

Object Recognition

Object Recognition is a big family of computer vision algorithms. [Computer Vision Tasks](#) gives an answer to what it means. The term is actually quite ambiguous

History of ideas in recognition

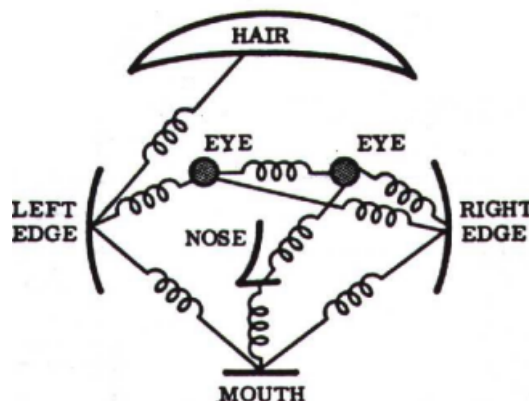
- 1960s – early 1990s: the geometric era
- 1990s: appearance-based models
- Mid-1990s: sliding window approaches
 - Turk and Pentland, 1991
 - Belhumeur, Hespanha, & Kriegman, 1997
 - Schneiderman & Kanade 2004
 - Viola and Jones, 2000



- Schneiderman & Kanade, 2004
- Argawal and Roth, 2002
- Poggio et al. 1993



- Late 1990s: local features
- Early 2000s: parts-and-shape models



- Object as a set of parts

- Relative locations between parts
- Appearance of part
- Mid-2000s: bags of features
- Present trends: Combined local and global methods, context, and deep learning

Image Classification



Assume given a set of discrete labels, classify images into the labels.

Data-driven Approach

- Collect a database of images with labels
- Use ML to train an image classifier
- Evaluate the classifier on test images

Bag of Features

The general idea is to represent a data item as a histogram over features.

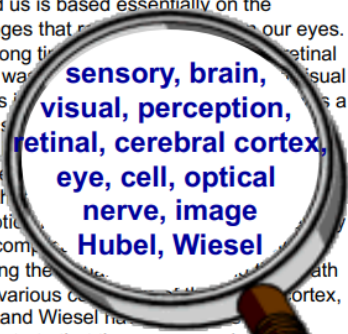
Image/Object	Bag of Word
	

A not so crazy assumption here is:
spatial information of local features can be ignored for object recognition (i.e., verification).

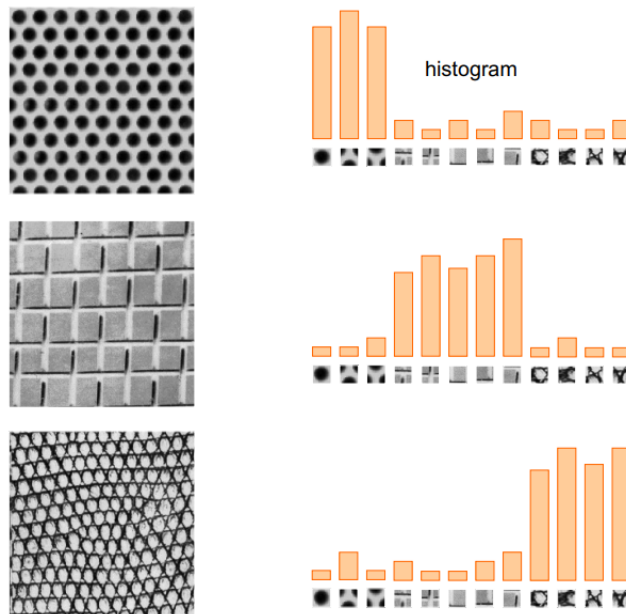
Examples:

- Orderless Document representation: frequencies of words from a dictionary

Of all the sensory impressions proceeding to the brain, the visual experiences are the dominant ones. Our perception of the world around us is based essentially on the messages that reach our eyes. For a long time, the retinal image was considered as a movie screen. It is now discovered that the visual system knows that the perception is more complex. Following the path to the various centers of the cortex, Hubel and Wiesel have demonstrated that the message about the image falling on the retina undergoes a piecewise analysis in a system of nerve cells stored in columns. In this system each cell has its specific function and is responsible for a specific detail in the pattern of the retinal image.



- Texture recognition:



An image is a vector of counts over each feature

$$\mathbf{v}_d = [n(w_1), n(w_2), \dots, n(w_T)]$$

where $n(\cdot)$ counts the number of occurrences, d is the index of the image, T is the number of features and w_i is the word. \mathbf{v}_d is essentially a histogram over words.

tf-idf - Wikipedia (Term Frequency - Inverse Document Frequency)

Term frequency

Suppose we have a set of English text documents and wish to rank them by which document is more relevant to the query, "the brown cow". A simple way to start out is by eliminating documents that do not contain all three words "the", "brown", and "cow", but this still leaves many documents. To further distinguish them, we might count the number of times each term occurs in each document; the number of times a term occurs in a document is called its **term frequency**. However, in the case where the length of documents varies greatly, adjustments are often made (see definition below). The first form of term weighting is due to [Hans Peter Luhn](#) (1957) which may be summarized as:

The weight of a term that occurs in a document is simply proportional to the term frequency.

Inverse document frequency

Because the term "the" is so common, term frequency will tend to incorrectly emphasize documents which happen to use the word "the" more frequently, without giving enough weight to the more meaningful terms "brown" and "cow". The term "the" is not a good keyword to distinguish relevant and non-relevant documents and terms, unlike the less-common words "brown" and "cow". Hence, an *inverse document frequency* factor is incorporated which diminishes the weight of terms that occur very frequently in the document set and increases the weight of terms that occur rarely.

Karen Spärck Jones (1972) conceived a statistical interpretation of term-specificity called Inverse Document Frequency (idf), which became a cornerstone of term weighting:

The specificity of a term can be quantified as an inverse function of the number of documents in which it occurs.

Not all words are created equal, weight each word by a heuristic

$$\mathbf{v}_d = [n(w_1)\alpha_1, n(w_2)\alpha_2, \dots, n(w_T)\alpha_T]$$

where

$$\alpha_i = \log \left\{ \frac{D}{\sum_{d'} \mathbb{1}\{w_i \in d'\}} \right\}$$

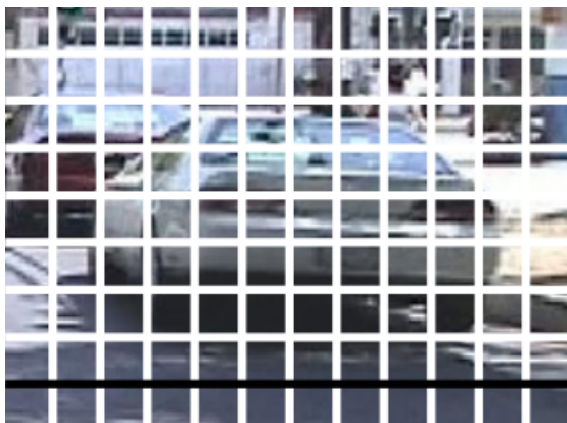
is the inverse document frequency, and D is the total number of images.

Standard Pipeline

1. Feature extraction

Extract features from images

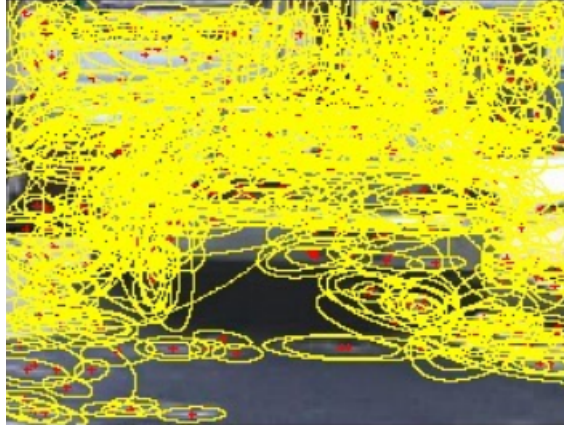
- Regular grid



- Interest point detector



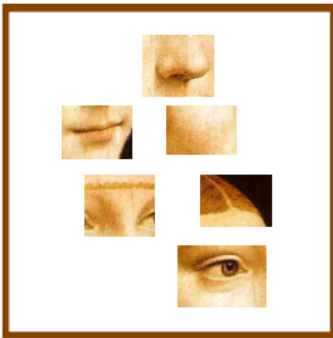
- Other methods



- Random sampling (Vidal-Naquet & Ullman, 2002)
- Segmentation-based patches (Barnard et al. 2003)

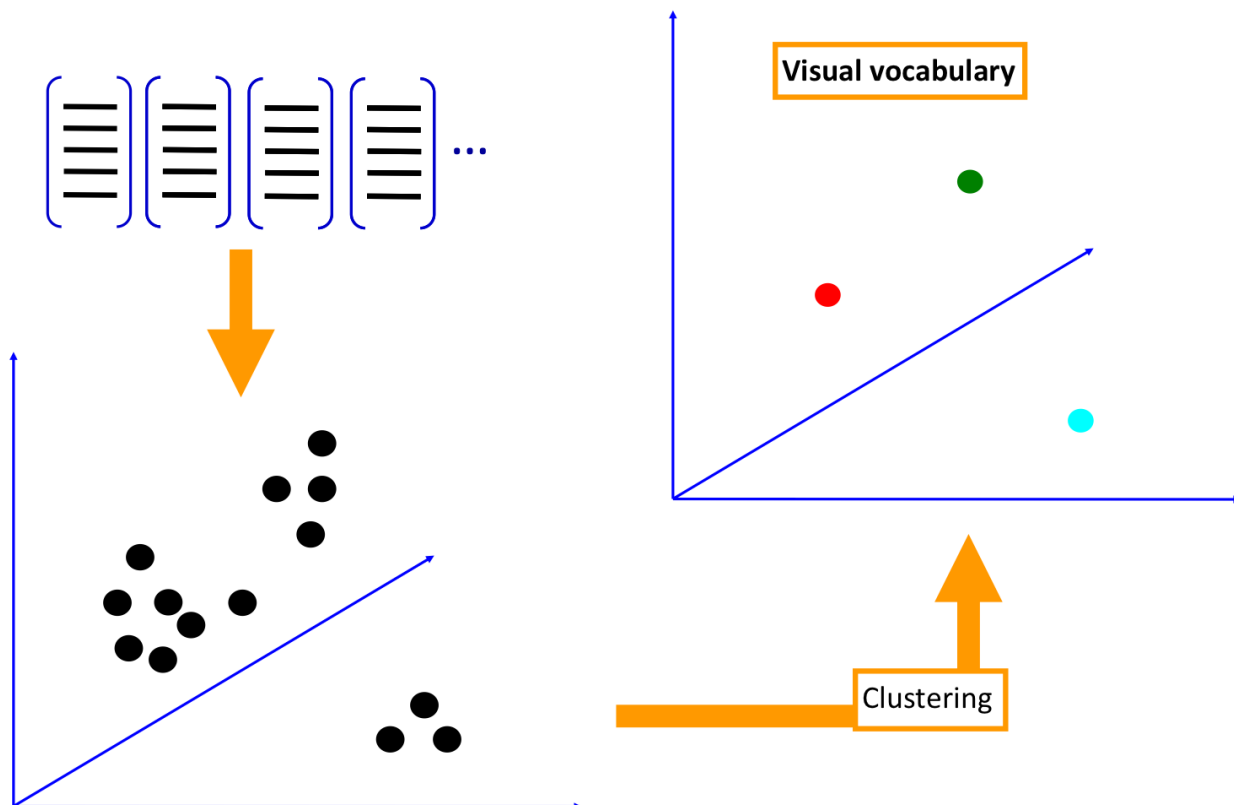
Note that the features are not image patches.

extract features (e.g., SIFT) from images



2. Dictionary learning

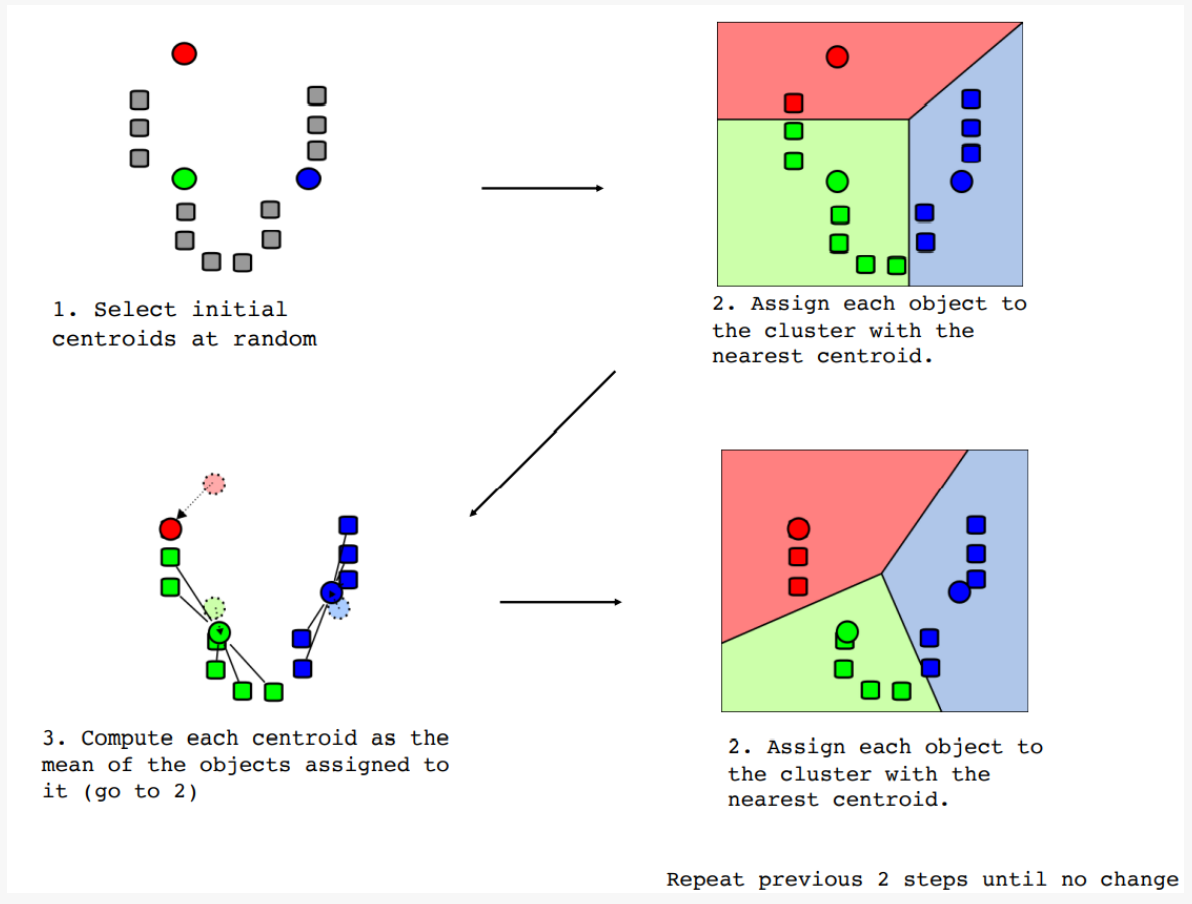
Learn visual vocabulary using clustering



K-means clustering

Given k

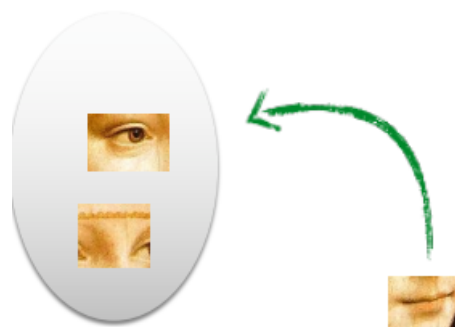
1. Select initial k centroids at random
2. Assign each object to the cluster with the nearest centroid
3. Computer each centroid as the mean of the objects assigned to it
4. Repeat previous 2 steps until no change



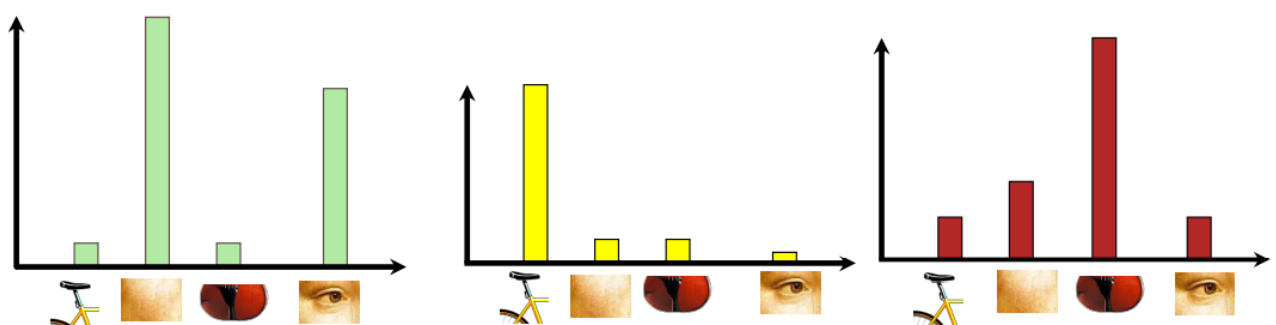
3. Encode

Build bags-of-words(BOW) vectors for each image

- Quantization: image features gets associated to a visual word(nearest cluster center)



- Histogram: count the number of visual word occurrences



4. Classify

Train and test data using BOWs

Pros:

- flexible to geometry / deformations / viewpoint
- compact summary of image content
- provides fixed dimensional vector representation for sets
- very good results in practice

Cons:

- background and foreground mixed when bag covers whole image
- optimal vocabulary formation remains unclear
- basic model ignores geometry

Specific Object Recognition

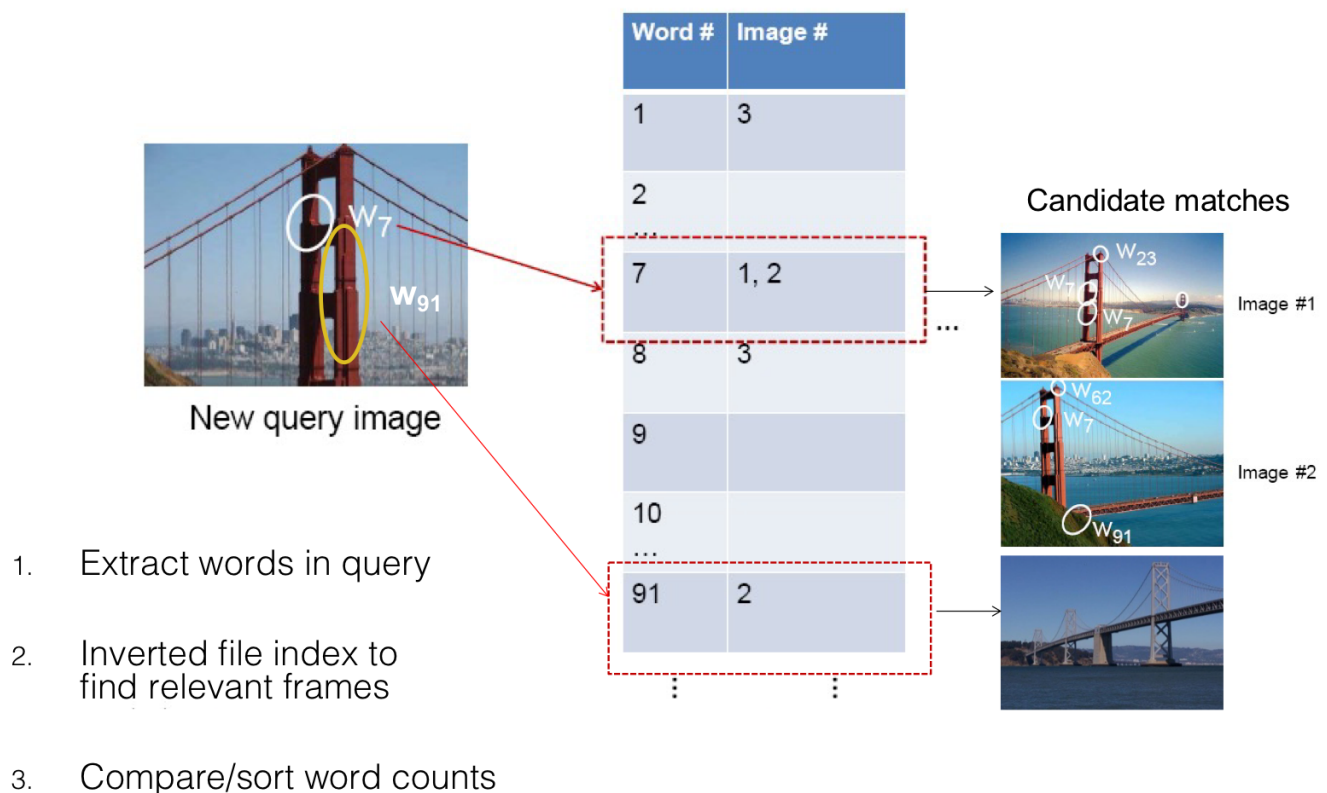
Given an image region, we want to quickly find images in a large database that match the given image region.

Fast Lookup: Inverted Index

Find all images in which a feature occurs

Query inverted index:

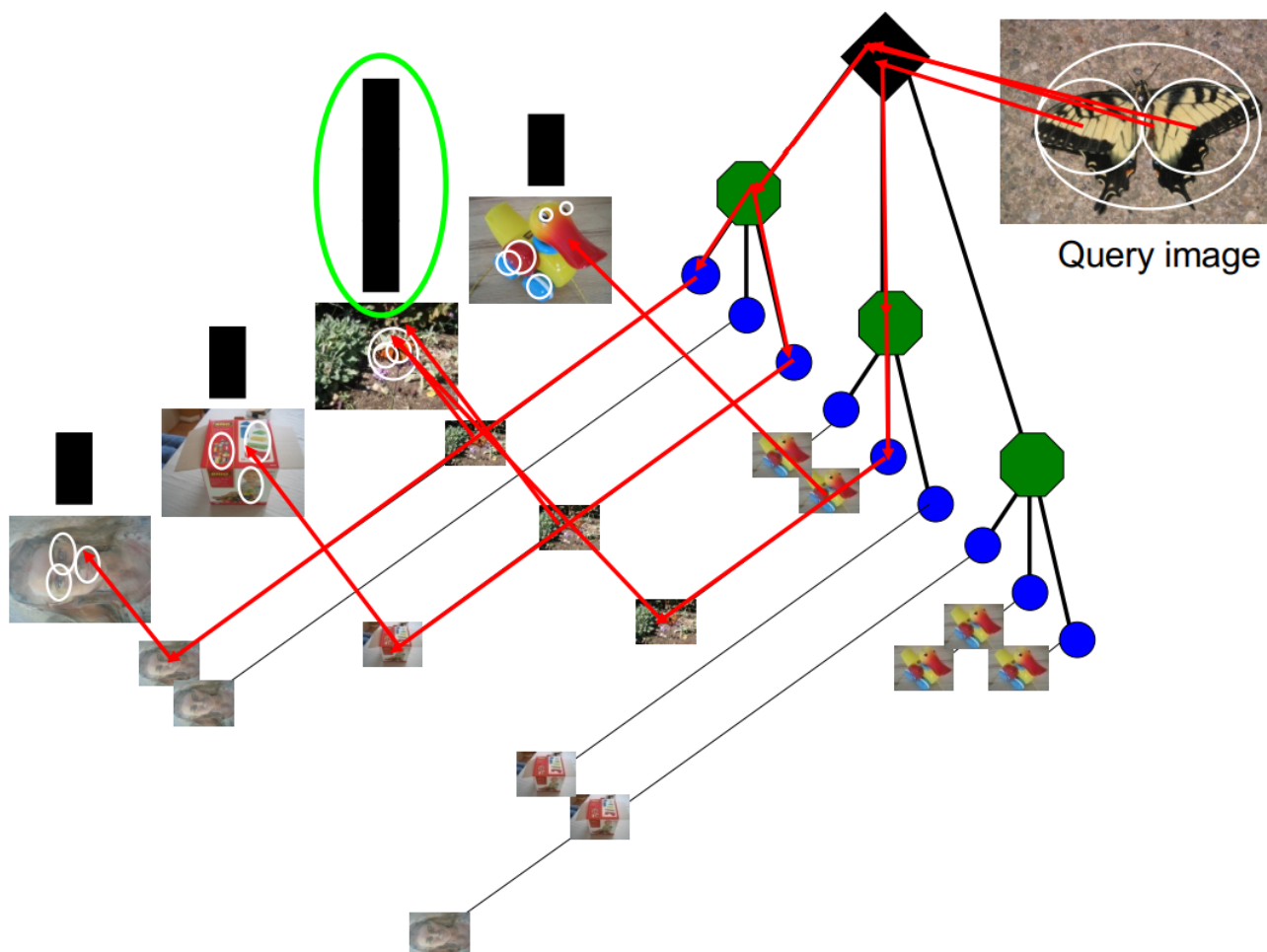
1. Extract words in query
2. Inverted file index to find relevant frames
3. Compare/sort word counts



The method requires sparsity. If most images contain most words, then we are not better off than exhaustive search.

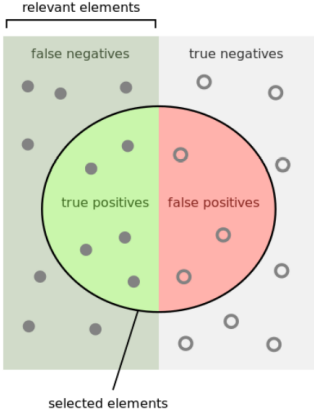
Vocabulary Tree

1. Build vocabulary tree recursively
2. Each leaf has inverted index
3. Build inverted index
4. Match



Score the Result

<p>The diagram shows a Venn diagram with two overlapping circles. The left circle is green and labeled "true positives". The right circle is red and labeled "false positives". The area where the two circles overlap is labeled "selected elements". The area outside the green circle but inside the red circle is labeled "false negatives". The area outside both circles is labeled "true negatives". The label "relevant elements" is placed above the green circle.</p>	<p>Precision</p>	<p>Recall</p>
<p>Figure</p>	<p>Precision = $\frac{\text{true positives}}{\text{true positives} + \text{false positives}}$</p>	<p>Recall = $\frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$</p>

	Precision	Recall
Formula	$\text{Precision} = \frac{tp}{tp + fp}$	$\text{Recall} = \frac{tp}{tp + fn} = \frac{\text{relevant}}{\text{returned}}$

- True positive (tp) → correct attribution
- True negative (tn) → correct rejection
- False positive (fp) → incorrect attribution
- False negative (fn) → incorrect rejection

Object Category Detection

Object Detection algorithms act as a combination of image classification and object localization.

1. Specify Object Model

We need an object model to describe the object.

- Statistical template in bounding box
- Articulated parts model
- Hybrid template/parts model
- Deformable 3D model

2. Generate Hypotheses/Proposals

Propose an alignment of the model to the image.

- 2D template model/ sliding window
Test patch at each location and scale, each window is separately classified
- Voting from patches/keypoints
- Region-based proposal

3. Score Hypotheses

Mainly gradient-based features, many classifiers.

4. Resolve Detections

Rescore each proposed object based on context information.

- Non-maximum Suppression

Typical Object detection pipeline has one [component](#) for generating proposals for classification. Proposals are nothing but the candidate regions for the object of interest. Most of the approaches employ a sliding window over the feature map and assigns foreground/background scores depending on the features computed in that window. The neighborhood windows have similar scores to some extent and are considered as candidate regions. This leads to hundreds of proposals. As the proposal generation method should have high recall, we keep loose constraints in this stage. However processing these many proposals all through the classification network is cumbersome. This leads to a technique which filters the proposals based on some criteria (which we will see soon) called Non-maximum Suppression.

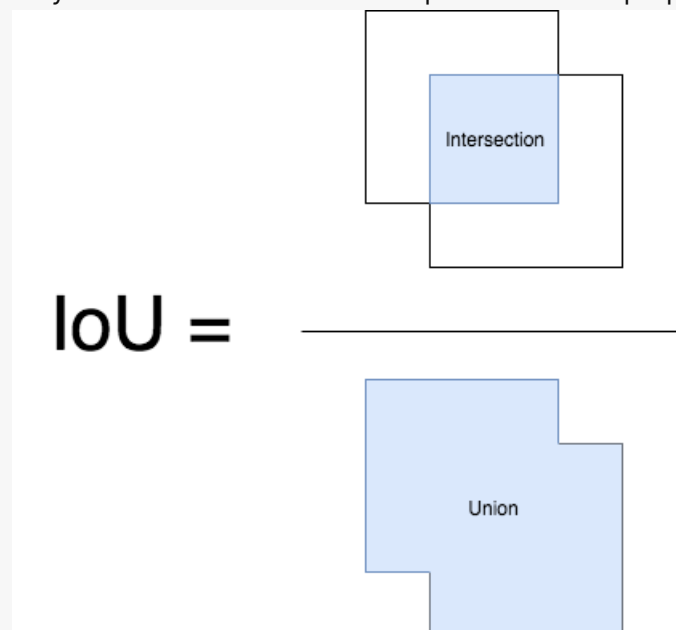
Input: A list of Proposal boxes B, corresponding confidence scores S and overlap threshold N.

Output: A list of filtered proposals D.

Algorithm:

1. Select the proposal with highest confidence score, remove it from B and add it to the final proposal list D. (Initially D is empty).
2. Now compare this proposal with all the proposals — calculate the IOU (Intersection over Union) of this proposal with every other proposal. If the IOU is greater than the threshold N, remove that proposal from B.
3. Again take the proposal with the highest confidence from the remaining proposals in B and remove it from B and add it to D.
4. Once again calculate the IOU of this proposal with all the proposals in B and eliminate the boxes which have high IOU than threshold.
5. This process is repeated until there are no more proposals left in B.

IOU calculation is actually used to measure the overlap between two proposals.



- Context/reasoning
 - Via geometry
 - Via known information or prior distributions