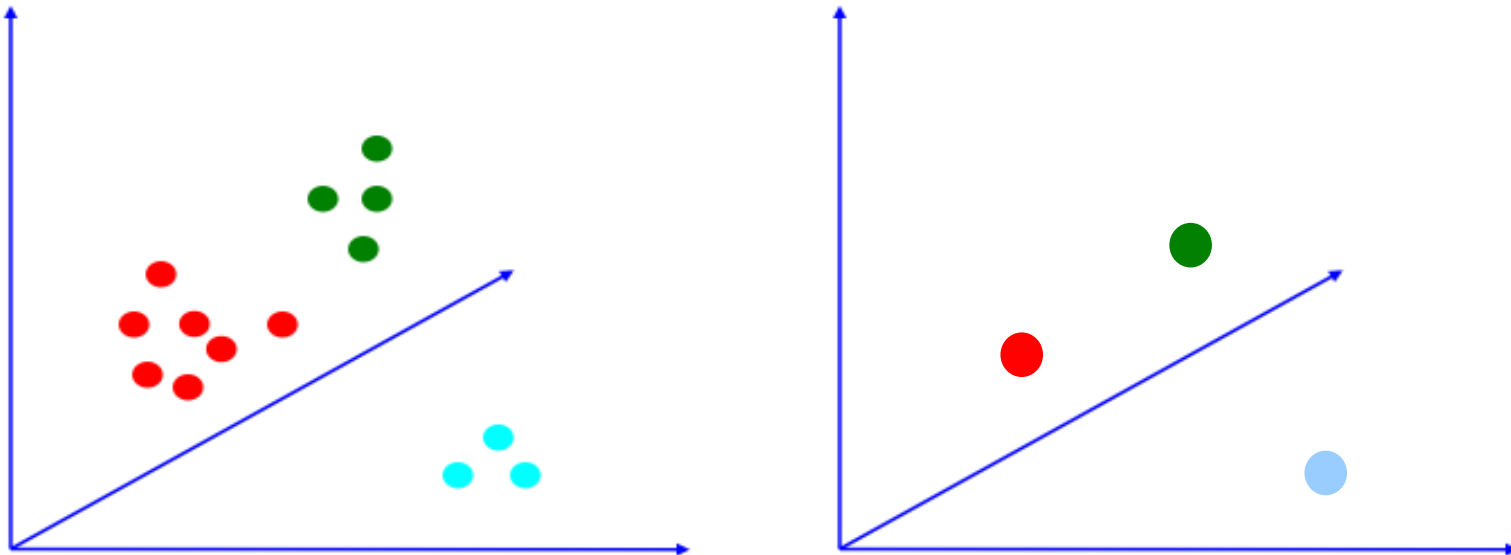
- At detected interest points
- Uniform sampling
- Random sampling
- Segmentation based

Descriptor: Typically SIFT or another similar descriptor



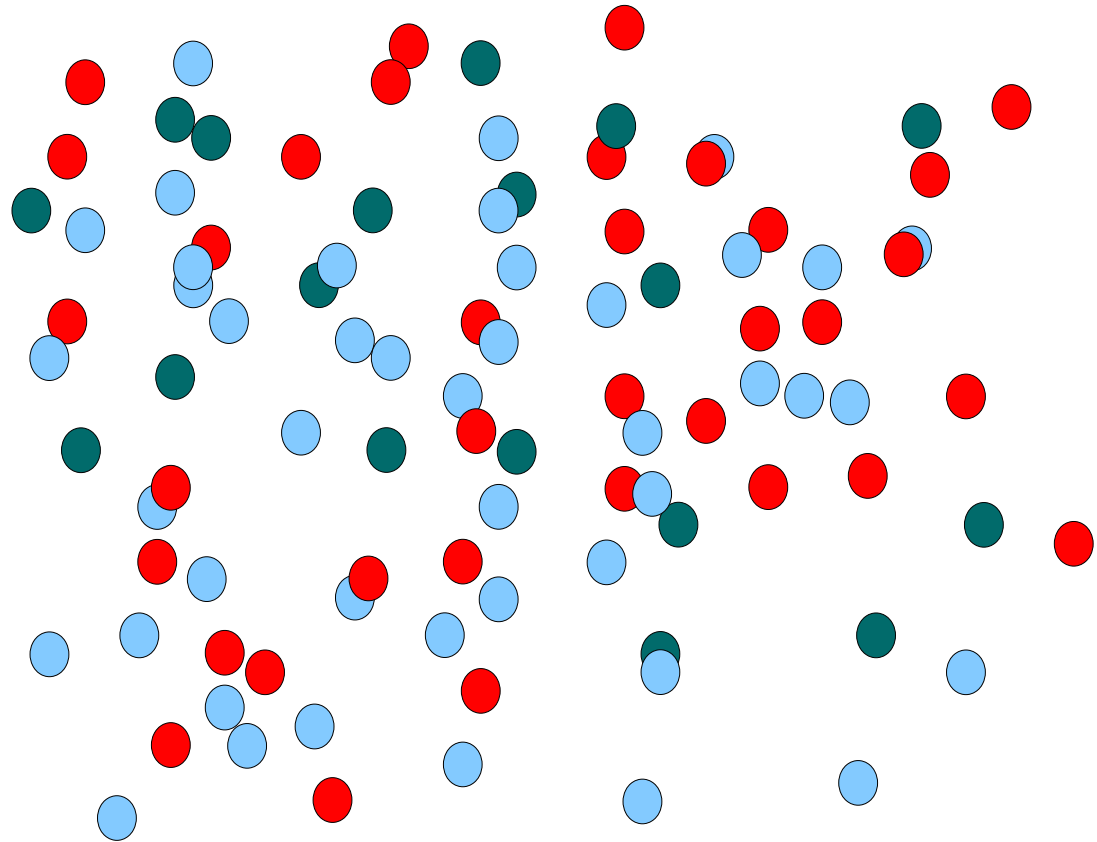
From Features to Codebook

- Map very similar detectors together to reduce the effective dimensionality.
- Quantize the words via clustering.
 - Clustering: K-Means
 - Hierarchical K-Means: To build a vocabulary tree
- Cluster centers: **visual words** of the vocabulary



Hierarchical K-Means

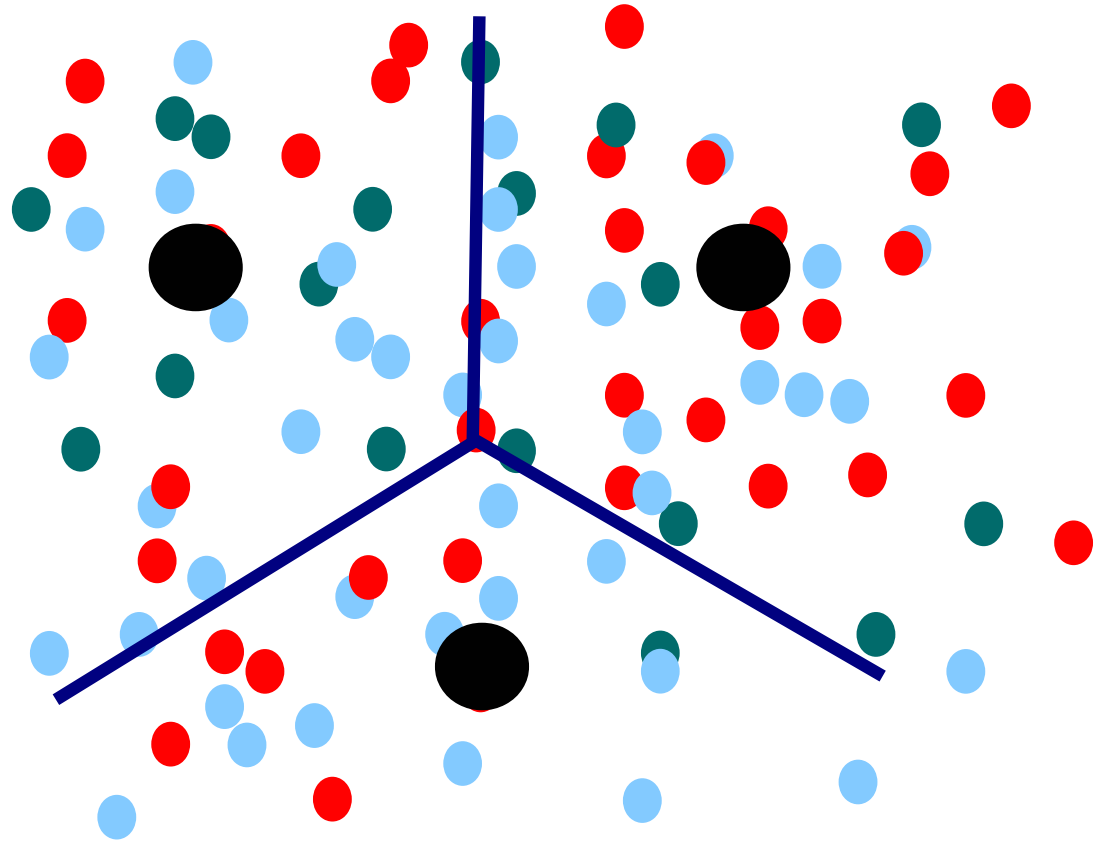
Put all points in the
descriptor space



Hierarchical K-Means

Put all points in the
descriptor space

Divide into k clusters



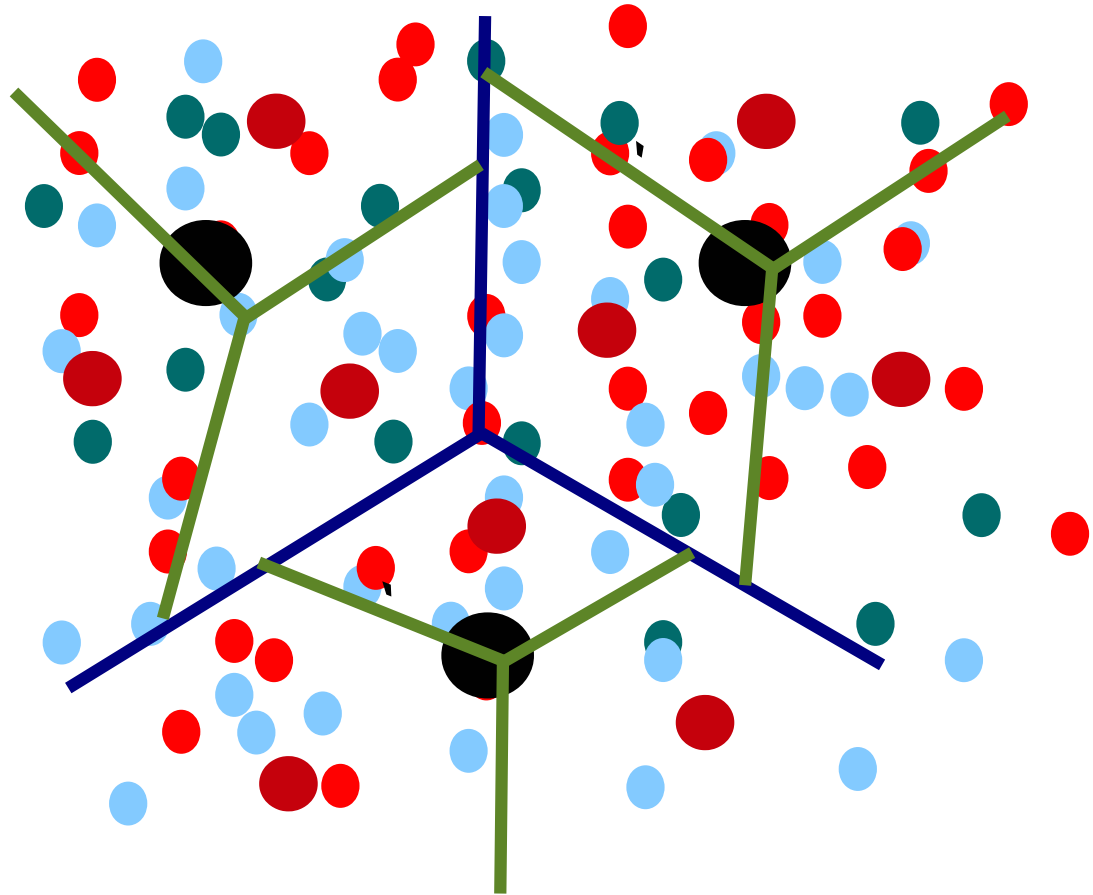
Hierarchical K-Means

Put all points in the
descriptor space

Divide into k clusters

Divide each cluster
into k more clusters

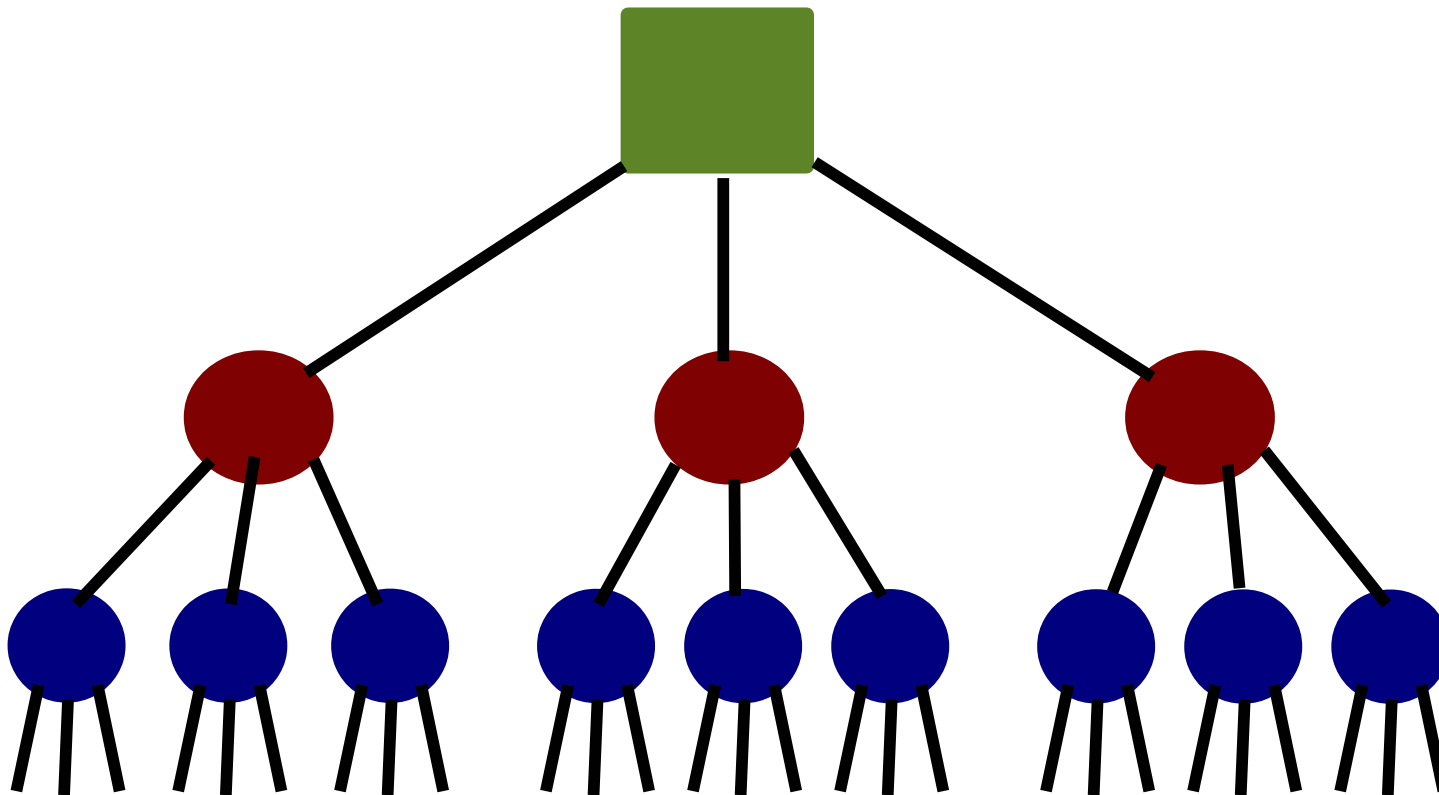
... and so on



Vocabulary Tree

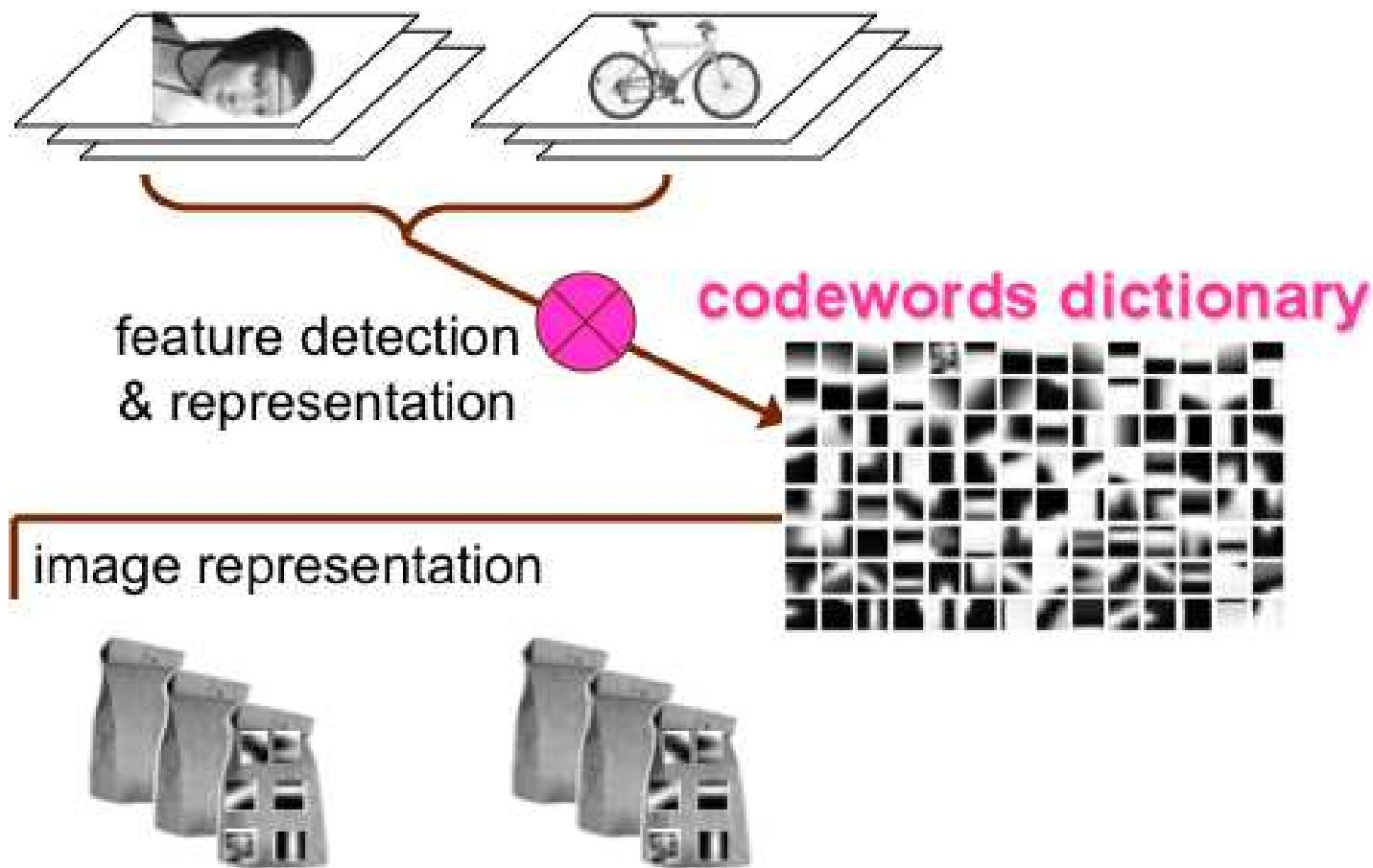
Visual words at the leaf level.

A feature vector is classified as one by following the tree path



Representation: Summary

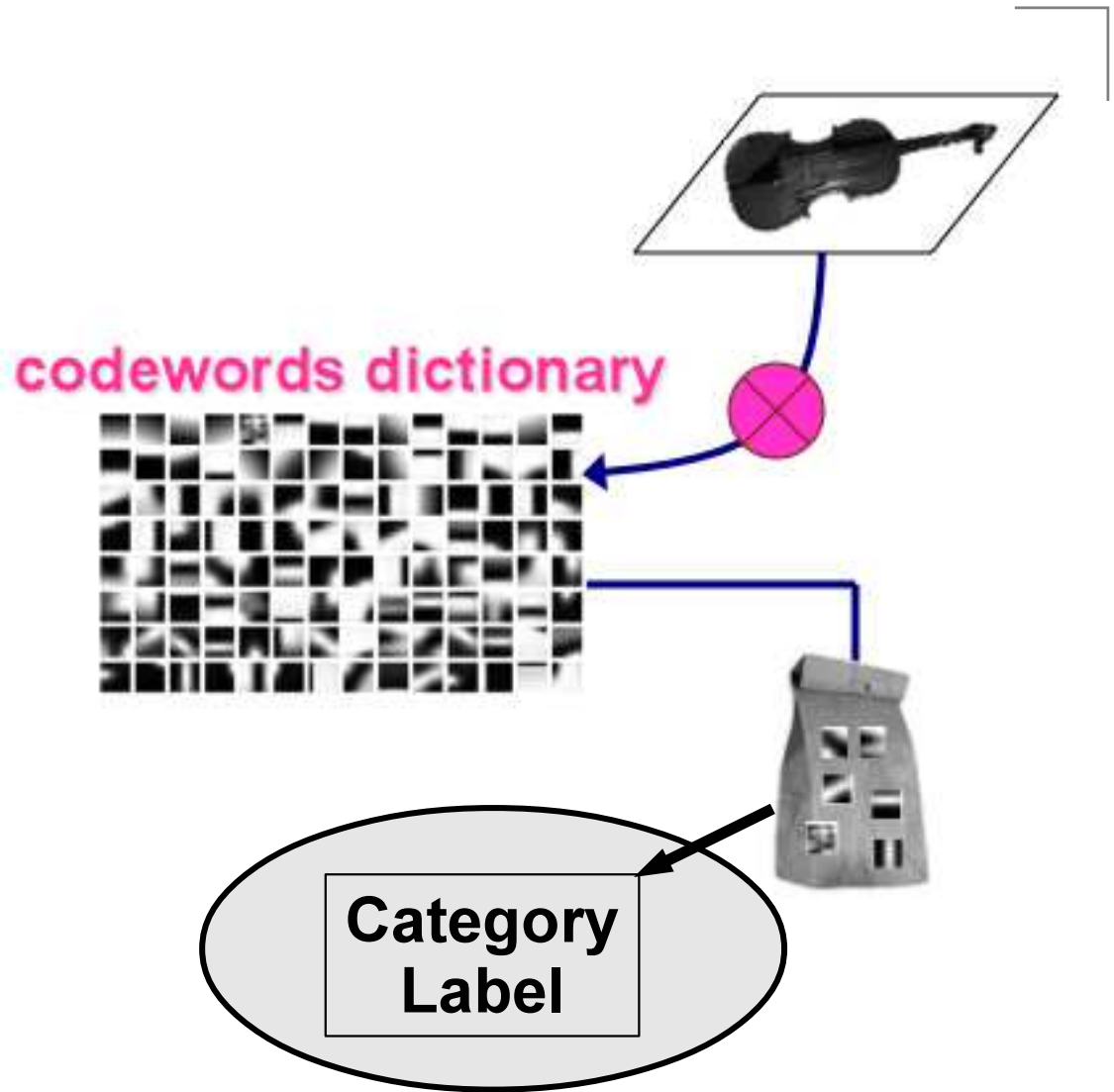
Histogram of words, length equal to the vocabulary size



Learning & Recognition

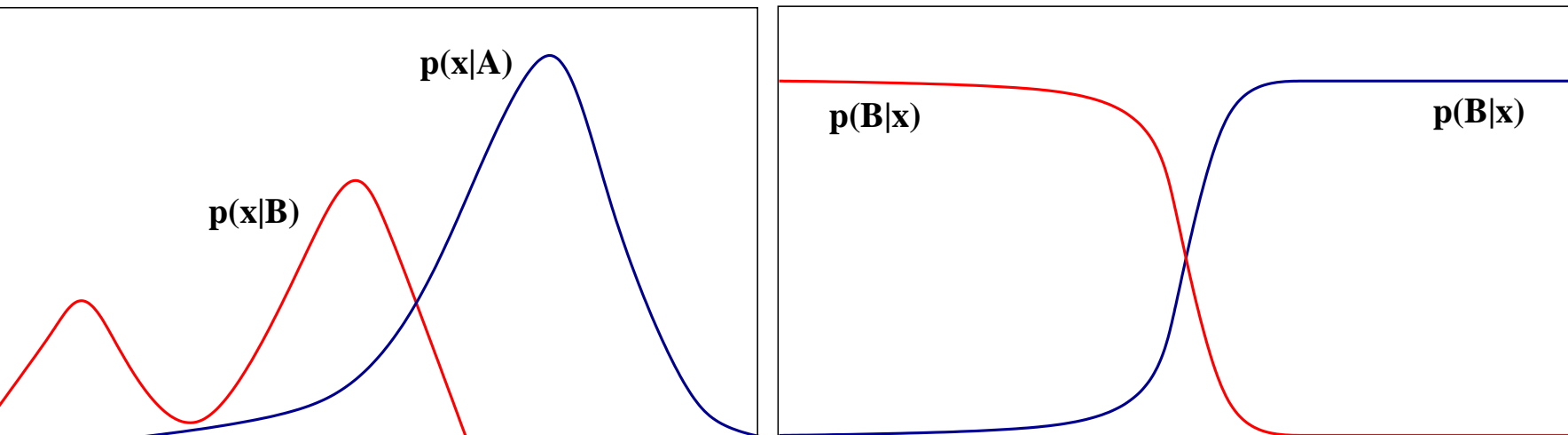
What do we do with the representation?

- Learn a model for the objects based on the labelled training data
- Apply it to new objects to be recognized
- Search/retrieval: A special case where ranked retrieval is expected



Recognition

- Direct matching using visual words
 - Treat all features that map to a word as matches
 - Apply other consistency checks to confirm
- Generative models: Model how an object generates an image and features in it
- Discriminative models: Distinguish between objects from the evidence

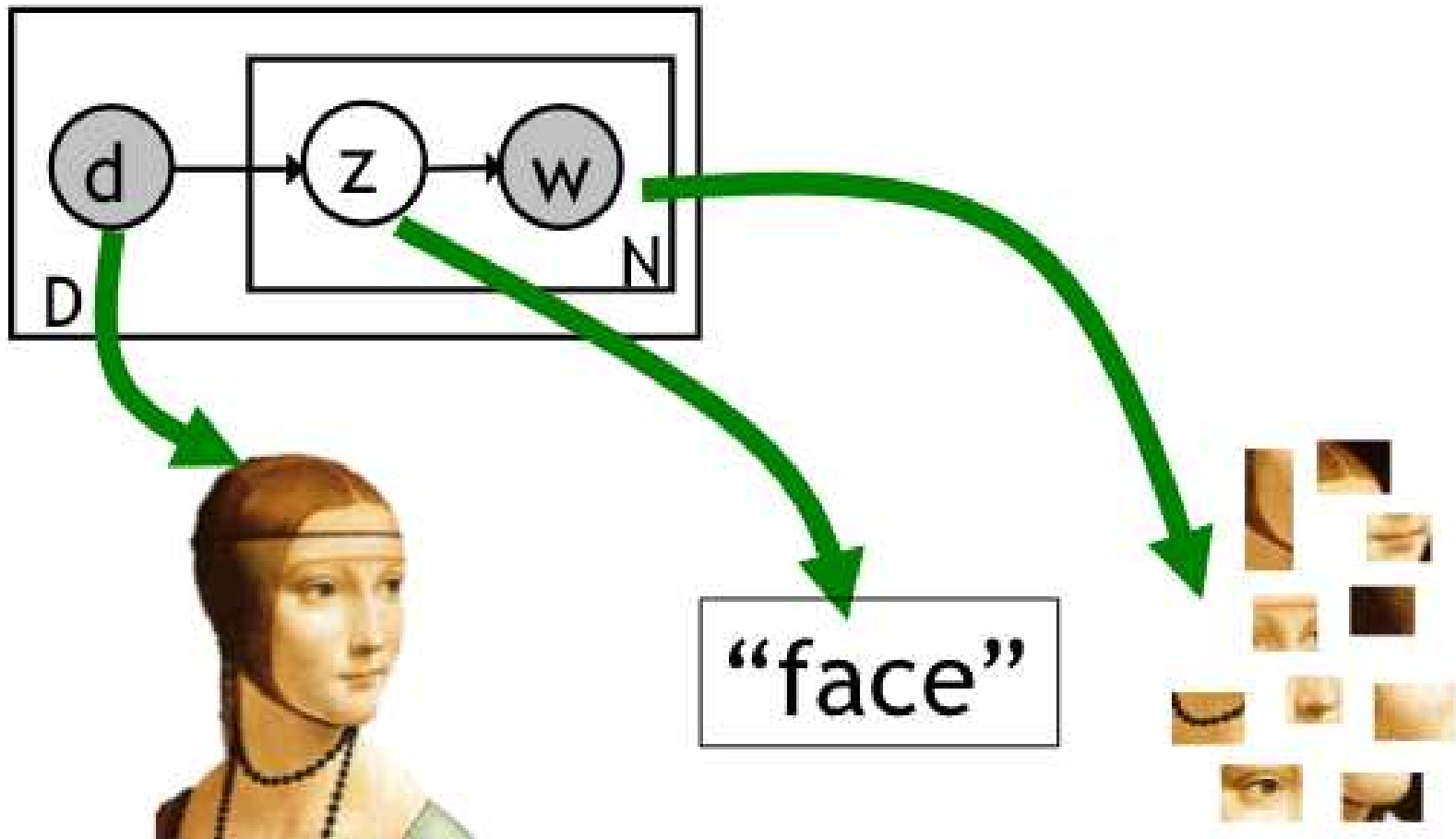


Direct Matching

- Quantizing to the same visual word is matching of features!
- Nister and Stewenius, CVPR2006
 - Matched and retrieved a CD by showing its cover image from a collection of 50K of CDs within a second
 - Match visual words, apply RANSAC to look for planarity
- Video Google: Match histograms of visual words directly to retrieve and rank images

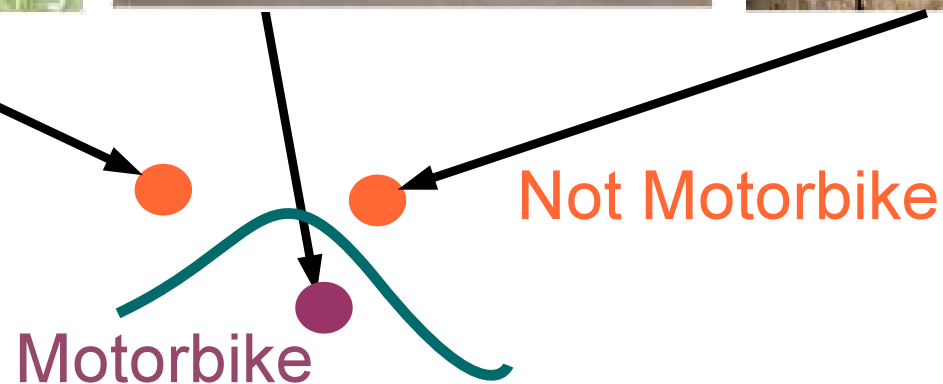
Generative Models

- Naive Bayes Classifier
- Probabilistic Latent Semantic Analysis (pLSA): How a model generates a set of features



Discriminative Models

- Nearest Neighbor
- SVM Classifier
- Neural Networks, etc.

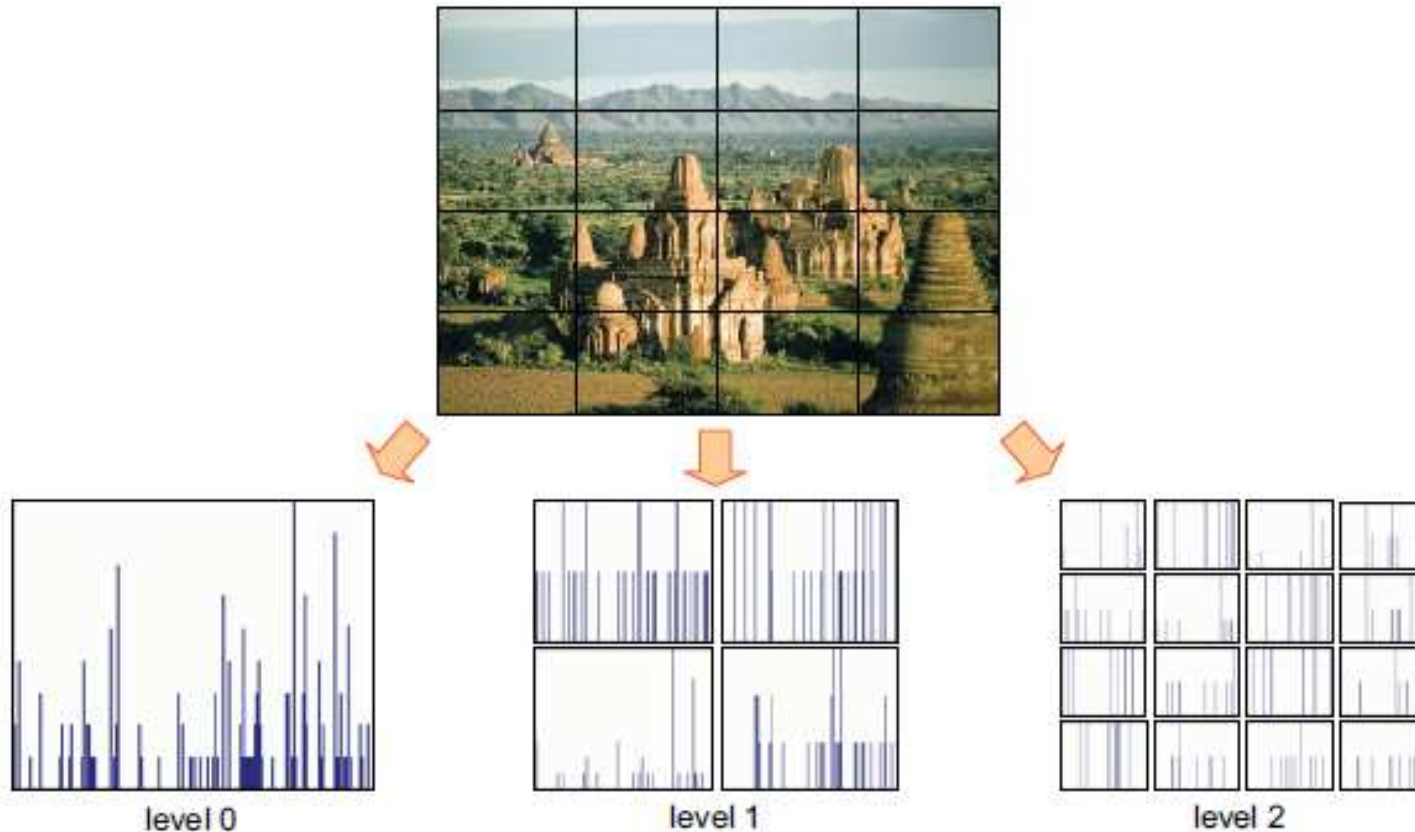


Bringing in Spatial Information

- Feature level spatial arrangement
 - Look for spatial consistency like Video Google
 - Proximity or a model like homography
- Enhanced histograms that include spatial pyramid information
 - Spatial Pyramid by Lazebnik et al. CVPR 2006
 - Pyramid HOG by Bosch and Zisserman, CIVR 2007
 - Matching such representations: Pyramid Match Kernel by Grauman and Darrell, ICCV 2005

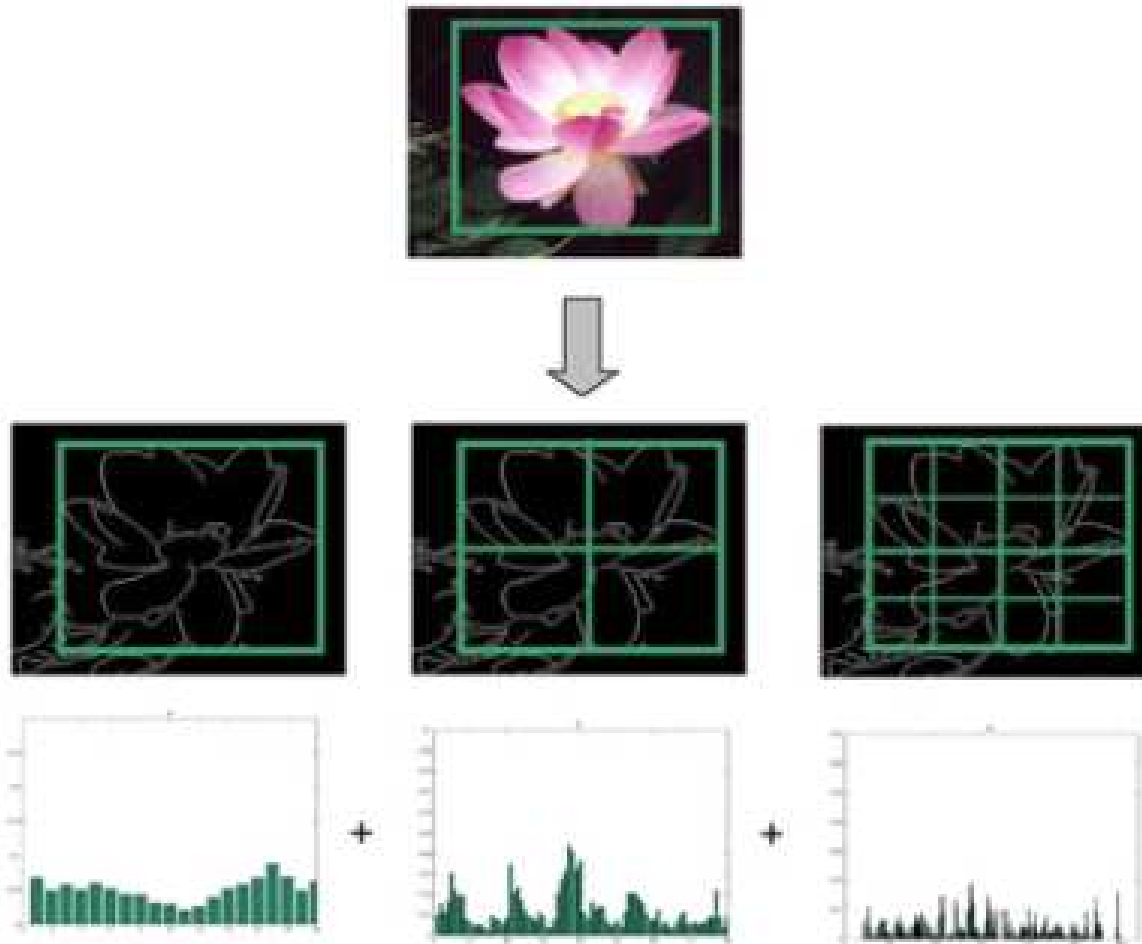
Spatial Pyramid using SIFT

- Divide the image into hierarchical levels like a quadtree
- Evaluate histograms for each level and cell
- Concatenate them to form a long vector



Spatial Pyramid using HOG

- Divide the image into hierarchical levels like a quadtree
- Evaluate HoG for each level/cell and concatenate them



Pyramid Match Kernel

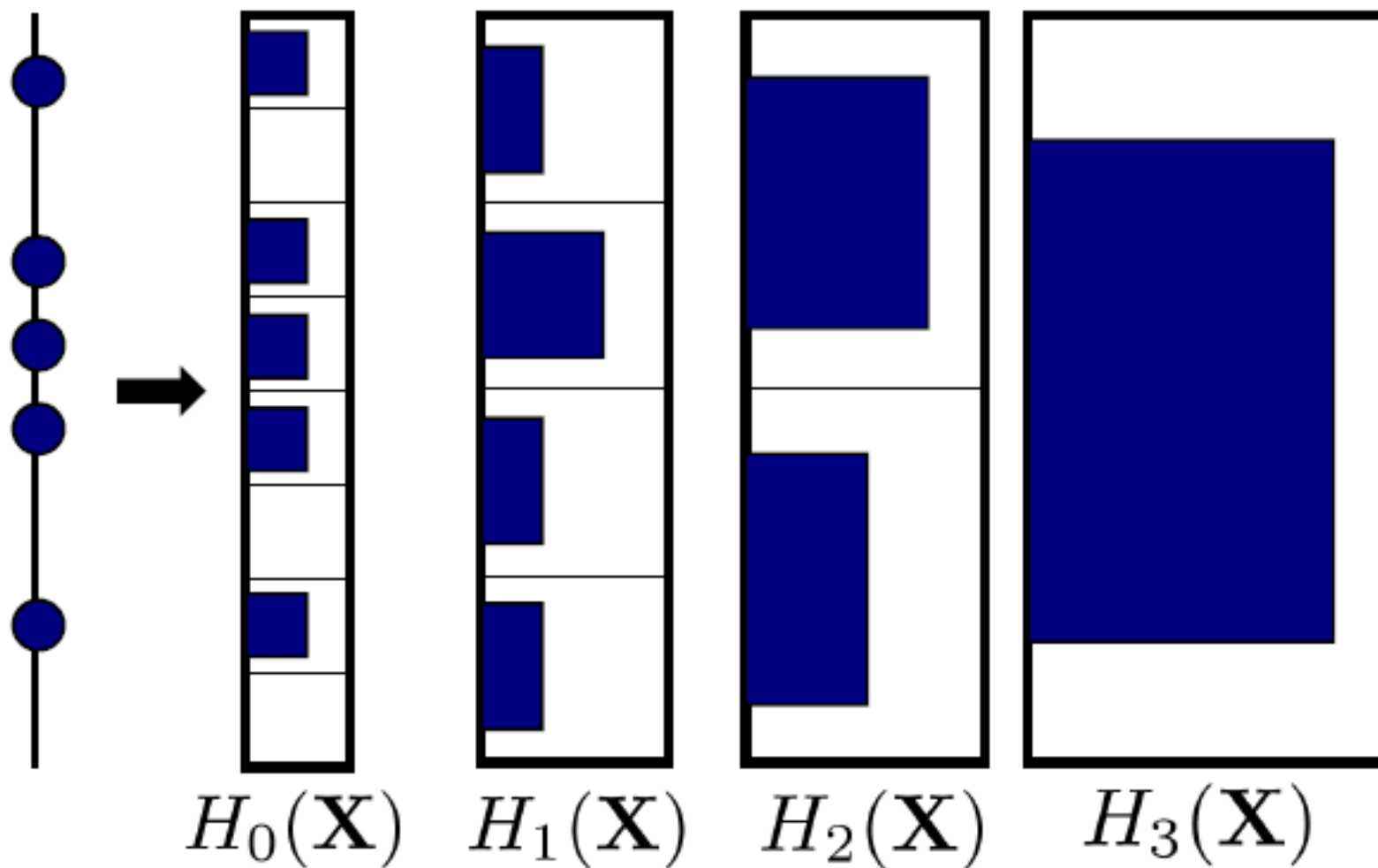
- How do we match a pyramid of histograms?
- Match points in bins by levels, starting with finest
- Weights reduce exponentially by the level
- Match score (kernel) $K(P(X), P(Y))$ is:

$$= \sum_{i=0}^l \frac{1}{2^i} (\cap(H_i(X), H_i(Y)) - \cap(H_{i-1}(X), H_{i-1}(Y)))$$

where

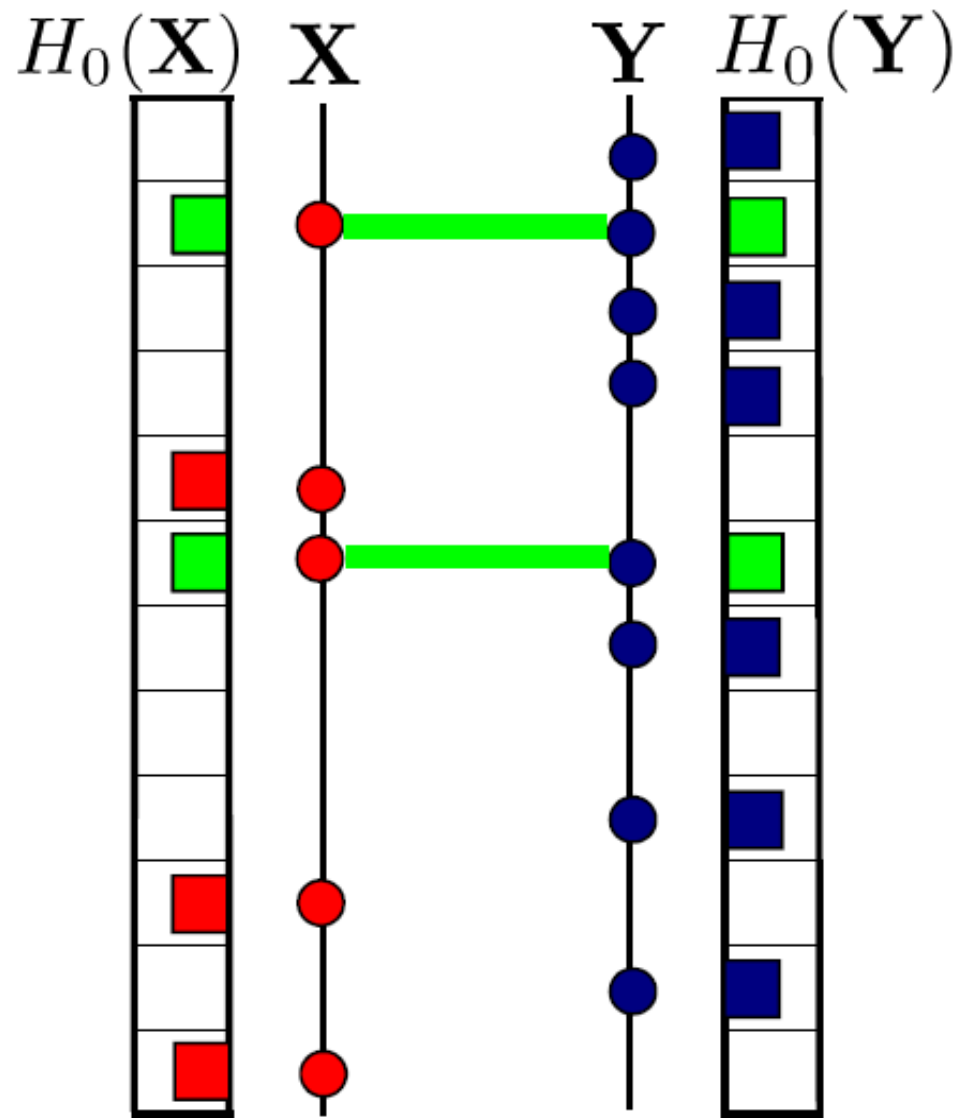
$$\cap(H_i(X), H_i(Y)) = \sum_{j=0}^r \min(H(X)_j, H(Y)_j)$$

Pyramid Representation



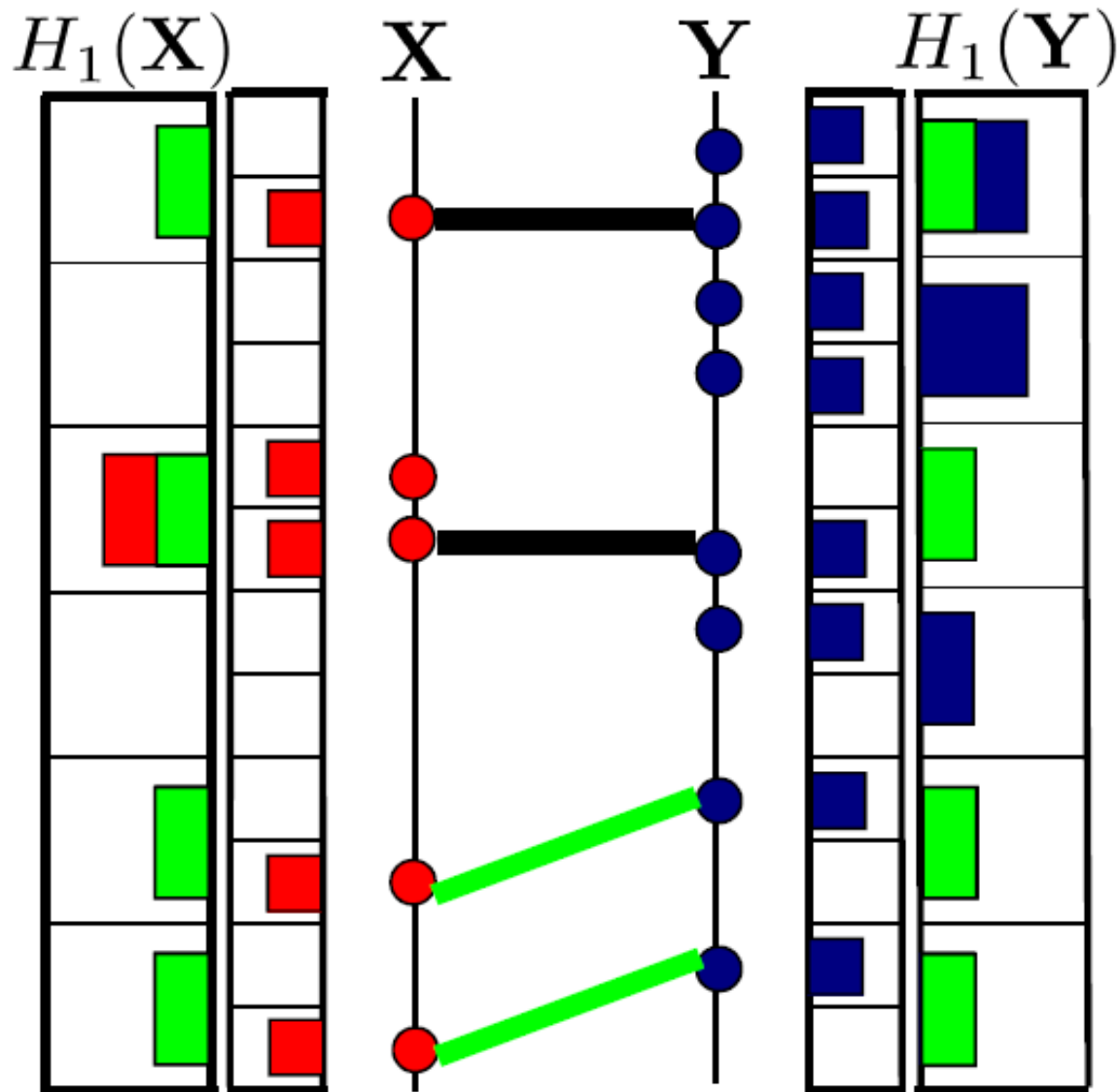
Multiple levels of the pyramid together: $P(X)$

Matching at Level 0

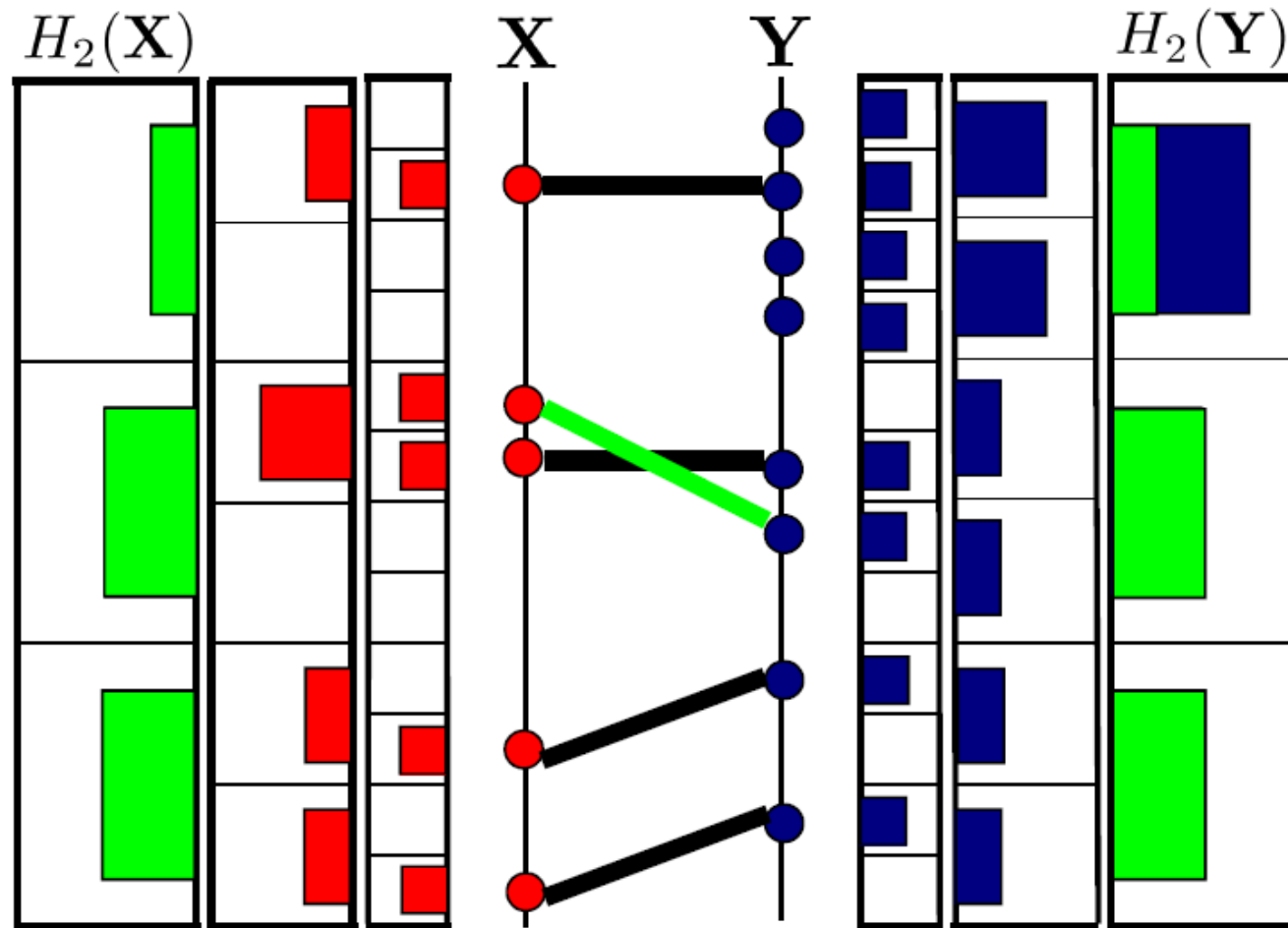


Score = 2

Matching at Level 1



Matching at Level 2



Score = 1. Total score: $2 + 2/2 + 1/4 = 3.25$

Optimal match: $O(n^3)$. Pyramid match: $O(n)$ where n is the number of feature vectors

Applying it to Spatial Pyramids

- Each visual word (or histogram bin) is a pyramid
- When two features fall in a cell (i.e., identical visual words in the present range/scale of search), they are deemed matched
- The finest level matches get a weight of 2. Coarser levels i , $i > 0$ get a weight of $\frac{1}{2^i}$
- Sum the scores for each visual word across pyramid levels. This can be done with one long vector consisting of all levels
- Sum the scores for all visual words together to get a matching score for the whole image

Much better recognition performance even with 3 levels and 400 visual words!! [Lazebnik 2006]

BoW Recognition: Summary

- + Flexible to changes in geometry, viewpoint, deformations
- + Compact, fixed-size summary of content
- + Gives good results
- Ignores geometry, which needs a separate verification step
- Background and foreground mixed in the representation
- Interest points or regular samplign? No clear answer
- Optimal vocabulary? How to form it?



Thank You!

Thanks to the Computer Vision community worldwide
for many of the figures!!