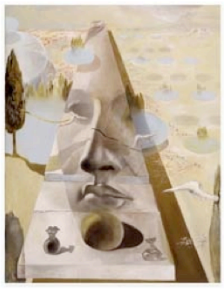
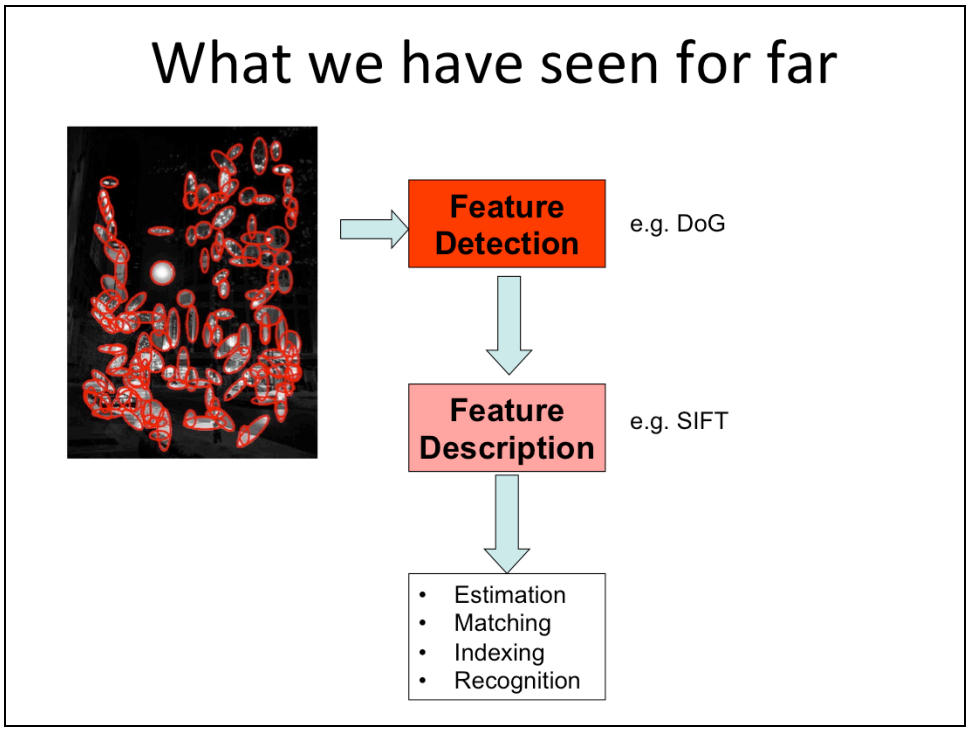
Lecture 11
Visual recognition



- An introduction to recognition
- Image classification – the bag of words model

[FP] – Chapters 6 (sec. 6.2)
[FP] – Chapters 16 (sec. 16.1)
[FP] – Chapters 17 (sec. 17.1)



As discussed in lecture 10, features detectors and descriptors provide the building blocks for higher level tasks such as

- Estimation
- Matching
- Indexing
- Recognition

The focus of today lecture will be on visual recognition

What's visual recognition?

Visual recognition is about recognizing and identifying the key semantic aspects of a scene from images.

This sounds like a rather generic problem definition. And let me discuss now a number of specific tasks that are concerned with visual recognition.

Classification:

Does this image contain a building? [yes/no]

One key task is image classification. The goal of classification is to answer a binary question such as: does this image contain a building or not? In a classification problem, we are not interested in localizing where an object in the image is as long as we can tell whether the object is in the image or not.

Classification:

Is this an beach?

Another example: "Is this an image of a beach?" Well, this photo is obviously not a beach.

This is a key ingredient in image search engines such as those introduced by Google, Flickr, or other related platforms.

Additional notes by C. Kao:

How does this work? Users use search engine by giving queries, but how does a computer find images that satisfies the query? It can pre-process the image database, and classify images with different tags. For example, here, the user search for "street". The first and second image seems alright, but the last image is a motorcycle. It can be seen as a misclassification from the recognition algorithm. Moreover, you can organize your huge photo collection, and search for something you are interested. For instance, you can show your friends only photos that are related to trees, leaves, etc. Then you will hope the image software can search them for you.

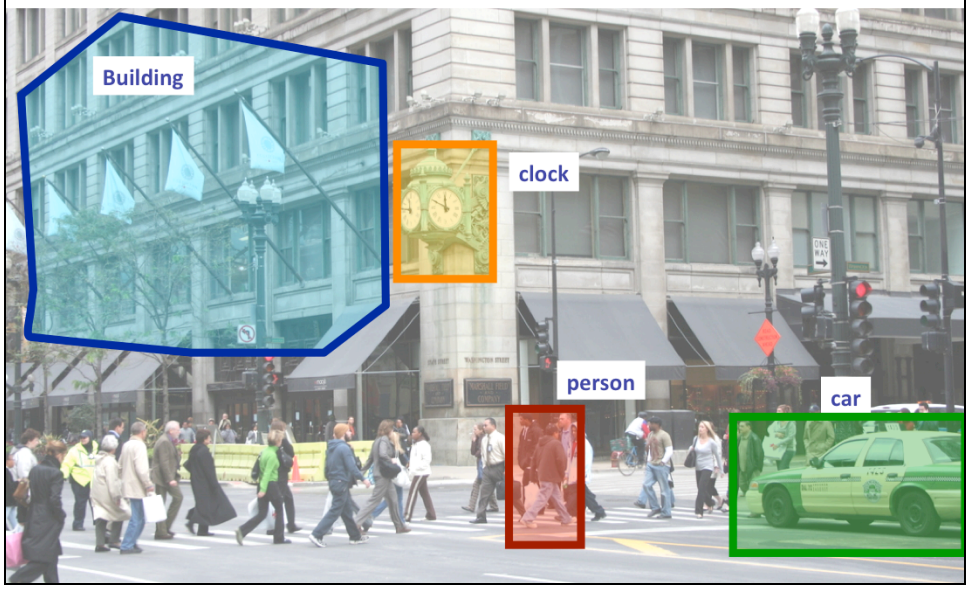
Detection:

Does this image contain a car? [where?]

Another important recognition task is detection: In detection, not only do we want to tell whether an object (say a car) is in the image or not, but also tell where it is. An object can be typically localized in the image by using a bounding box such as the one shown in the slide.

Detection:

Which object does this image contain? [where?]

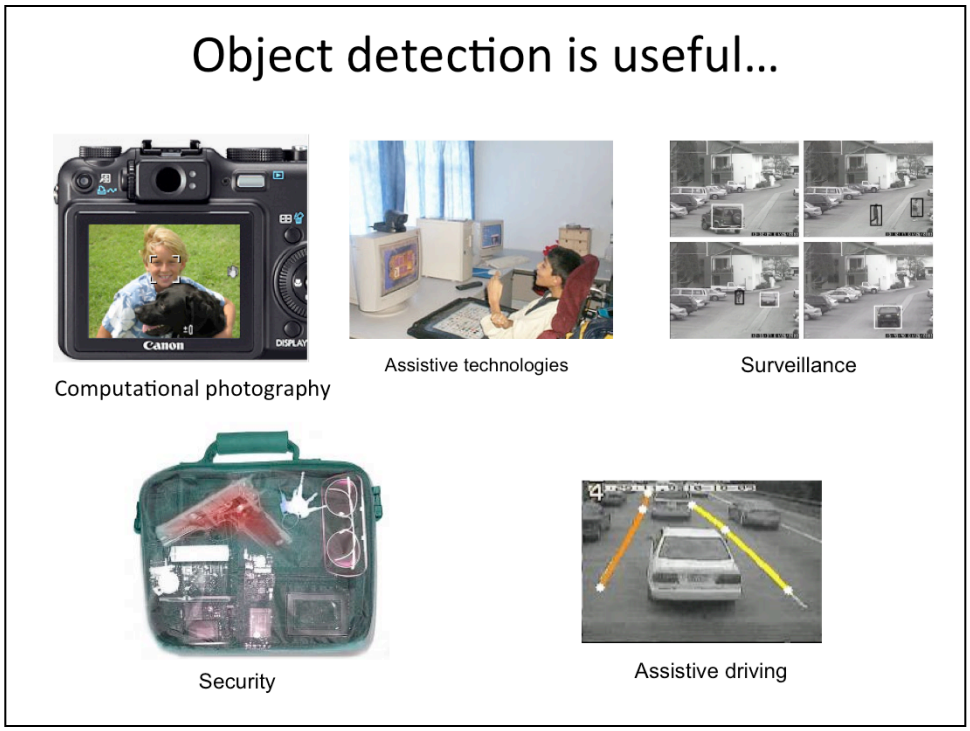


Here we see other examples of detected objects.

Detection:

Accurate localization (segmentation)

Localizing an object with a bounding box may not be accurate enough, and in many applications it is required to identify the actual object boundaries – this is called object segmentation. In the example in the slide, the clock of the building has be identified and segmented out.



Object detection is useful in many applications as indicated in the slide.

Additional notes by C. Kao:

Face detection can be used in photography, which helps you to focus on human face when you take a picture. Hand gesture detection can be applied to control computers. It is also widely used in surveillance, for automatically tracking criminals. This saves a huge amount of work that is traditionally done manually. In security, for example, it can be used in airport. With object detection, X-ray scanner is able to detect gun shape that may not be detected by human. Last year there was a final project that detected gun shape from X-ray images. For autonomous vehicles, it is very important to detect the lane boundary, so the vehicle can stay in a lane. These are all technologies that can benefit from object detection.

Categorization vs Single instance recognition

Which building is this? *Marshall Field* building in Chicago

Another important task in visual recognition is single instance object recognition. The goal here is to detect a specific object instance (say the *Marshall Field* building) as opposed to detect a generic object category such as a “building”.

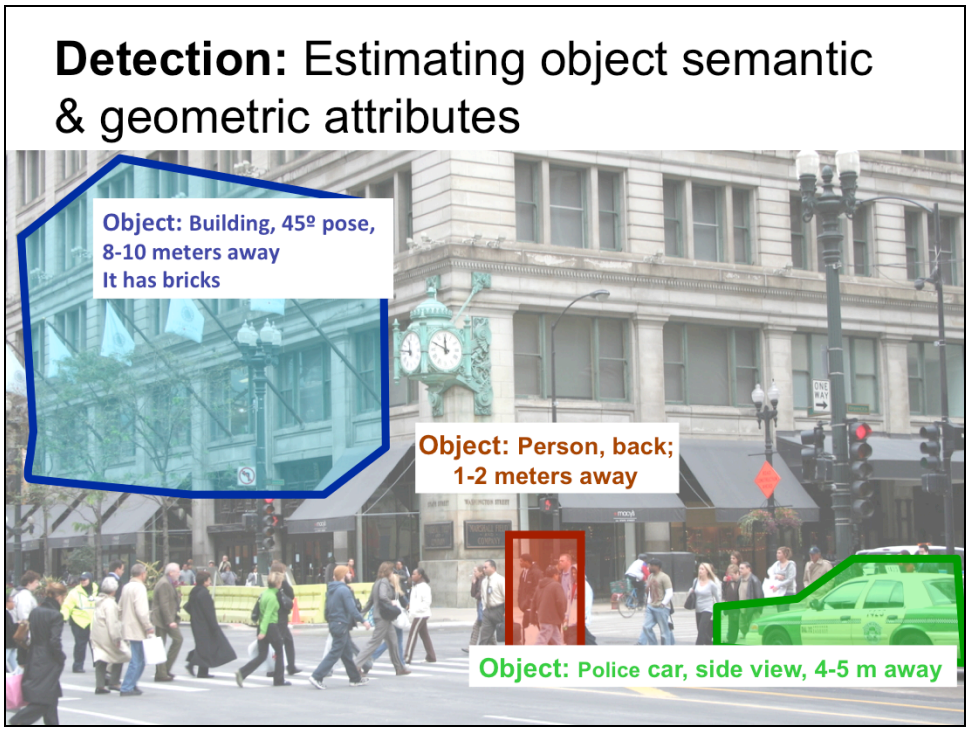
Categorization vs Single instance recognition

Where is the crunchy nut?

Another example of single instance object recognition. We want to detect in the right image the cereal box we see on the left (not a generic cereal box, but exactly that type and brand we see on the left).

Recognizing landmarks in mobile platforms

Single instance object recognition is useful in applications where one needs to identify landmark buildings from mobile devices.



Going beyond detection, other useful recognition tasks include the ability to estimate:

- Semantic attributes such as “this building is made of bricks”; ” this object is made of metal”.
- Geometrical attributes such as “this car is observed from the side” (i.e. 3D object pose) or is 4-5 meter away from the camera.

The later task is critical, for instance, in applications related to autonomous driving.

Activity or Event recognition

What are these people doing?

Other recognition tasks include human activity understanding or event recognition such as: a person is walking, a number of people are crossing the street.

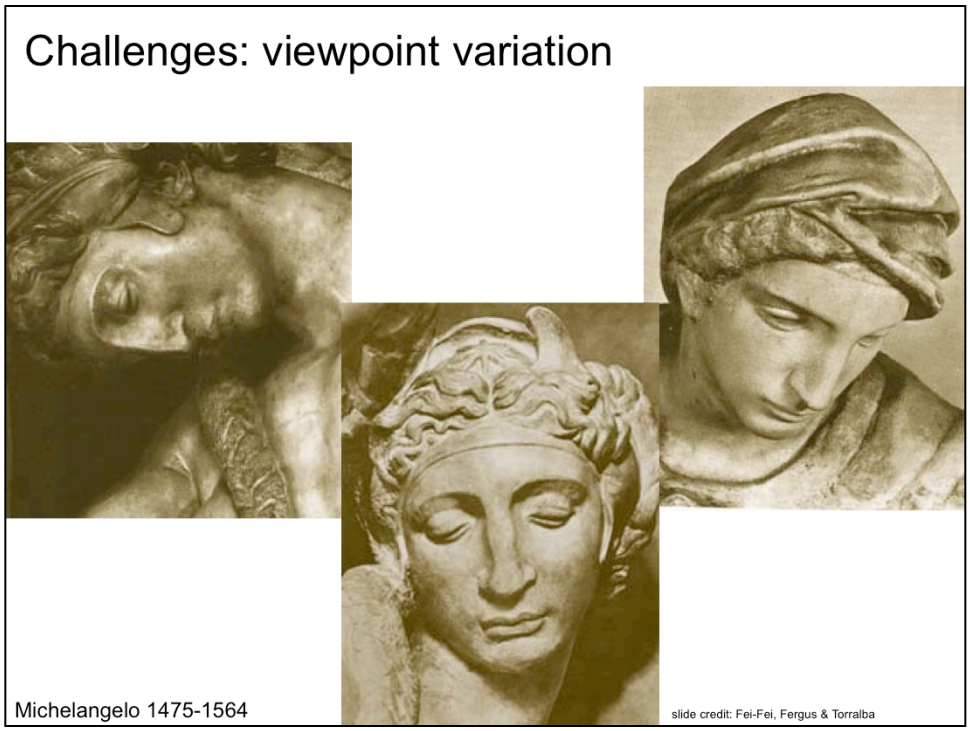
Visual Recognition

- Design algorithms that are capable to
  - Classify images or videos
  - Detect and localize objects
  - Estimate semantic and geometrical attributes
  - Classify human activities and events

Why is this challenging?

Therefore, in visual recognition, we study algorithms that can classify images and videos, detect object instances or categories as well as estimate their semantic and geometric attributes. Also, we study algorithms for classifying human events and activities. Solving all these tasks remain a great challenge in computer vision research. Why is that?

Recognition algorithms must be scalable to a very large number of categories: 10,000-30,000, if we only consider super-categories (e.g., a “car”), as demonstrated by I. Biederman in the early eighties. In fact, this number can be much larger if we include fine-grain categories (e.g., “sedan”, “SUV”, etc…).



For a given category, recognition algorithms must be able to characterize:

1. view point changes – the object appearance can be dramatically different as the object is observed from different view point.

Challenges: illumination

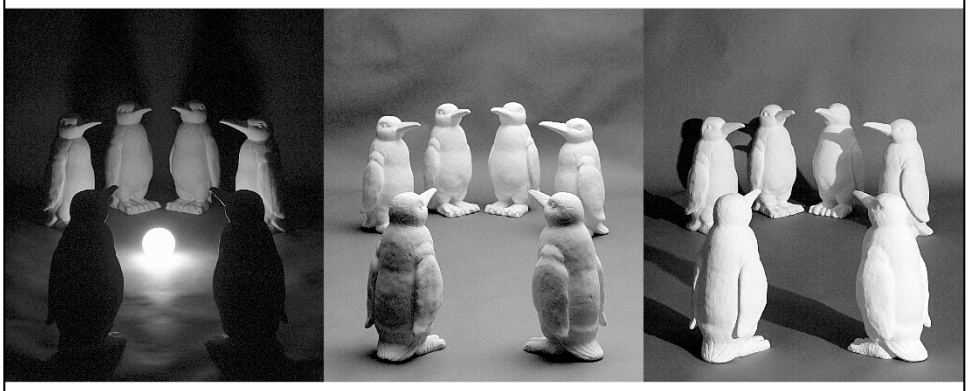
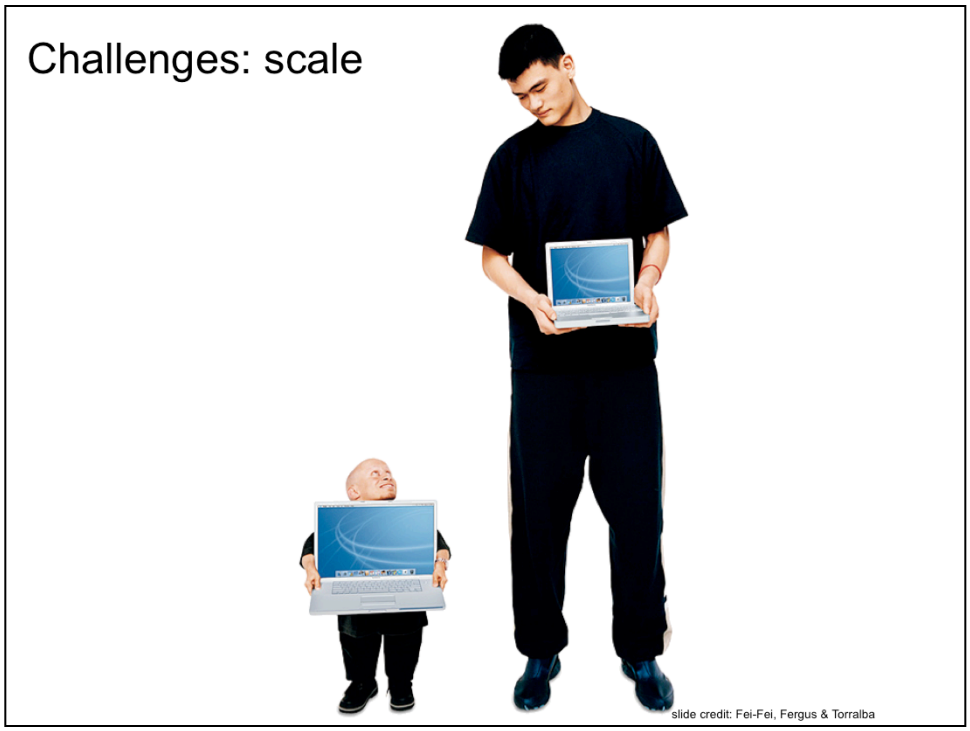


image credit: J. Koenderink

2. Illumination changes – the object appearance can be dramatically different as the object is observed under different illumination conditions.

Additional notes by C. Kao:

Another challenge is illumination variation. These are photos of the same scene, but with various lighting condition. The two penguins in the front become very different, with color changed from black to white. Human can easily determine these are the same object by shape, but it is still difficult for computers.



3. Scale variations – the object appearance can be dramatically different if an object in a given category changes its scale.

Additional notes by C. Kao:

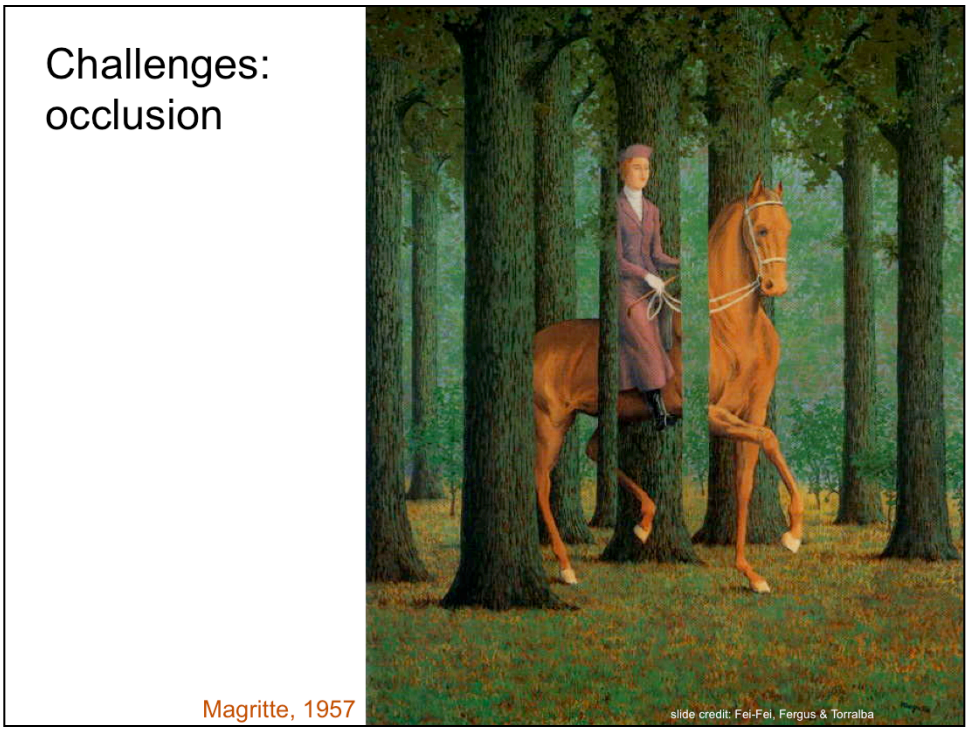
Scale variation is another challenge. Are these laptop the same size? Sometimes an object looks smaller because it is placed further from the camera, but sometimes it is because it is actually a smaller object.

Challenges: deformation

4. Deformations – the object appearance can be dramatically different if an object in a given category changes its shape because of deformations.

Additional notes by C. Kao:

Even an object is deformed, say the jeep is crashed, human is able to recognize the remaining part and recognize the jeep.



5. Occlusions – the object appearance can be dramatically different if it is occluded or truncated by another object.

Additional notes by C. Kao:

What if an object is partially occluded? Then we need to infer the geometric structure of the scene and figure out the ordering of objects. It becomes really hard to connect all parts together to be a complete object.

Challenges: background clutter

Kilmeny Niland. 1995

6. Background clutter – the object appearance can be similar to the image luminance properties of the background; in this case it's hard to segment the object from the background.

Additional notes by C. Kao:

When the background is cluttered, especially when they contain the same texture, detection and segmentation becomes error-prone. There are simply too many instances of the same dotted feature.

5. Intra-class variability – different object instances within the same category can show completely different appearance and shape properties. In this example, we see different instances of chairs; they look very different, yet they are all chairs!

Basic properties

- Representation
  - How to represent an object category
- Learning
  - How to learn the classifier, given training data
- Recognition
  - How the classifier is to be used on novel data

Building a recognition system generally requires three steps:

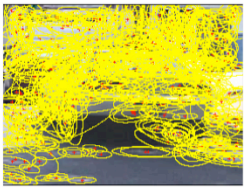
Representation – How do we represent an object; what model I can use to represent an object?

Learning – How do we learn the parameters of the object model from observations and labels?

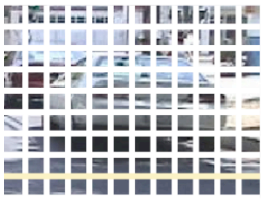
Recognition – how we do use the learnt model to recognize objects from images that have never been seen before in training?

Representation

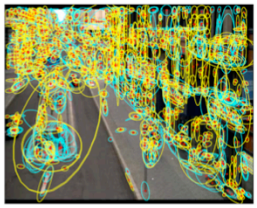
- Building blocks: Sampling strategies



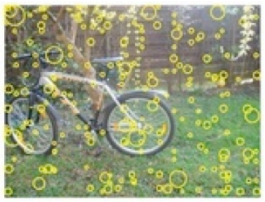
Interest operators



Dense, uniformly



Multiple interest operators



Randomly

Image credits: F.-F. Li, E. Nowak, J. Sivic

Representing an object or designing a model to characterize an object from images requires:

1. Selecting the features and the sampling strategies that can be used to describe an image. For instance, we can use interest operators such as DOG, HARRIS, etc.., or just use a dense, uniform sampling grid. Overlaying interest operators multiple times on the same image is another choice. Random sampling is also possible.

Representation

- Building blocks: Choice of descriptors
  [SIFT, HOG, codewords….]

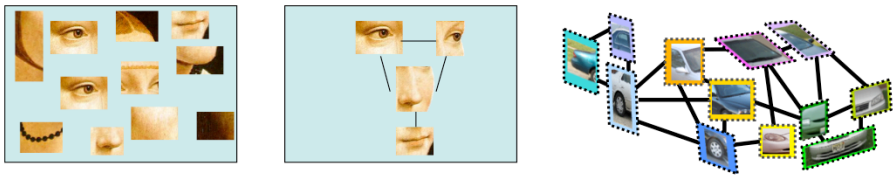
2. Selecting a descriptor associated to those features – e.g.. SIFT, HOG or codewords (as we shall see in the second part of this lecture).

Additional notes by C. Kao:

Various feature descriptors exist, most notably SIFT, HOG, and codewords. A good descriptor should have the ability to handle intensity, rotation, scale, and affine variations. Scale-Invariant Feature Transform (SIFT) is one of the most famous descriptors, which converts each image patch to a 128-dimension vector. Histogram of Oriented Gradients (HOG) counts occurrences of gradient orientation in localized portions, based on the distribution of intensity gradients or edge directions. A codeword is converted from vector-represented patches, defined as the center of the clustered patches.

Representation

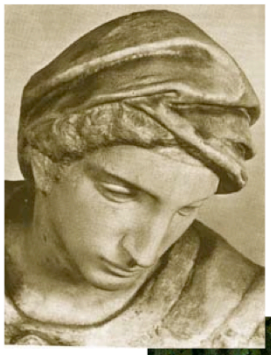
– Appearance only
– 2D location and appearance
– 3D location and appearance



3. Deciding how to combine and process those features. A simple option is to retain the appearance information captured by their descriptors and ignore their location in the image altogether (left panel). An other option is to encore their 2D locations as well (central panel). Finally, it is possible to combine features and their descriptors so as to model portions of the object in 3D (right panel).

Representation

- Invariances
  - View point
  - Illumination
  - Occlusion
  - Scale
  - Deformation
  - Clutter
  - etc.



When we decide how to represent an object we also need to think of which strategy we can use to make the model invariant with respect to:

View point
Illumination
Occlusion
Scale
Deformation
Clutter
…

Representation

- How to handle intra-class variability?
  - → It is convenient to describe object categories using probabilistic models
  - → Generative – vs – discriminative

…an in particular, how to handle intra-class variability.

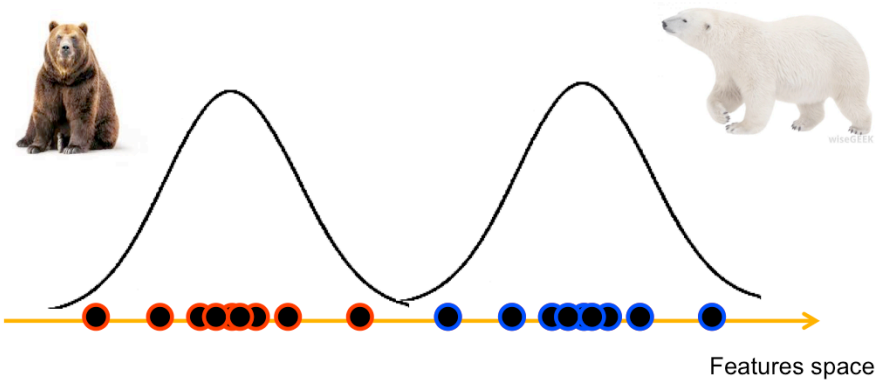
In order to account for the fact that objects shape and luminance properties change because of intra-class variations (as well as because of the other variations we described earlier), it is often the case that object models are described using probabilistic functions. Probabilistic models are convenient in classification. Hand-crafted rules are unwieldy because a large number of rules and exceptions are needed. A better way to generalize is by learning probabilistic models directly from training data.

We can divide probabilistic models or, in general, models for representing visual categories into two types:

Generative and discriminative

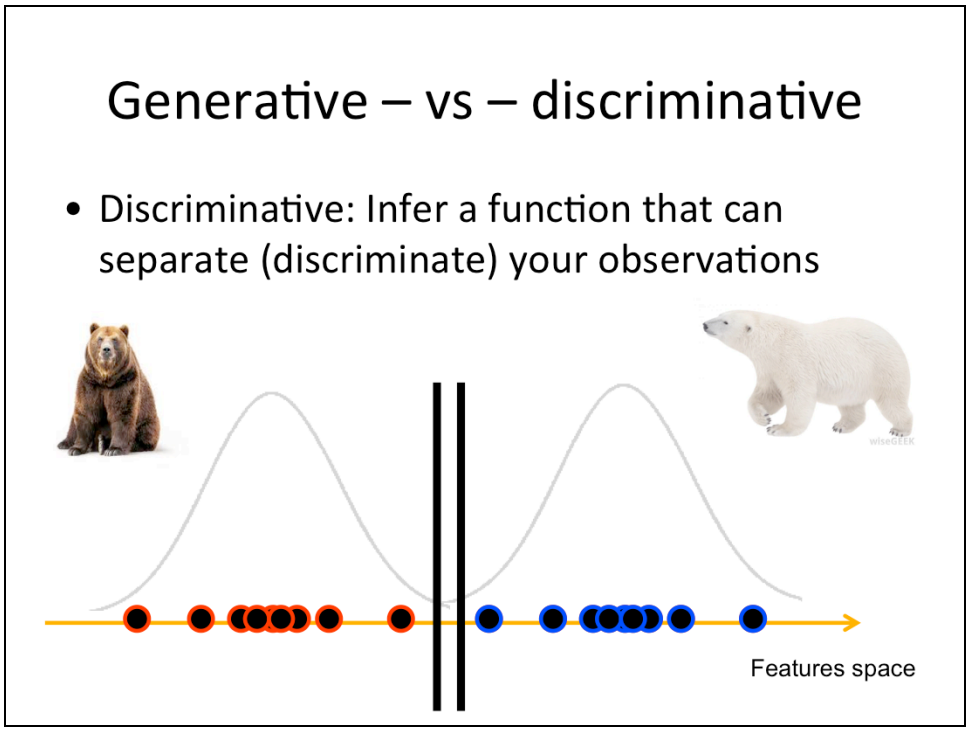
Generative – vs – discriminative

- Generative: Infer a function that can generate (explain) your observations



In generative models the goal is to design a function that is capable of *generating* (explaining) the observations.

In this example the yellow axis represents the feature space – the space where features are distributed. This axis could, for instance, capture the intensity values of pixels in images of polar and brown bears. Polar bears (mostly white) could be associated to the features points in blue whereas brown bears (mostly dark) could be associated to the features points in red. How to describe these observations (features)? Generative models seek to discover a function (like the Gaussians superimposed in black) that best explain the features points. In this examples, the left Gaussian best explains the red points and can be used to "model" the brown bear. Whereas the right Gaussian best explains the blue points and can be used to "model" the polar bear.



In discriminative models the goal is to design (estimate) a function that is capable of *separating* (discriminating) the observations. Thus, given the set of observations (red and blue feature points) the goal is find the best function that separates the two sets as opposed to estimate the functions that could have generated the two sets (associated to the brown and polar bears).

Generative models

- Naïve Bayes classifier
  - Csurka Bray, Dance & Fan, 2004
- Hierarchical Bayesian topic models (e.g. pLSA and LDA)
  - Object categorization: Sivic et al. 2005, Sudderth et al. 2005
  - Natural scene categorization: Fei-Fei et al. 2005
- 2D Part based models
  - Constellation models: Weber et al 2000; Fergus et al 200
  - Star models: ISM (Leibe et al 05)
- 3D part based models:
  - multi-aspects: Sun, et al, 2009

Here we see a few examples of classification methods that build upon generative models.

Additional notes by C. Kao:

Generative models have seen success in Natural Language Processing (NLP), medical diagnosis, and bioinformatics. Naive Bayes use the Bayes rule and, by conditioning on the joint pdf, allow to form a classifier. Hierarchical Bayesian topic models find underlying latent topics in documents, and has been used in scene categorization. Other part-based models seek to detect a small number of features and their relative positions to determine whether the whole object is present in the image in a generative fashion.

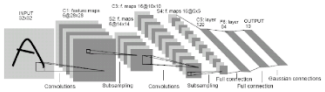
Discriminative models

Nearest neighbor

10^6 examples

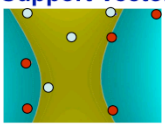
Shakhnarovich, Viola, Darrell 2003
Berg, Berg, Malik 2005...

Neural networks



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

Support Vector Machines



Guyon, Vapnik, Heisele, Serre, Poggio…

Latent SVM
Structural SVM

Felzenszwalb 00
Ramanan 03…

Boosting

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

Courtesy of Vittorio Ferrari
Slide credit: Kristen Grauman

Slide adapted from Antonio Torralba

Here we see a few examples of classification methods that build upon discriminative models.

Additional notes by C. Kao:

There are multiple discriminative models. Nearest neighbor classifies objects based on nearest training samples. Neural networks comes from the inspiration of neural structure of the brain, and has been really successful recently. Support Vector Machine (SVM) is one of the most common classifiers that deals with non-linear decision boundary. Latent SVM is introduced to utilize a star-structured part-based model with sliding window detection across the image. Boosting is a way to learn weak classifiers that as a whole forms a strong classifier.

Basic properties

- Representation
  - How to represent an object category; which classification scheme?
- Learning
  - How to learn the classifier, given training data
- Recognition
  - How the classifier is to be used on novel data

After we propose a strategy for modeling an object class, how can we learn such a model from the training data and define a suitable classification task?

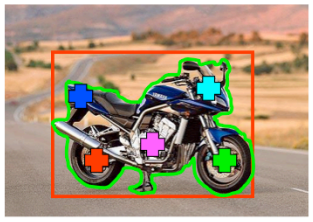
Learning

- Learning parameters
- Generative functions or separating functions?

One of the key questions is how to learn the model parameters. In generative models, we want to maximize the joint likelihood. While in discriminative models, we want to estimate the posterior probability from the training set and validation set. The validation set is part of the data we take out from the training set to assess the quality of the trained model.

Learning

- Learning parameters
- Generative functions or separating functions?
- Level of supervision
  - Noisy labels; image labels; bounding box; manual segmentation; part annotations
- Batch/incremental
- Priors



Another aspect of learning is the level of supervision. Supervised learning "learns" the model from labeled training data, which is accurate but not scalable and tend to be more costly in terms of labors involved. Unsupervised learning, on the other hand, tries to find the hidden structure in unlabeled data. Adding supervision to the training process may include adding knowledge as for where the object may be is in the image. This can be done by specifying the object bounding box, a segmentation mask or by even identifying specific object parts (wheels, lights, etc…)

Batch learning refers to learning from a batch of data. Due to the large amount of data, it would be inefficient and slow if the model is retrained every time a new image is added to the data set. This is when incremental learning becomes useful, which gradually estimates the best model as the dataset grows incrementally.

Prior knowledge may be useful in learning, especially when the dataset is dedicated for a certain category of images.

Learning

- Learning parameters
- Generative functions or separating functions?
- Level of supervision
  - Noisy labels; image labels; bounding box; manual segmentation; part annotations
- Batch/incremental
- Priors
- Training images:
  - Issue of overfitting
  - Negative images for discriminative methods

There are some challenges in learning, and a common one is how to avoid over-fitting. It may be better to have a simpler model (which doesn't behave perfectly well on the training set), rather than a complex model that does very well on the training set but it is hard to generalize.

A final issue is related to the choice of negative images. These depend on the classification task we want to perform: classifying a face-vs-non face requires a certain set of negative images; classifying a giraffe-face-vs-non-giraffe-face requires a different set of negative images (see slide).

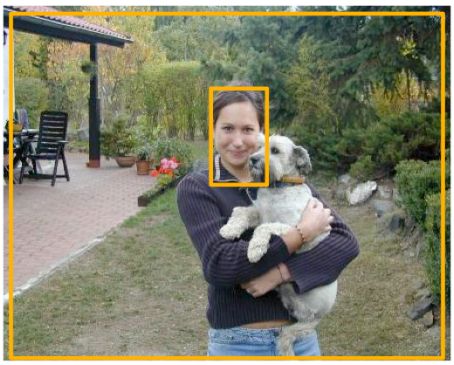
Basic properties

- Representation
  - How to represent an object category; which classification scheme?
- Learning
  - How to learn the classifier, given training data
- Recognition
  - How the classifier is to be used on novel data

Now that we have learnt the parameters that describe our object models and train a relevant classifier. How can this be used to perform a recognition task on novel unseen test images?

Recognition

– Recognition task: classification, detection, etc..



First, we need to know our goal. Is it for classifying the whole image? Or is it for object detection (localization) in the image?

Recognition

– Recognition task

– Search strategy:

  – Sliding Windows

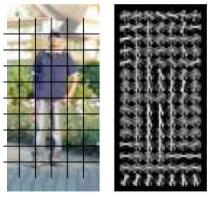
- Viola, Jones 2001
- Dalal and Bill Triggs, 2005

If the goal is to implement an object detector (say a face detector), we need to choose a search strategy to explore the test image. The simplest and most intuitive one is the sliding windows (SW) approach. SW explores the image at various locations and for each of these locations (x,y) compute the probability that an object model of class k (say the class "face") is found within an image window of scale **S** around (x,y)**.** The set of locations that are associated to a probability that is above a certain threshold return the locations of the detected object.

Dalal-Triggs pedestrian detector

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05

Section 17.1 [FP]



Represent an object as a collection of HoG templates

1. Extract fixed-sized window at each position and scale
2. Compute HOG (histogram of gradient) features within each window
3. Score the window with a linear SVM classifier
4. Perform non-maxima suppression to remove overlapping detections with lower scores

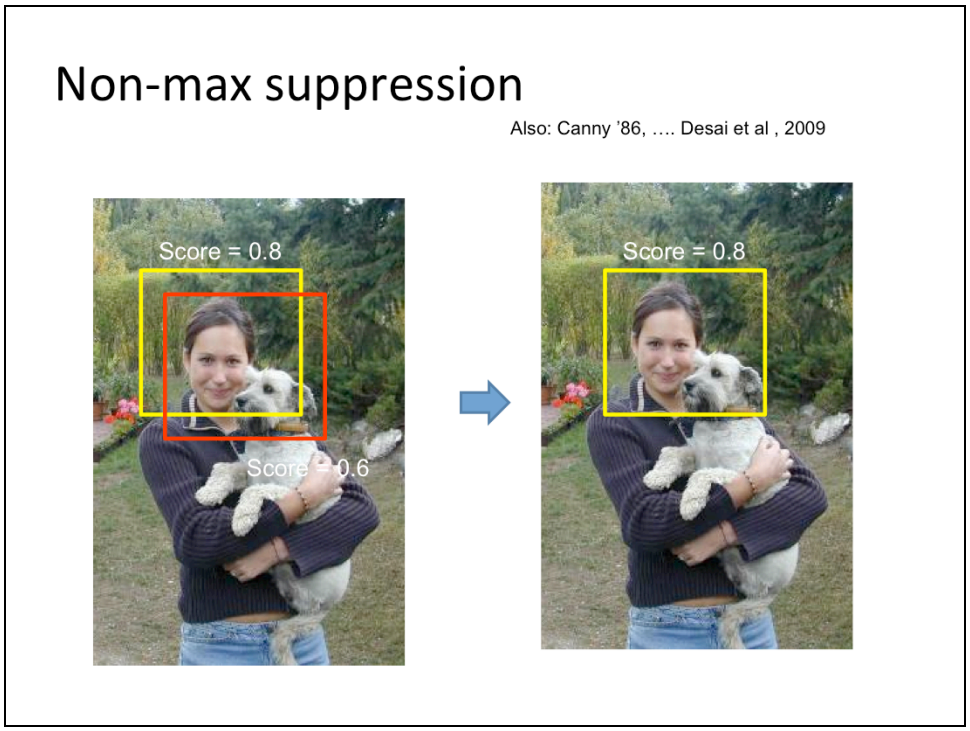
A great example of this approach is the pedestrian detector developed by Dalal and Triggs in 2005. With 6000+ citations this is another of the most influential papers in computer vision! In this approach Dalal and Triggs developed a feature descriptor called Histograms of Oriented Gradients, or HOG (see lecture 10) to match images to templates. An object detector can be easily designed following these 4 steps:

1. Extract fixed-sized window at each position and scale
2. Compute HOG (histogram of gradient) features within each window
3. Score the window with a linear SVM classifier
4. Perform non-maxima suppression to remove overlapping detections with lower scores

Non-max suppression

Section 17.1 [FP]

Non-maximum suppression is a very important step in most detection algorithms. The idea is to suppress bounding boxes that significantly overlap with bounding boxes with a higher detection score. As a result, each actual object in the scene has a unique detection, located in the place with the strongest detector response. See Section 17.1 [FP].



The slide shows an example of non-maximum suppression

Recognition

– Recognition task

– Search strategy:

  – Sliding Windows

Simple!

But computational expensive..

While it is very simple to implement, the major drawback of SW is its high complexity. In order to explore the image, one needs to consider all possible locations (x,y) (2 dimensions), scales S (1 dimension), rotation θ (1 dimension) of the window. Moreover, this number increases even more if one wants to find more than one class per image (say N classes).

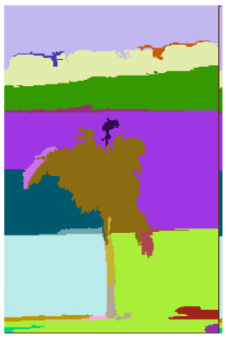
Recognition

– Recognition task

– Search strategy:

  – Sliding Windows

  – Bottom-up cues (segmentation)



Felzenszwalb and Huttenlocher, 2004

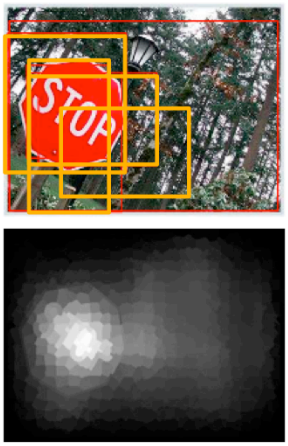
To mitigate this issue, other approaches leverage bottom-up cues (e.g., segmentations – regions in the image that share similar appear properties) to slide the window in regions that are more likely to contain the object.

Recognition

– Recognition task

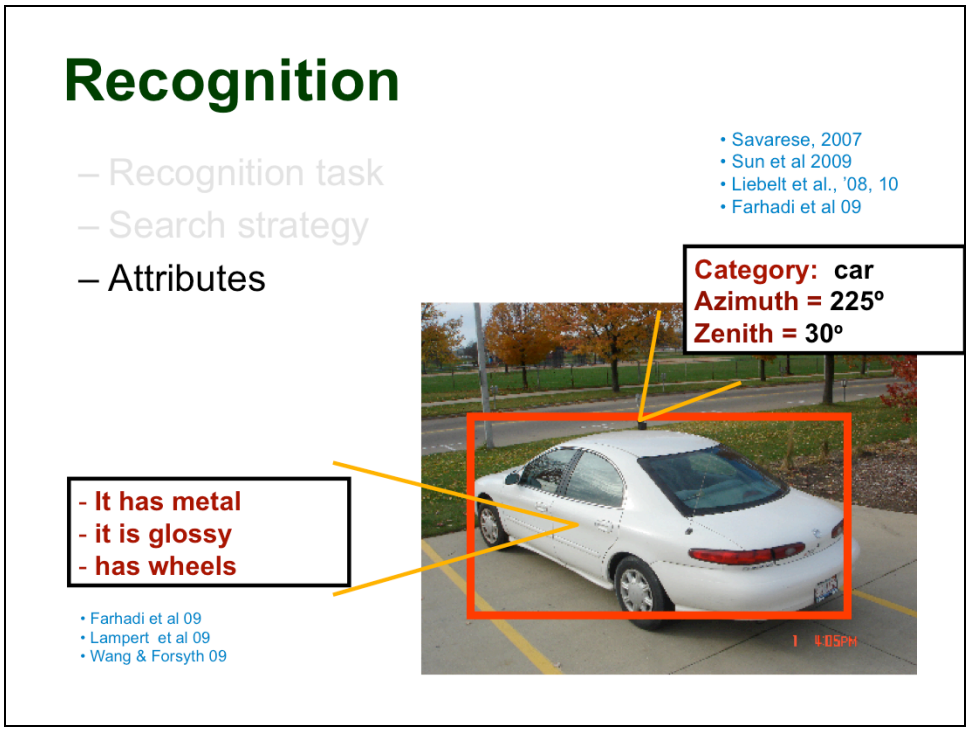
– Search strategy:

  – Sliding Windows
  – Bottom-up cues (segmentation)
  – Saliency



Jia & Han, 13
Alexe, et al 10
…

Finally, a popular recent strategy is based on the concept of region proposals. We'll discuss in a few more details the idea of region proposals during the lecture on CNNs. The intuition here is to use mechanisms based on saliency to indicate likely locations wherein the objects could be located.



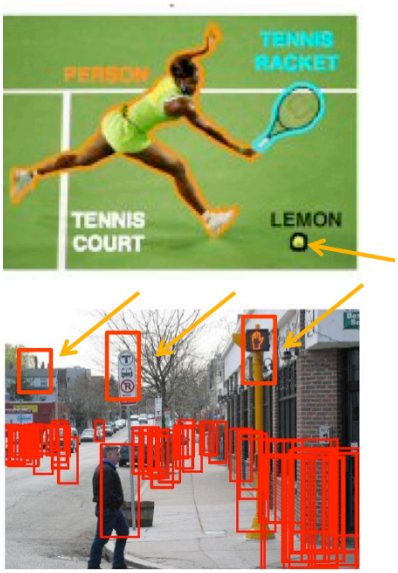
Other important aspects of a recognition task is the ability to infer semantic attributes (whether an object contains metal or not, whether it is glossy or has wheels) as well as geometrical attributes (e.g. estimate its 3D pose or shape).

Recognition

– Recognition task
– Search strategy
– Attributes
– Context

Semantic:
- Torralba et al 03
- Rabinovich et al 07
- Gupta & Davis 08
- Heitz & Koller 08
- L-J Li et al 08
- Bang & Fei-Fei 10

Geometric
- Hoiem, et al 06
- Gould et al 09
- Bao, Sun, Savarese 10



Finally, another key question is whether we want to use contextual cues to help solving a certain recognition task. Context can be semantic or geometric. The top figure shows an example of semantic context: it's very unlikely that the object highlighted by the orange arrow is a lemon– it's much more likely that it is a tennis ball given that we are recognizing a tennis player and a tennis court (contextual co-occurances).

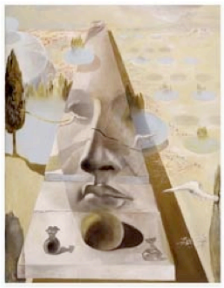
The bottom figure shows an example of geometric context: red bounding boxes are detected pedestrians; it's very unlikely that the bounding boxes highlighted by the orange arrows are actually a person– it's much more likely that these are false alarms given that these detections are above the street, floating in the air.

Agenda on recognition

- Image classification (lecture 11, 12, 15)
  - Bag of words representations
- Object detection (lecture 12, 14, 15)
  - 2D object detection
  - 3D object detection
- Scene understanding (lecture 13, 16)

These are the topics covered in the next few lectures on recognition.

Lecture 11
Visual recognition



- An introduction to recognition
- Image classification – the bag of words model

In the next slides we will explain a methodology for representing images or objects, called the bag of words model. and an approach for object classification which is built on top of it.

Bag of words models

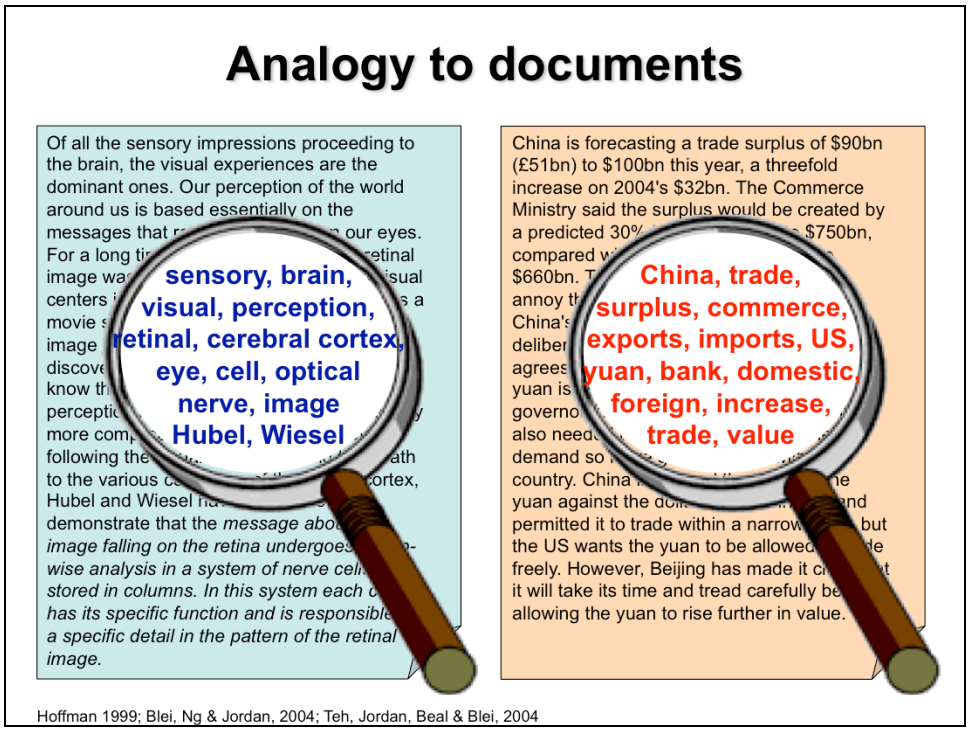
- Used for image and object classification
- Designed to handle variability due to:
  - View point
  - Illumination
  - Occlusions
  - Intra-class

Bag of words models have been used for solving image and object classification problems. The models can handle various types of appearance variability due, for instance, to view point, illuminations, occlusions and intra-class variations.

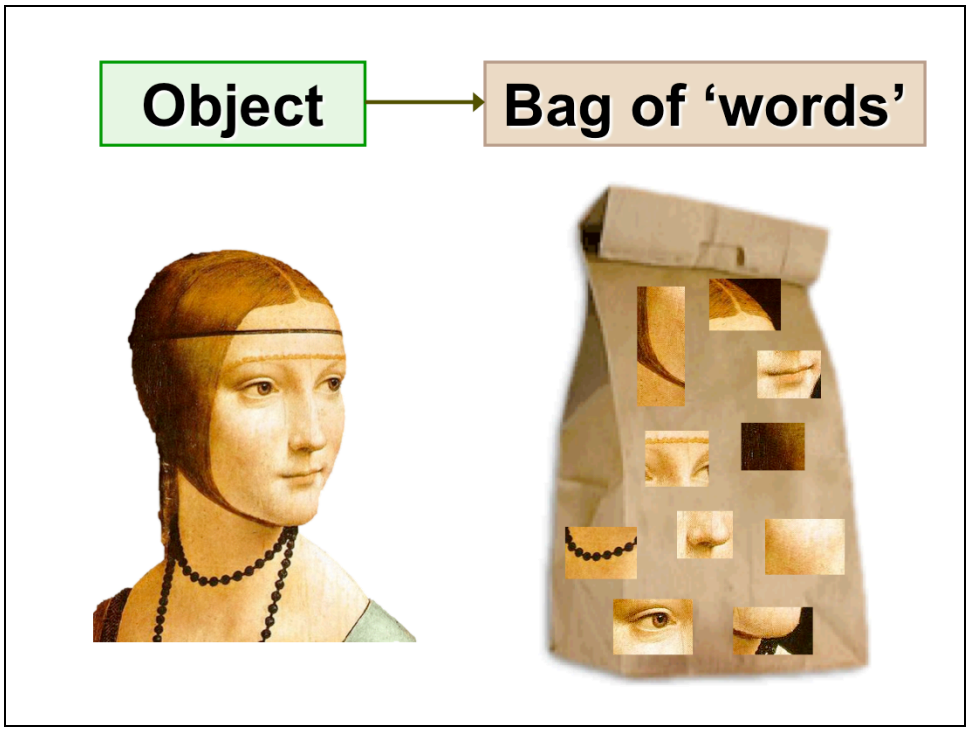
Inspired by works on document analysis!

- Early "bag of words" models: mostly texture recognition
  - Cula & Dana, 2001; Leung & Malik 2001; Mori, Belongie & Malik, 2001; Schmid 2001; Varma & Zisserman, 2002, 2003; Lazebnik, Schmid & Ponce, 2003;
- Hierarchical Bayesian models for documents (pLSA, LDA, etc.)
  - Hoffman 1999; Blei, Ng & Jordan, 2004; Teh, Jordan, Beal & Blei, 2004

Some of the early basic concepts related to Bag of Words (BoW) models (along with the relevant learning machinery) were proposed in the Natural Language Processing (NLP) community; Concurrently, computer vision scientists introduced BoW models for solving texture recognition problems. Later, they were also employed for object and scene categorization.



The idea of Bag of Words (BoW) was first borrowed from NLP. An early reference to "bag of words" is the Z. Harris's article on Distributional Structure (1954) in computational linguistics. The basic idea of BoW in NLP is the following: Instead of representing a document as a series of words grouped into sentences, the BoW model represents a document as a *distribution* of words that typically occur in the document (frequency of occurrences) whereby the spatial structure that connects all the words is lost.

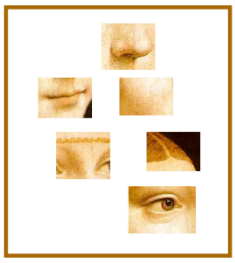


The idea of the Bag of Words (BOW) for visual recognition is inspired by the previous discussion and it's very is simple. Instead of representing an object as a collection of features along with their locations, we represent the object as an unordered distribution of features (the bag) that can capture characteristic epitomic properties of the object (the words). For instance, we can represent a face as a collection of epitomic features associated to eyes, nose, lip, etc.

definition of “BoW”

– Independent features

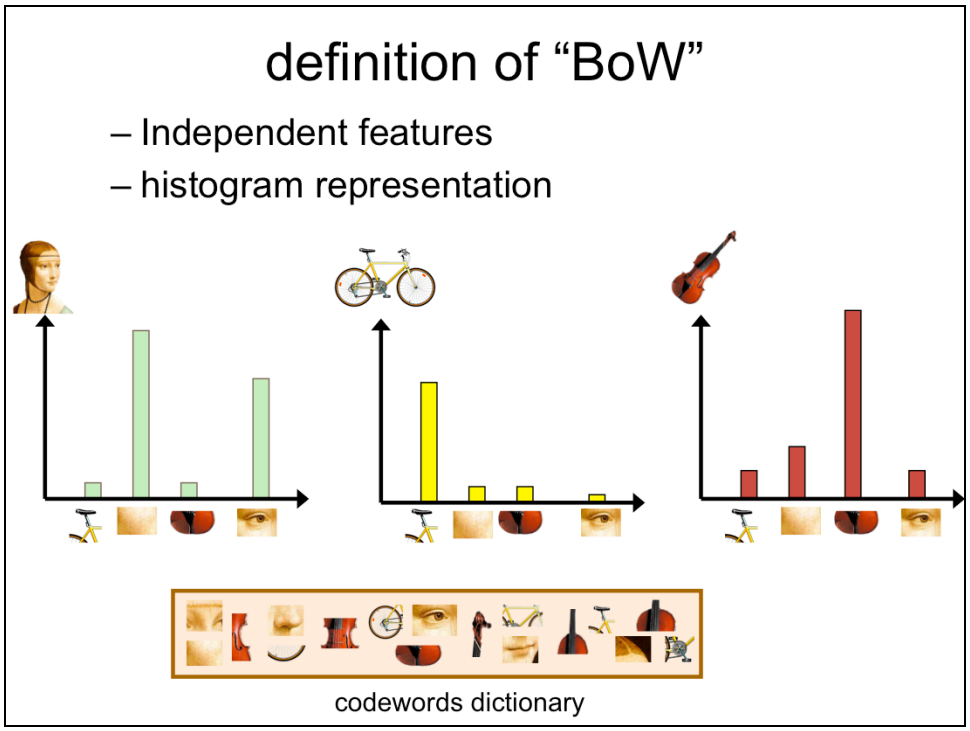
face



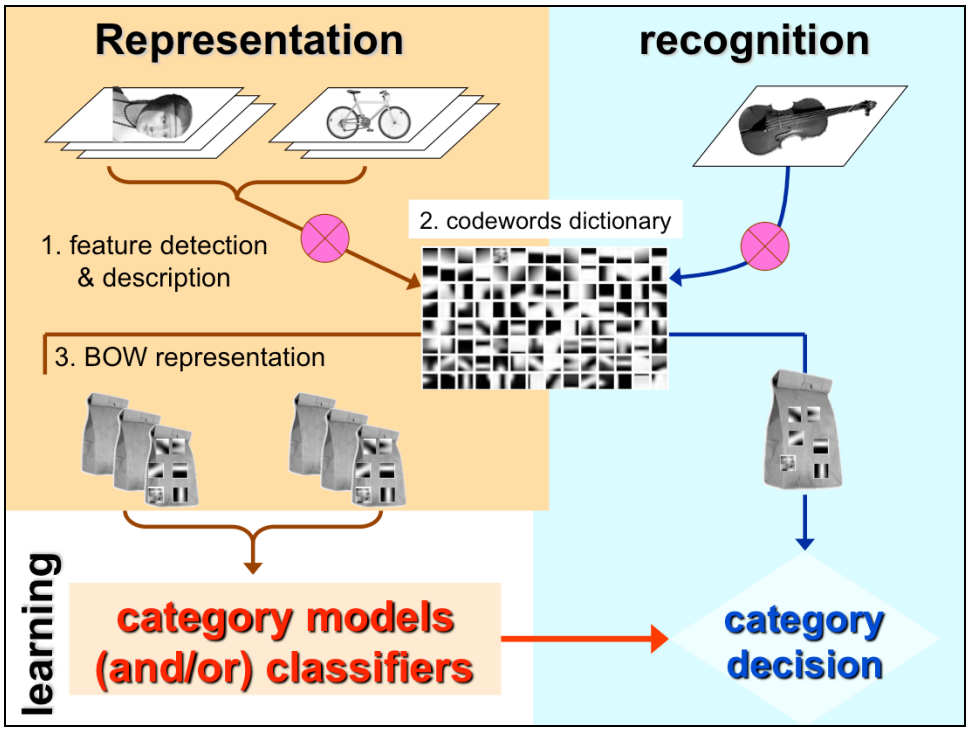
bike

violin

The slide shows 3 examples of object categories (face, bike and violin) which are represented by a bag of various visual words.



In more details, we first construct a dictionary of words (or code-words) from a training set. Than, we represent each category as a frequency distribution of codewords– that is, as a histogram that counts how many times each codeword appears in the image (object). Examples of such histograms are shown in the slides for 3 categories.



Let's now describe in details an object classification approach that uses the BoW model as a key ingredient. We start with an overview of such an approach. The slide highlights the 3 properties of a recognition system: representation, learning and recognition.

Suppose we want to learn a classifier that can classify images into one out of k different classes.

Representation:

1. We build a dictionary of code words using the features (and their descriptors) that are detected from a training set of images.

2. For each image of class j we associate a histogram of codewords using the dictionary.

3. Thus, each class is represented by a collection of histograms. This collection forms a category model

Learning:

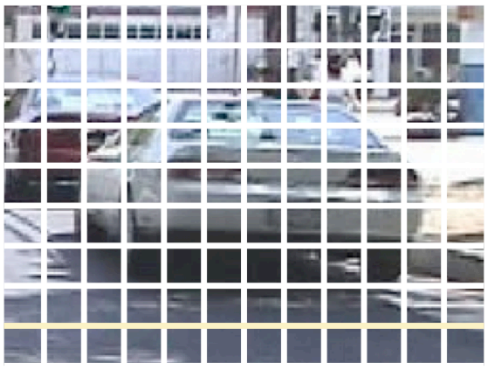
1. We learn a classifier that is capable of separating different classes of histograms (different category models)

1.Feature detection and description

The first step is feature detection and description

1.Feature detection and description

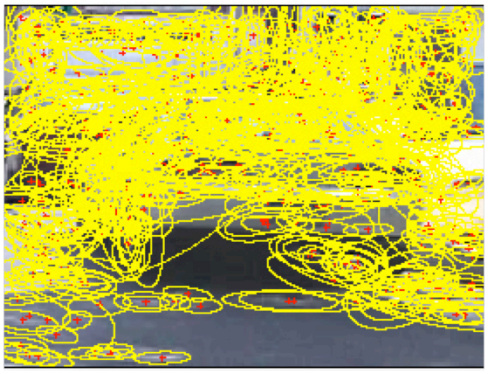
- Regular grid
  - Vogel & Schiele, 2003
  - Fei-Fei & Perona, 2005



There are several ways to extract features. One way is to sample features along a grid structure.

1.Feature detection and description

- Regular grid
  - Vogel & Schiele, 2003
  - Fei-Fei & Perona, 2005
- Interest point detector
  - Csurka, et al. 2004
  - Fei-Fei & Perona, 2005
  - Sivic, et al. 2005

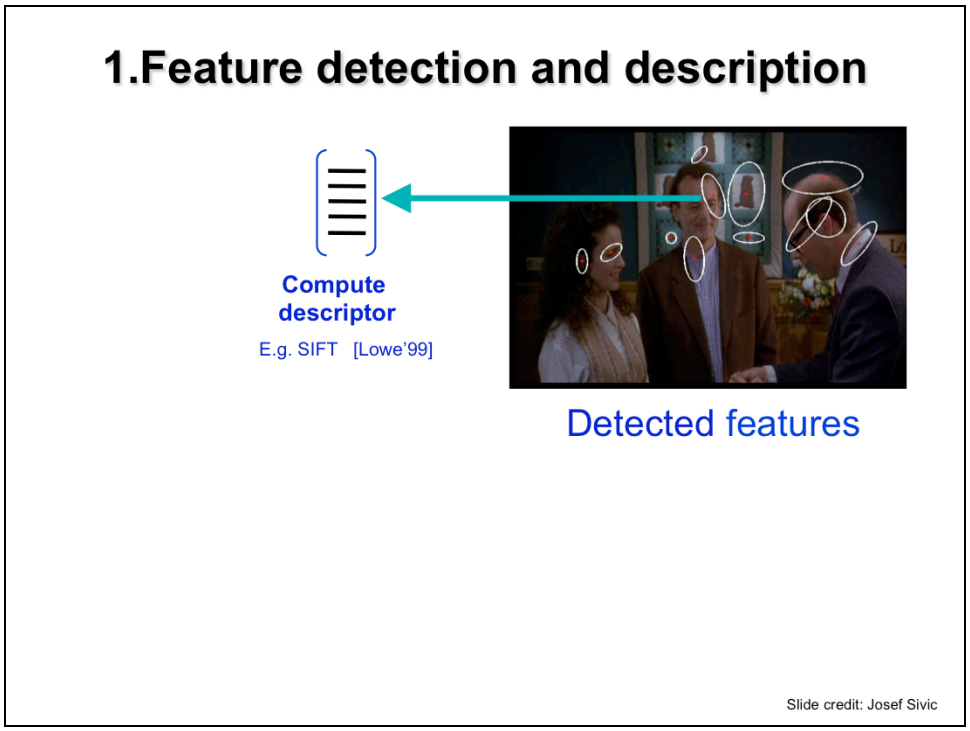


Another way is to extract features using interest point detectors (DOG, Harris, etc…)

1.Feature detection and description

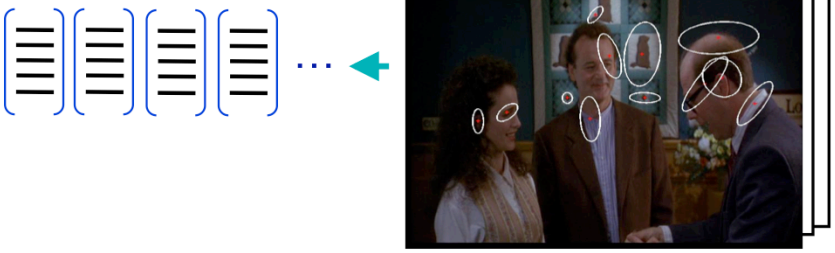
- Regular grid
  - Vogel & Schiele, 2003
  - Fei-Fei & Perona, 2005
- Interest point detector
  - Csurka, et al. 2004
  - Fei-Fei & Perona, 2005
  - Sivic, et al. 2005
- Other methods
  - Random sampling (Vidal-Naquet & Ullman, 2002)
  - Segmentation based patches (Barnard, Duygulu, Forsyth, de Freitas, Blei, Jordan, 2003)

Finally, another way is by randomly sampling the image or by extracting features using a segmentation scheme.

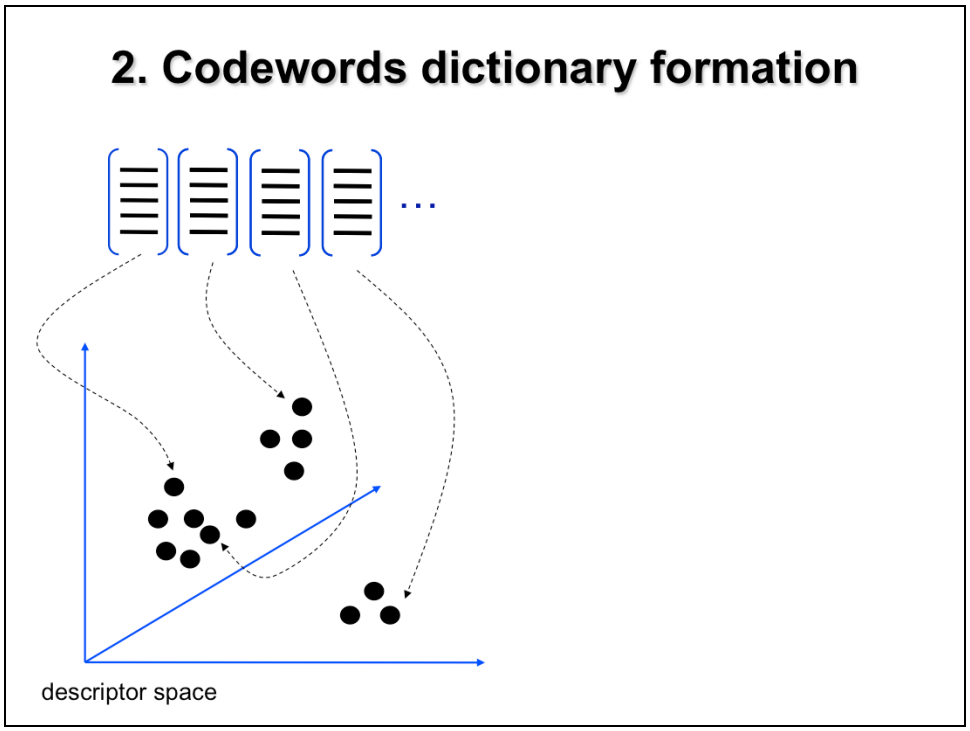


For each of the detected key points, we can associate a descriptor (e.g. SIFT, etc…). A descriptor is a Nx1 vector.

2. Codewords dictionary formation



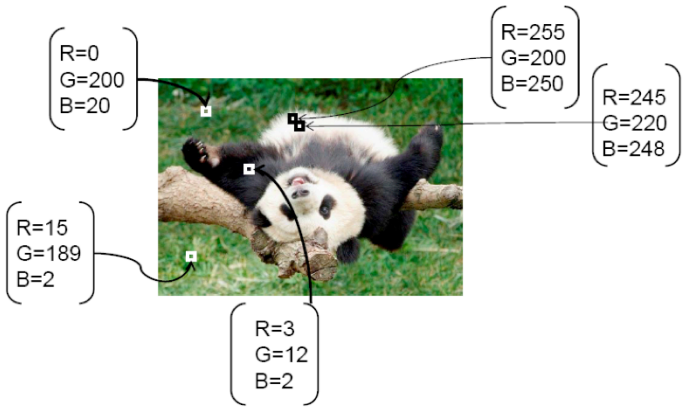
Thus, we obtain a collection of such descriptors for each image and or each class.



Such vectors are can be associated to a point in the descriptor space – each axis of this space is associated to a coordinate of the descriptor.

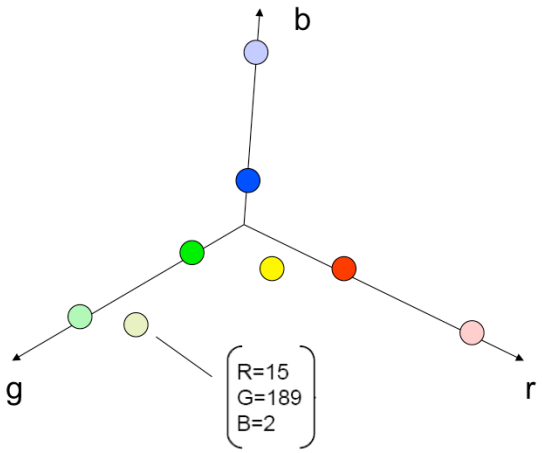
This can be a high dimension space (e.g. N) if the descriptor has a large number of dimensions.

Example: color feature

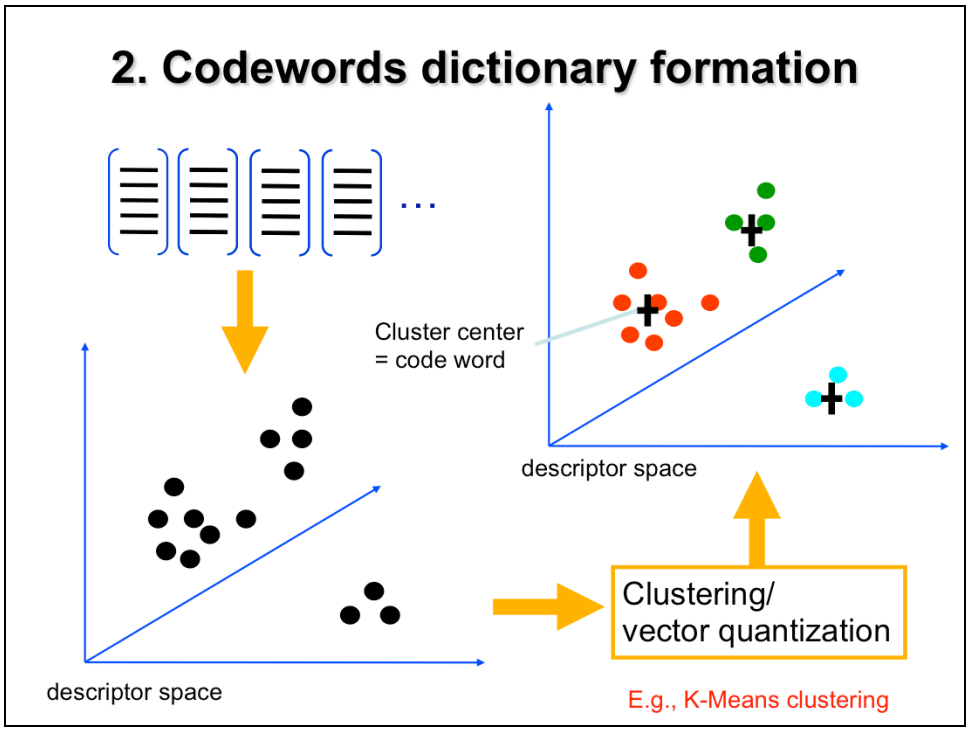


For instance, suppose the descriptor associated to each key point is a 3x1 vector that captures the 3 RGB values of the image at the detected key point location. Some examples are shown in the figure.

Example: color feature

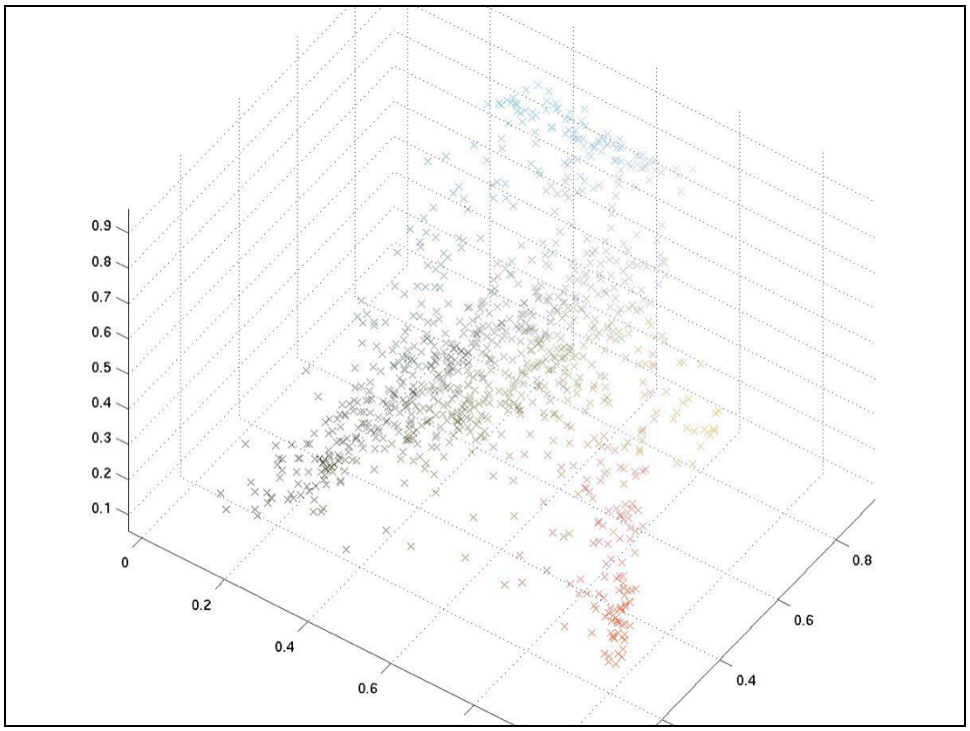


A descriptor in this space will be a point in the RGB (3D) space.

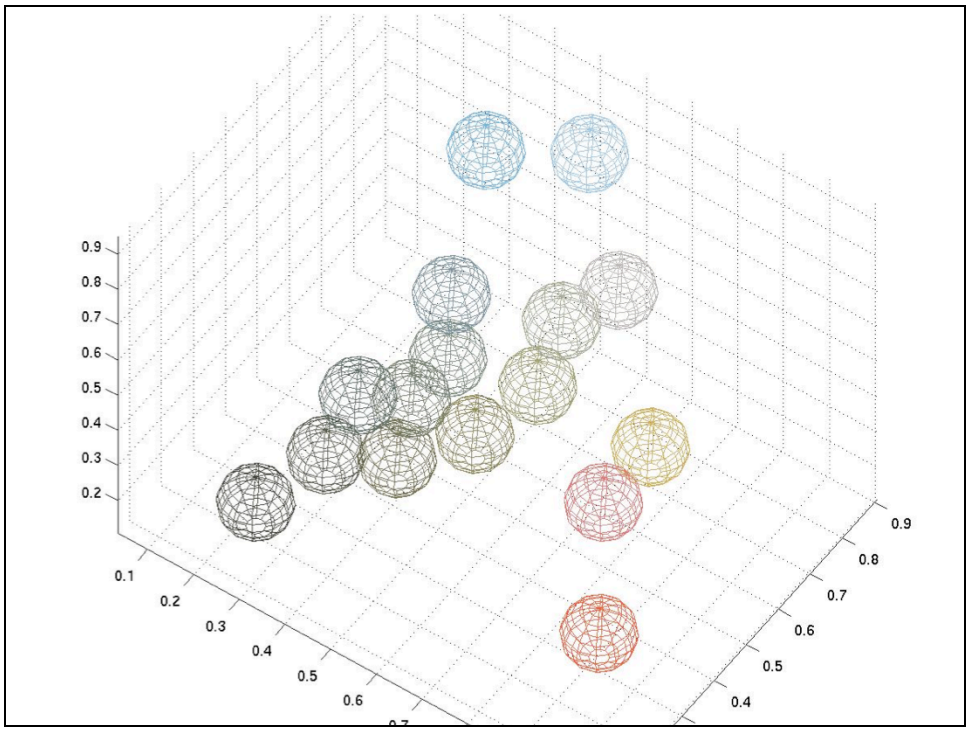


By repeating the same operation for all the training images we cumulate a large number of points in the descriptor space. Then, we preform a clustering procedure which allows to aggregate points that are close to each other in the descriptor space (and thus, that share similar descriptions). Clustering can be performed using techniques such as k-means. Examples of clusters are shown in the upper right figure (different clusters are shown in different color codes). The center of a cluster returns the codeword and is indicated by a cross “+”.

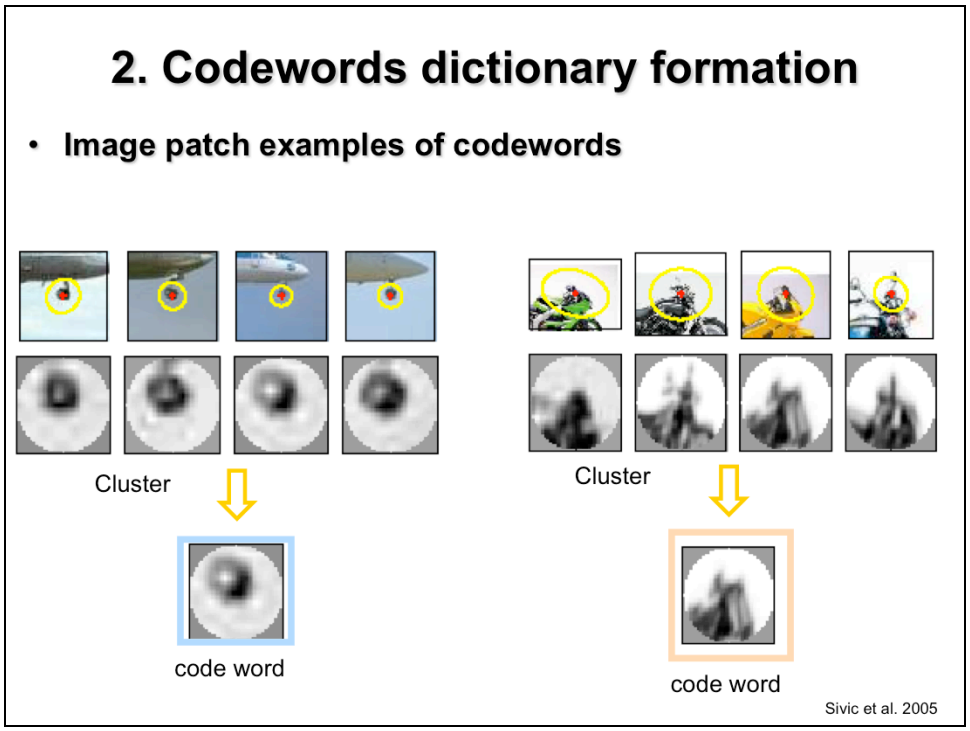
Here is an example when an RGB descriptor is used



These are points in the RGB space

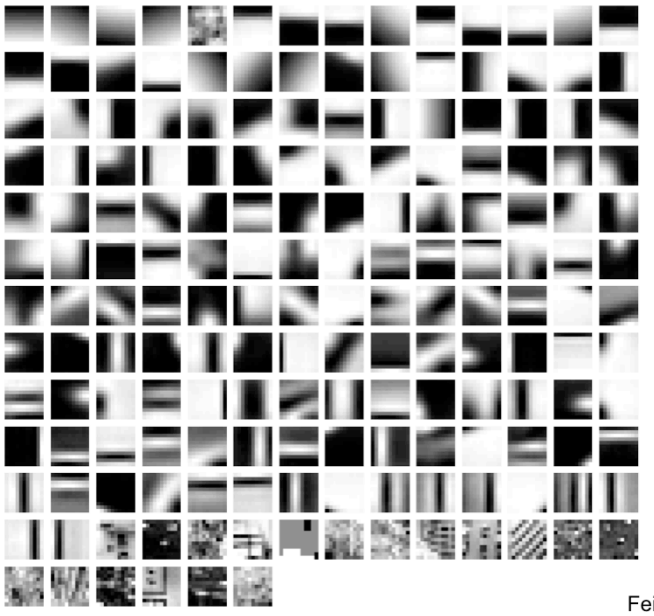


These are corresponding clusters



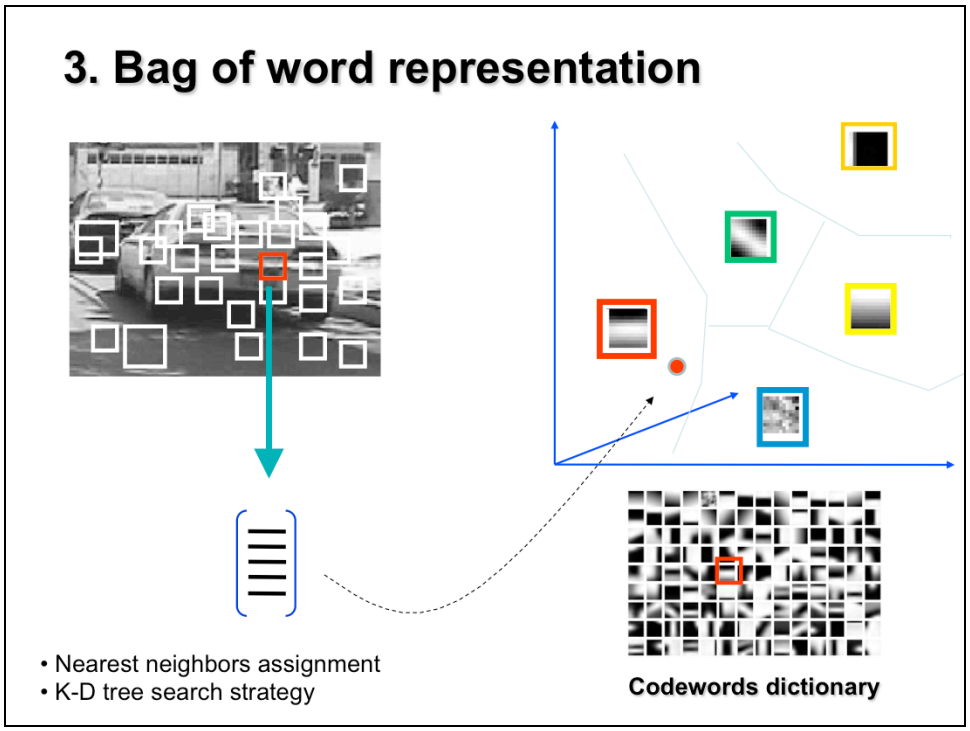
Here we see examples of clusters when a SIFT descriptor is used. For visualization purposes, we visualize the pixel intensities values in the SIFT window instead.

2. Codewords dictionary formation



Fei-Fei et al. 2005

Here is an example of learnt dictionary.



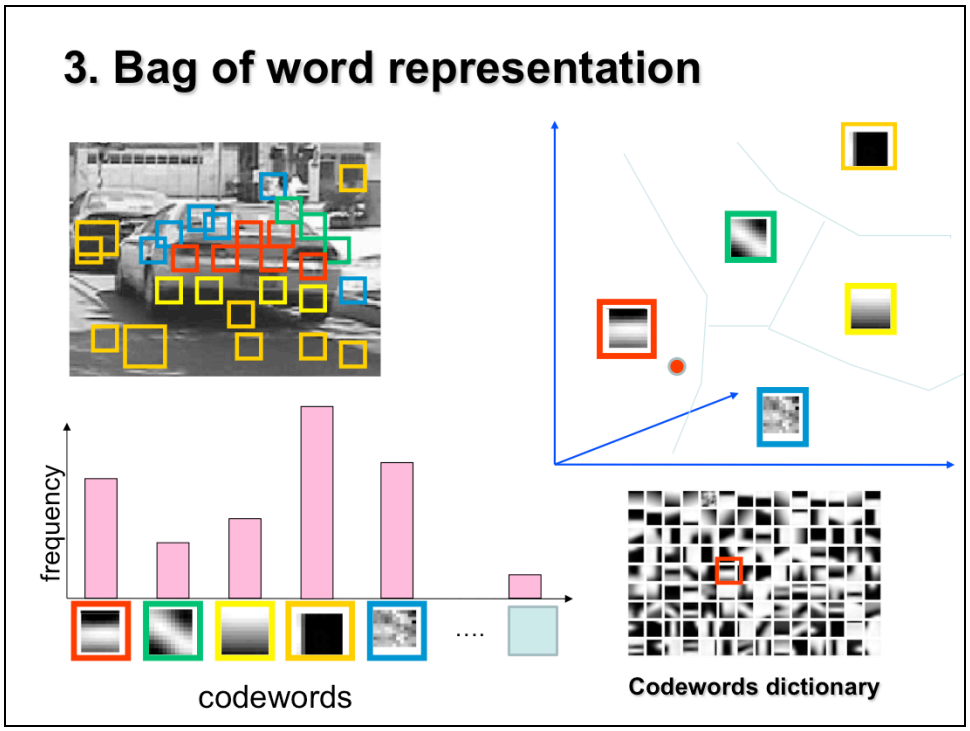
Once the dictionary is built, we can use it to represent an image or an object as an histogram of codewords. This is done with the follow steps:

1. We extract feature points in the image we want to represent (e.g., white squares in the figure)

2. We associate a descriptor vector to each of these features

3. We associate to each descriptor a corresponding codeword (red codeword) in the dictionary by analyzing which cluster the descriptor (red point) belongs to (red cluster).

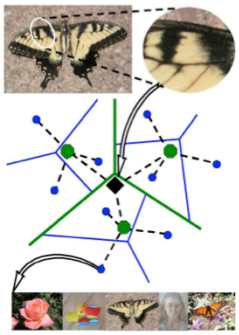
Note that assigning a feature to a word in the dictionary requires finding the nearest neighbor which we can be speeded up using K-D tree. However, for high dimensional features, the speed up is generally worse than O(log n) and only sub-linear (where n is the number of words in dictionary).



4. We build the histogram by counting how many times each codeword appears in the image

Visual vocabularies: Issues

- How to choose vocabulary size?
  - Too small: visual words not representative of the object appearance distribution
  - Too large: quantization artifacts, sparse histograms, overfitting
- Computational efficiency
  - Vocabulary trees (Nister & Stewenius, 2006)



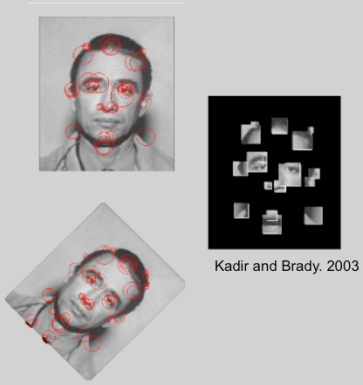
Some of the problems related to a BoW representation include how to choose the vocabulary size.

- If the size is too small, the visual words won't be representative of the appearance distribution within and across classes.
- If the size is too large, it may generate quantization artifacts and overfitting

Also, associating a codeword to a descriptor can be computationally expensive; organizing the clusters in a tree structure (also called dictionary tree) can ameliorate the issue.

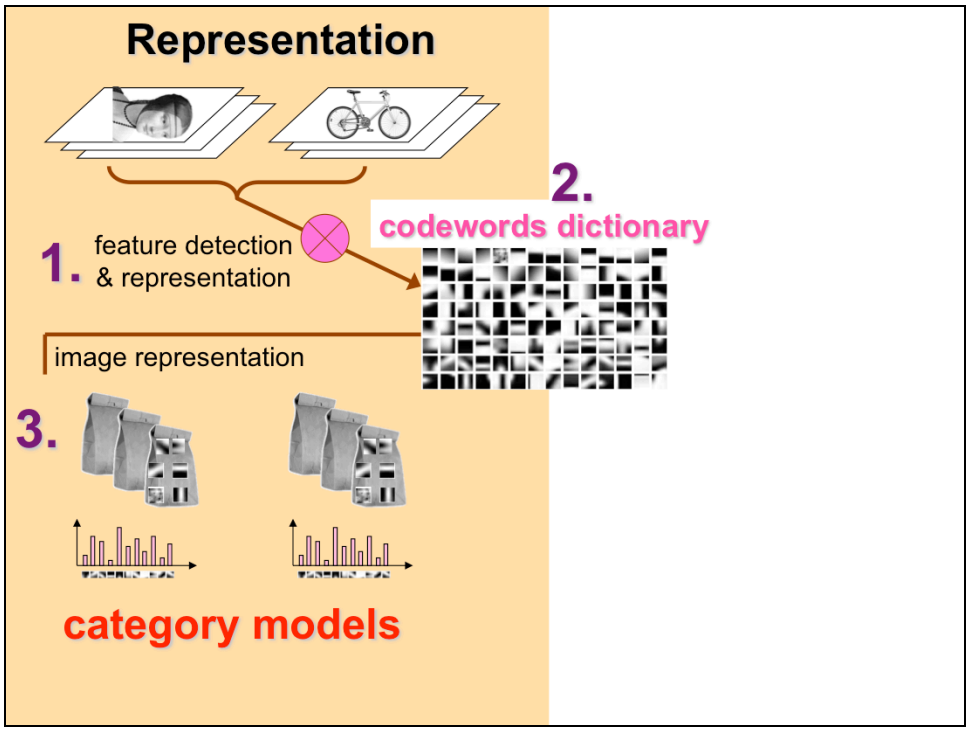
Invariance issues

- Scale? Rotation? View point? Occlusions?
  - Implicit
  - Depends on detectors and descriptors



Kadir and Brady. 2003

Going back to one of the questions we asked at the beginning: Is a BoW representation invariant (or robust) to geometric transformations such as scale, rotation, view point as well as to occlusions? By construction, BoW models disregard any information related to the location of features in the images. Thus, they are invariant to scale and rotation as long as the visual words themselves are also scale and rotation invariant. This is the case (at least in theory) if one uses rotational normalized SIFT feature descriptors that are equipped with a DOG detector (which retains the characteristic scale of a local patch).



So far we have discussed how to characterize an image of an object with a histogram of codewords using a dictionary of codewords. This characterization can be used to represent an object category model (say of class k) as a collection of histograms computed from the training images of class k (in other words, each training image of class k can be associated to a histogram; the collection of such histogram yields the model of class k). From these collections of histograms for each class, we can learn a multiclass classifier.

Next Lecture

- Object classification – BoW models part 2
- 2D object detection

Appendix

Object categorization: the statistical viewpoint

$$p(zebra \mid image)$$

vs.

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

- Bayes rule: $p(A|B) = \frac{p(B|A)\ p(A)}{p(B)}$

$$\frac{p(zebra \mid image)}{p(no\ zebra \mid image)}$$

Now let's see an example. An object classification problem can be modeled by asking: given this image what is the probability that there is a zebra v.s. the probability that there isn't? We can calculate this ratio, and if the ratio is larger than 1, we conclude that there is a zebra in the image, or vice versa.

Object categorization: the statistical viewpoint

$$p(zebra \mid image)$$

vs.

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

- Bayes rule: $p(A|B) = \frac{p(B|A)\ p(A)}{p(B)}$

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

Using Bayes rule, we can expand the ratio as a product of the likelihood ratio and the prior ratio. Notice the conditional probability is reversed in the likelihood ratio. The prior simply indicates the probability of finding a zebra (or a non-zebra) in a image. For instance, if we know that the picture is taken at the north pole, the prior of finding a zebra in the image is very low.

Object categorization: the statistical viewpoint

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

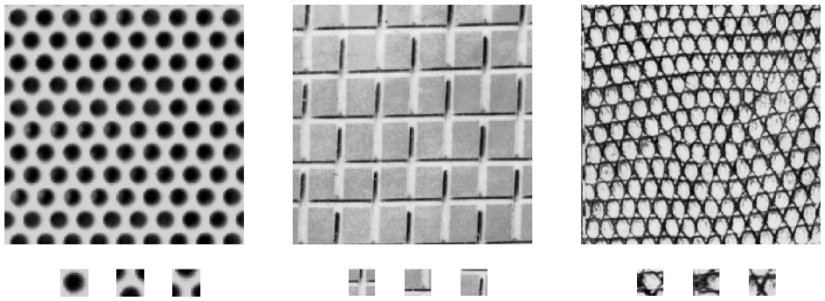
- Bayes rule:

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

Both sides of the equation are equivalent. The ratio on the left side directly estimates posterior probabilities, and lead to discriminative methods. These methods make no attempt to model the underlying probability distributions and seek to maximize the classification performance. The product on the right hand side models the class-conditional PDFs (probability density function) and prior probabilities, and lead to generative methods.

Representing textures

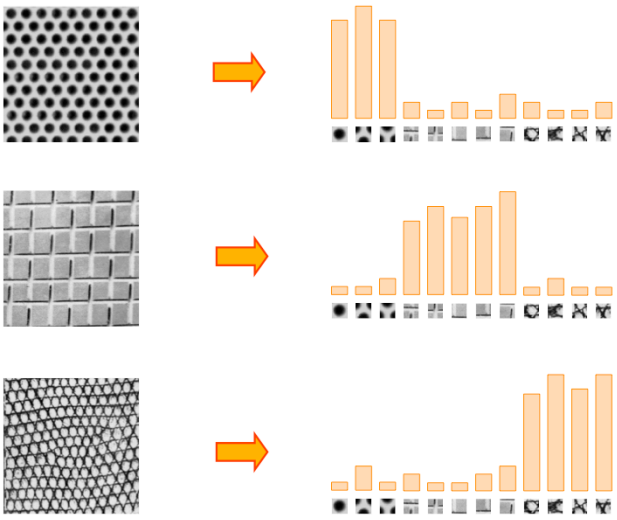
- Texture is characterized by the repetition of basic elements or *textons*
- For stochastic textures, it is the identity of the textons, not their spatial arrangement, that matters



Julesz, 1981; Cula & Dana, 2001; Leung & Malik 2001; Mori, Belongie & Malik, 2001; Schmid 2001; Varma & Zisserman, 2002, 2003; Lazebnik, Schmid & Ponce, 2003

Credit slide: S. Lazebnik

Representing textures



Julesz, 1981; Cula & Dana, 2001; Leung & Malik 2001; Mori, Belongie & Malik, 2001; Schmid 2001; Varma & Zisserman, 2002, 2003; Lazebnik, Schmid & Ponce, 2003

Credit slide: S. Lazebnik