Computer Vision (7CCSMCVI / 6CCS3COV)

Recap

- **Image formation**
- **Low-level vision**
- **Mid-level vision**
  - Segmentation and grouping
  - Correspondence problem
    - Stereo and Depth
    - Video and Motion
- **High-level vision**
  - Object recognition ←Today

Today

- What is Object Recognition
  - Identification
  - Categorisation
  - Localisation
- Methods for performing Object Recognition
  - template matching
  - sliding window
  - edge matching
  - model-based
  - intensity histograms
  - implicit shape model
  - SIFT feature matching
  - bag-of-words
  - geometric invariants

Object recognition tasks

Identification

Determine identity of an individual instance of an object

Clinton vs Bush

Samsung Galaxy On8 vs iPhone 7 Plus

Object recognition tasks

Classification

Determine the category of an object

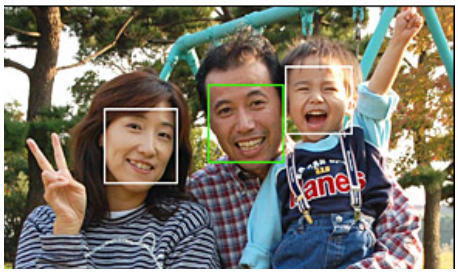
Human vs Chimpanzee

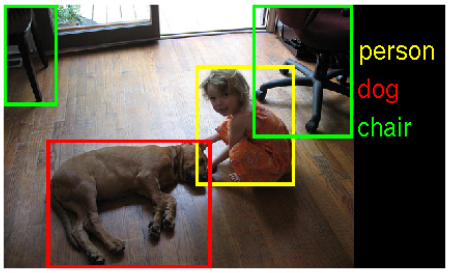
Telephone vs Calculator

Object recognition tasks

Localisation

Determine presence of and/or location of object in an image

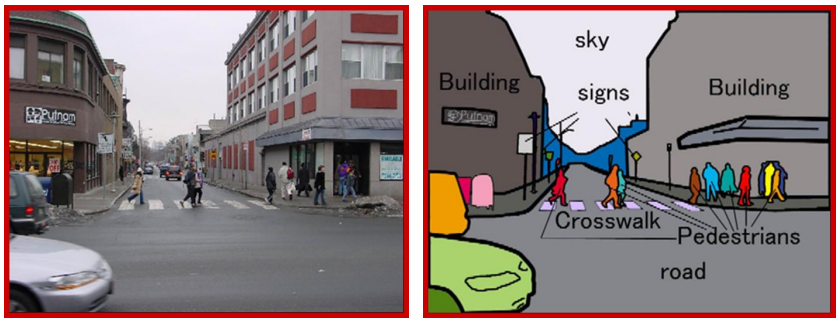




Object recognition tasks

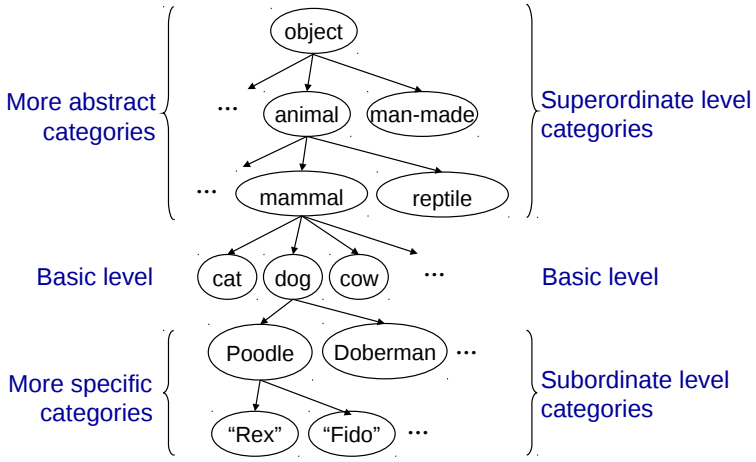
Localisation

If localisation is sufficiently fine grained and for a sufficiently large number of categories, then result can be like image segmentation (called “semantic segmentation”)



Category hierarchy

Classification can take place across a hierarchy of different category types.



Identification is classification at one extreme in this hierarchy.

Category hierarchy

The basic level has a special significance for human object recognition.

It is the level that:

- humans are usually fastest at recognizing category members.
- humans usually start with basic-level categorization *before* doing identification.
- is first named and understood by children.

Category hierarchy

The basic level is the:

- highest level at which category members share many common features
  - i.e. basic level is perceptually homogeneous
  - e.g. apples are similar shape, size, texture, colour, etc.
  - c.f. the next level up, fruits, don't have many features in common
- lowest level at which category members have features distinct from other categories at the same level
  - i.e. basic level is relatively easy to discriminate
  - e.g. apple vs banana vs grape, etc.
  - c.f. the next level down, Brameley vs Coxs' vs Granny Smiths

Object recognition requirements

Object recognition requires:

Sensitivity to (possibly small) image differences relevant to distinguishing one object identity/category from another.

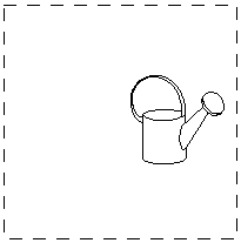
Insensitivity or tolerance to (possibly large) image differences that do not affect object identity or category.

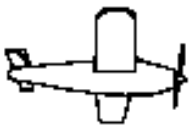
Object recognition requirements

e.g. Insensitivity to “background” clutter, orientation and occlusion.

Object recognition requirements

e.g. Insensitivity to viewpoint





e.g. Insensitivity to lighting

Object recognition requirements

e.g. Insensitivity to non-rigid deformations

e.g. Insensitivity to within category variation

Object recognition procedure

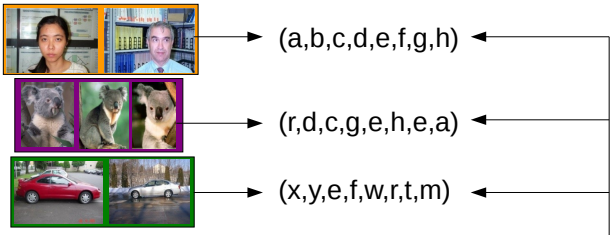
Associating information extracted from images with objects

- Requires image data
- Requires *representations* of objects
- Requires *matching* techniques

Object recognition procedure

Off-line:

Extract representations from training examples



On-line:

Extract representation from input image

Match image with training examples to determine object class or identity

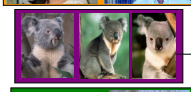
Object recognition procedure

Methods vary in terms of:

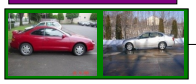
representation used:

local vs global features; 2D vs 3D; pixel intensities vs other features

(a,b,c,d,e,f,g,h)



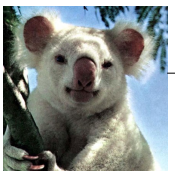
(r,d,c,g,e,h,e,a)



(x,y,e,f,w,r,t,m)

matching procedure:

top-down vs bottom-up; measure of similarity, classification criteria



(r,d,c,h,e,h,e,b)

Template matching

Representation:

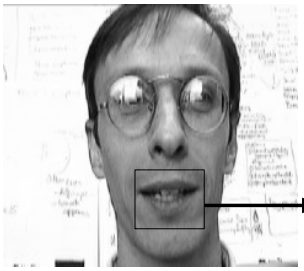
A template is an image of the object that is to be recognized (i.e. an array of pixel intensities).

Matching:

For every template:

- Search every image region
- Calculate similarity between template and image region

Choose the "best" match, or all matches where similarity exceeds a threshold



template:

$= I_1$

image region:

$= I_2$

Similarity Measures

We can **maximise** the following measures:

Cross-correlation: $\sum_{i,j} I_1(i,j) I_2(i,j)$

If I_1 and I_2 are considered to be vectors in feature space, then the cross-correlation is the dot-product of these vectors $I_1 . I_2 = \|I_1\| \, \|I_2\| \cos\theta$

Note: template matching using cross-correlation can be implemented using convolution and a rotated template.

Normalised cross-correlation (NCC): $\frac{\sum_{i,j} I_1(i,j) I_2(i,j)}{\sqrt{\sum_{i,j} I_1(i,j)^2} \sqrt{\sum_{i,j} I_2(i,j)^2}} = \frac{I_1 . I_2}{\|I_1\| \, \|I_2\|} = \cos\theta$

Correlation coefficient: $\frac{\sum_{i,j} \left(I_1(i,j) - \bar{I}_1\right)\left(I_2(i,j) - \bar{I}_2\right)}{\sqrt{\sum_{i,j} \left(I_1(i,j) - \bar{I}_1\right)^2} \sqrt{\sum_{i,j} \left(I_2(i,j) - \bar{I}_2\right)^2}}$

equals normalised cross-correlation if means are zero

Similarity Measures

We can **minimise** the following measures:

Sum of Squared Differences (SSD): $\sum_{i,j} \left(I_1(i,j) - I_2(i,j)\right)^2$

Euclidean distance: $\sqrt{SSD} = \sqrt{\sum_{i,j} \left(I_1(i,j) - I_2(i,j)\right)^2}$

Sum of Absolute Differences (SAD): $\sum_{i,j} |I_1(i,j) - I_2(i,j)|$

Template matching

To recognise multiple objects use multiple templates.

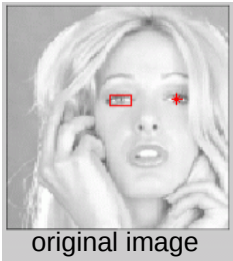
mouth template

eye template

nose template

Template matching: example

Template of right eye is flipped and used to locate left eye

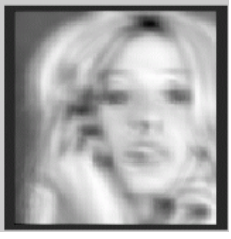


original image

Sum of absolute differences

note SAD peak over the left eye (cross in left image)

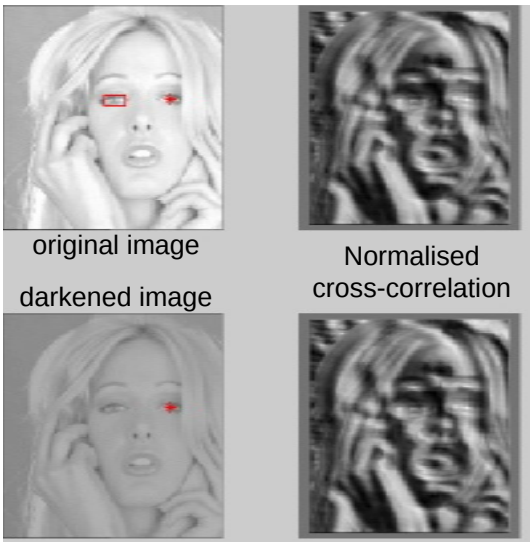
darkened image



note SAD peak over forehead (cross in left image)

Template matching: example

Template of right eye is flipped and used to locate left eye



note NCC peak over the left eye (cross in left image)

note NCC peak over the left eye (cross in left image)

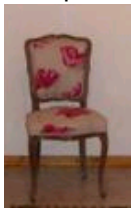
Template matching: problems

Distinguishing true matches from false matches

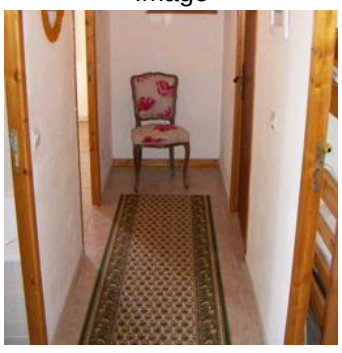
What constitutes a match? How many peaks?

Template matching: problems

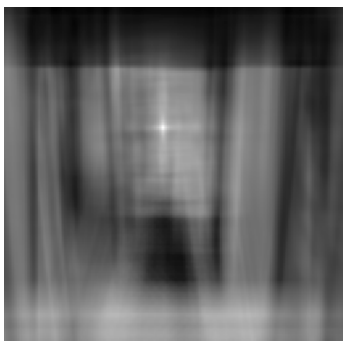
Template



Image



NCC

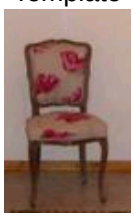


Template matching: problems

Image

NCC

Template



Template needs to be very similar to the target object

If an object appears scaled or rotated in the image, the match with its template will not be good.

Template matching: problems

Hence, to provide tolerance to viewpoint / within category variation it is necessary to use multiple templates for each object.

Template matching: problems

With so many comparisons, the threshold needs to be high, otherwise we will almost always find false matches.

However, because any deviation between the template and the image patch will result in a weak match, the threshold needs to be low to find all the true matches.

Template matching: problems

Tractability is an issue, especially if we need to deal with a wide range of variations in appearance.

e.g. for a small 250 by 200 pixel image, if we need to recognize 5 object, each of which can occur at 30 viewpoints and 5 scales then we need to perform 37.5 million matches!

Template matching: problems

If an object appears occluded, then this may result in a template failing to match.

i.e. template matching is sensitive to occlusion.

Template matching: problems

The underlying issue is that the metric used for comparison is fundamentally not robust to changes in appearance between the template and the image patch.

- even small changes in scale, orientation, and viewpoint, as well as changes due to within class variation, non-rigid deformations, or occlusion will result in weak matches (indistinguishable from false matches)

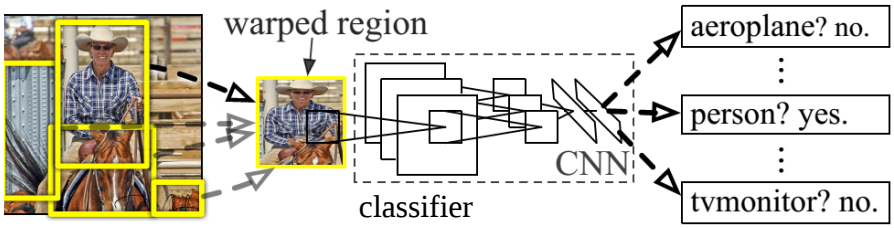
Hence

- invariance is difficult to achieve

Sliding window

Like template matching, except:

1) For each image patch, use a classifier to determine if it contains the object (i.e. replace simple comparison of intensity values with method that is more tolerant to changes in appearance).
2) Choose image patches of different shapes and sizes and resize them before inputting to the classifier (to also increase tolerance to changes in appearance).

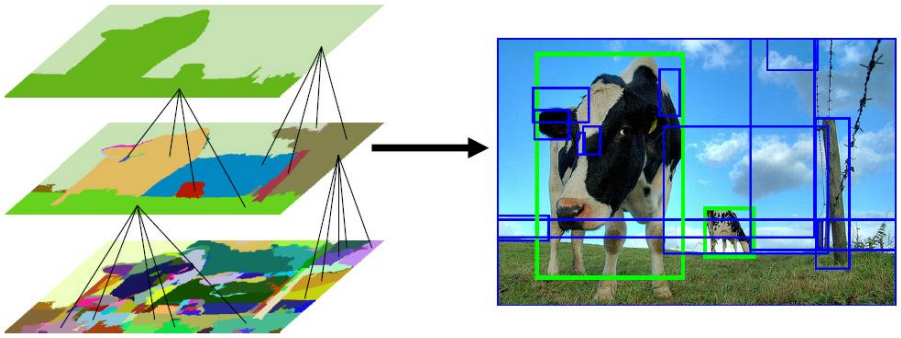


Sliding window

Classifying every image region of every possible size and shape (typically 100k+ regions) is very computationally expensive!

To improve tractability, images can be pre-processed to select regions (typically 1k+ regions) that are good candidates to contain an object.

This pre-processing is typically done using image segmentation.



Edge matching

Representation:

Like template matching, but template and input images pre-processed to extract edges

Matching:

For every edge template:

Search every image region

Calculate average of the minimum distances between points on the edge template (T) and points on the edge image (I)

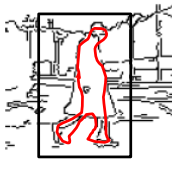
$$D(T,I)=\frac{1}{|T|}\sum_{t\in T} d_I(t)$$

Choose the minimum value as best match


Input image


Edges detected


Template shape

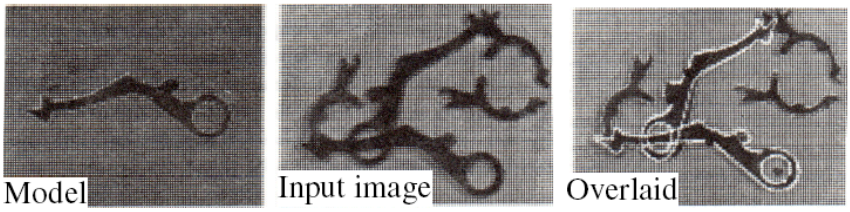

Best match

Model-based object recognition

General idea

- Hypothesize object identity and pose
- Render object in image ("back-project")
- Compare to image

e.g. 2D:

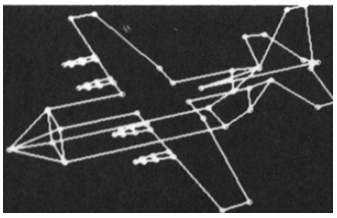


Model-based object recognition

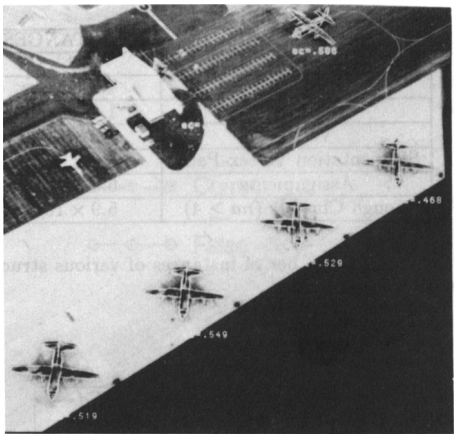
General idea

- Hypothesize object identity and pose
- Render object in image ("back-project")
- Compare to image

e.g. 3D:



model



overlaid

Model-based object recognition

Representation:

2D or 3D model of object shape

Matching:

Comparison of **back-projected** model with image, using:

- Edge score
  - are there image edges near predicted object edges?
  - very unreliable; in texture, answer is usually yes
- Oriented edge score
  - are there image edges near predicted object edges with the right orientation?
  - better, but still hard to do well

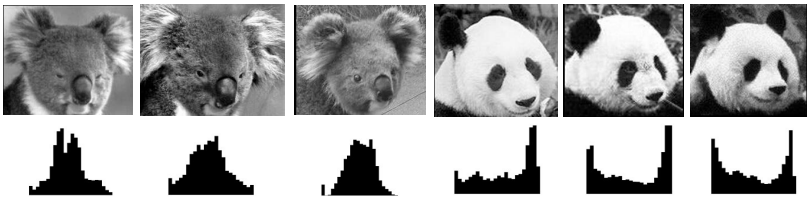
Intensity histograms

Representation:

Histogram of pixel intensity values (either greyscale or colour).

Matching:

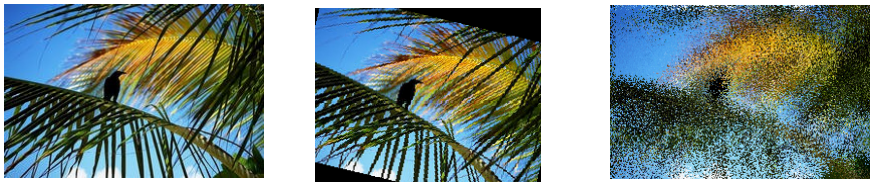
Compare histograms to find closes match

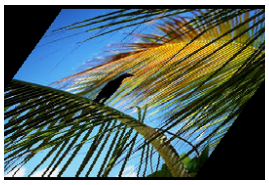


- ✓ Histogram is fast and easy to compute.
- ✓ Insensitive to small viewpoint changes (unlike templates)
- ✗ Sensitive to illumination and intra-class appearance variation
- ✗ Insensitive to different spatial configurations (as spatial information not represented)

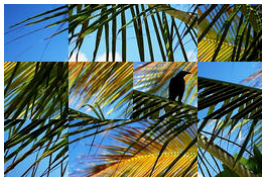
Intensity histograms

All these images have the same colour histograms





Insensitivity to viewpoint changes (useful)

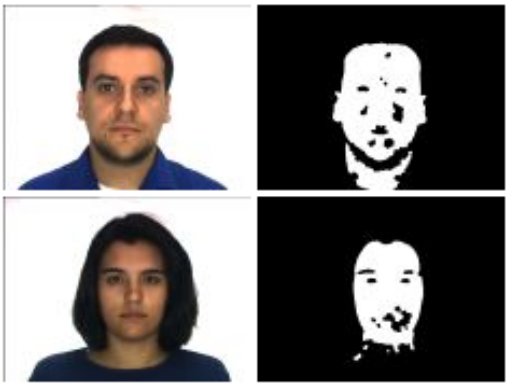


Insensitivity to spatial configuration (not always useful)

Intensity histograms

Skin has a very small range of (intensity independent) colours.

Hence, colour histograms are often used as part of face detection/recognition algorithms.



original images

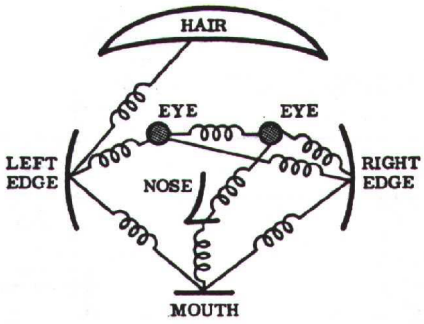
binary images segmented using skin colour

Implicit Shape Model (ISM)

Representation:

has two components

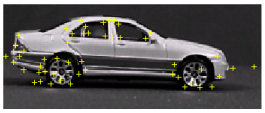
- parts
  (2D image fragments)
- structure
  (configuration of parts)



Implicit Shape Model (ISM)

Extraction of local object features:

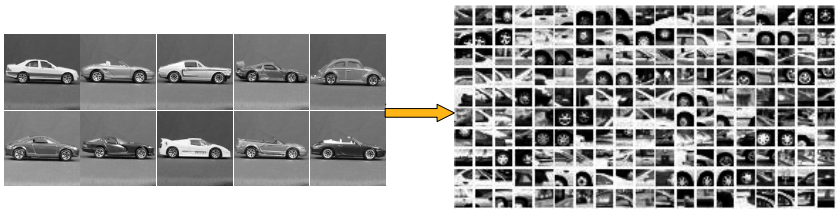
Training image



Locate interest points (e.g. using Harris detector)

Extract 2D image fragments from around each interest point

Collect 2D patches (features) from whole training set:



Implicit Shape Model (ISM)

Create appearance codebook:

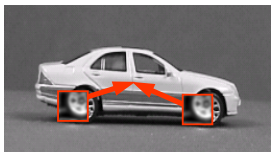
- Cluster features e.g. using hierarchical agglomerative clustering.
- Store cluster centers as *Appearance Codebook*

Implicit Shape Model (ISM)

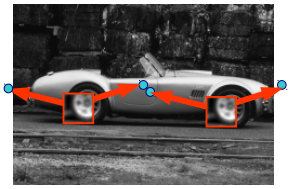
Learn configuration of parts:

– Match codebook features to training images

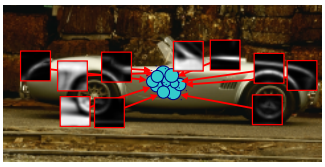
– For every codebook entry record possible object centres



During matching procedure, each features votes for possible object centres:



one feature

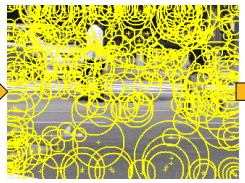


many features

Implicit Shape Model (ISM)

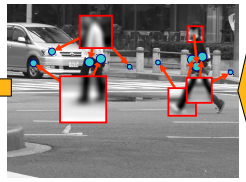
Matching:

Input image

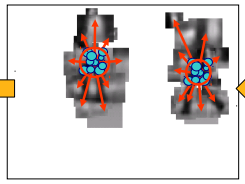


Interest Points

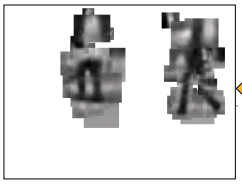
Matched Codebook Entries



Voting



Backprojection of Maxima

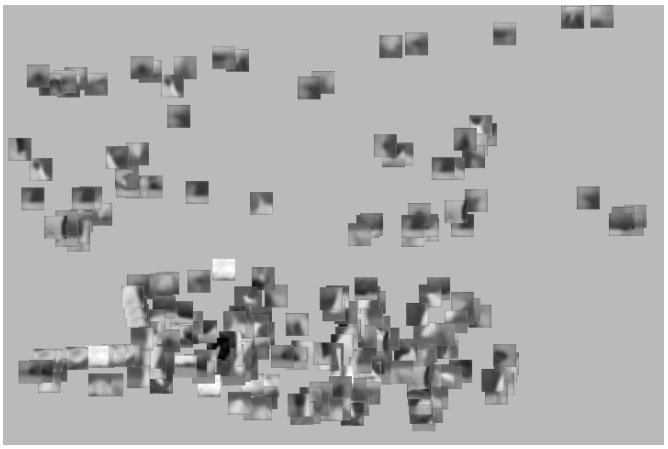


Backprojected Hypotheses

Implicit Shape Model (ISM): example

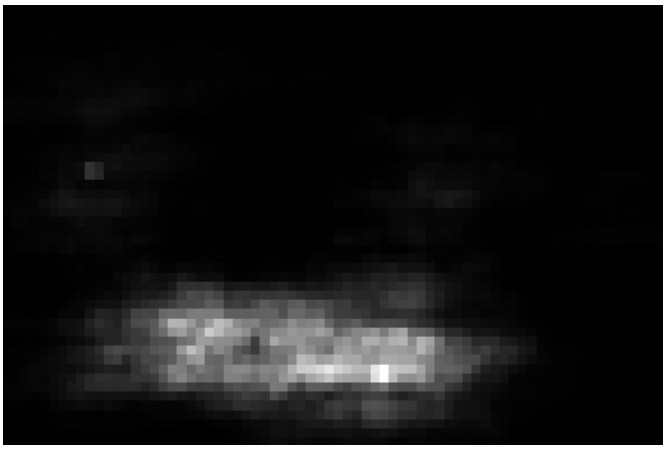
Interest points

Implicit Shape Model (ISM): example



Matched patches

Implicit Shape Model (ISM): example



Votes

Implicit Shape Model (ISM): example

1st hypothesis

Implicit Shape Model (ISM): example

2nd hypothesis

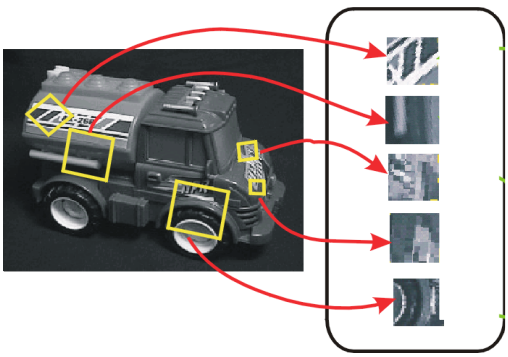
Implicit Shape Model (ISM): example

3rd hypothesis

Feature-based object recognition

Representation:

Training image content is transformed into local features that are invariant to translation, rotation, and scale



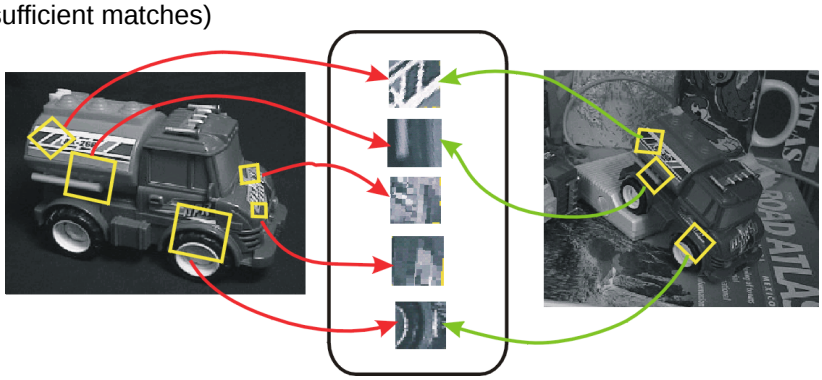
Local Features

Could be patches, but more likely to be descriptors
(like SIFT feature vectors)

Feature-based object recognition

Matching:

Local features are extracted from new image in same way, and matched to those from training image (object recognized if there are sufficient matches)



Local Features

Feature-based object recognition

- Feature detection needs to be repeatable despite:
  - Translation, rotation, scale changes
  - Lighting variations
- Feature detection needs to find sufficient features to cover the object

  e.g. Harris corner detector, SIFT interest point detector.

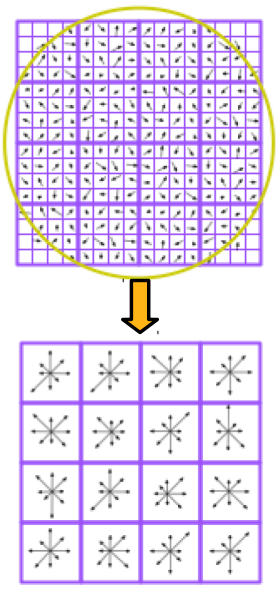
- The features should contain “interesting” structure that can be reliably matched
- The feature description should be invariant to:
  - Translation, rotation, scale changes
  - Lighting variations

  e.g. SIFT descriptor, or one of many others that have been proposed

SIFT feature matching

Representation:

A 128 element histogram of the orientations of the intensity gradients (binned into 8 orientations) in 4x4 pixels windows around the interest point, normalized so that vector has unit length and rotated so that dominant orientation is vertical.

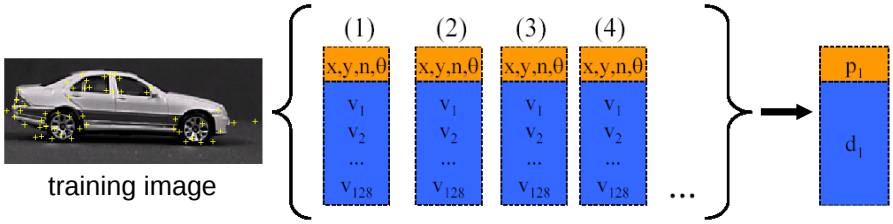


- **Locality:** features are local, so robust to occlusion and clutter (no prior segmentation)
- **Distinctiveness:** individual features can be matched to a large database of objects
- **Quantity:** many features can be generated for even small objects
- **Efficiency:** close to real-time performance

SIFT feature matching

Representation:

- A set of keypoints are obtained from each training image
- Each such keypoint has a descriptor which is a 128 components vector
- All (keypoint location, image number, feature vector) sets are stored in a database

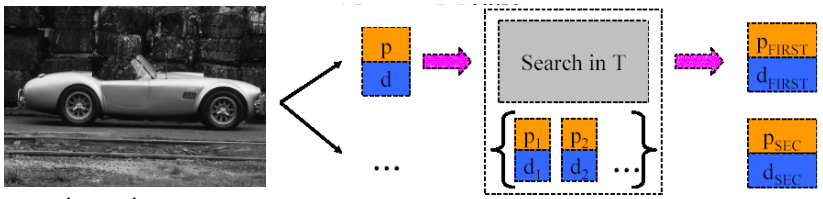


training image

SIFT feature matching

Matching:

- An input image gives a new set of (keypoint, vector) pairs
- For each such pair, find the top 2 best matching descriptors in the training database

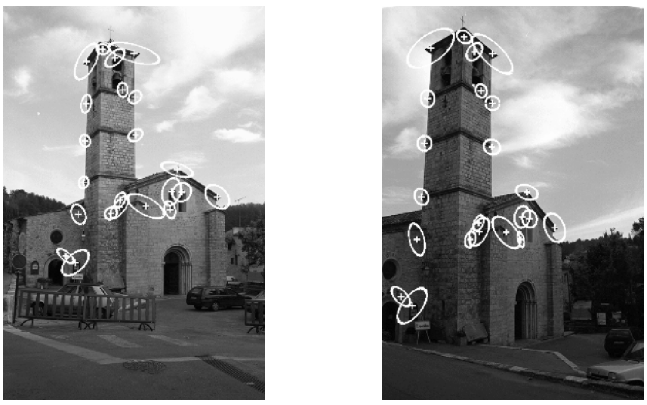


input image

- Match accepted IF
  - Ratio of distance to first nearest descriptor to that of second < threshold

SIFT feature matching: example

Recognition with changes in viewpoint and illumination



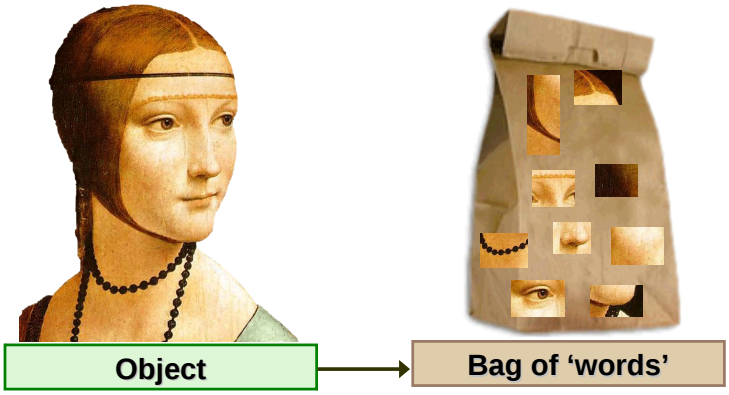
22 correct matches

SIFT feature matching: confirmation

Given three non-collinear model points P_1, P_2, P_3, and three image points p_1, p_2, p_3, there is a unique transformation (rotation, translation, scale) that aligns the model with the image.

Use RANSAC or Generalised Hough Transform to determine if matched locations are consistent (i.e. can be modelled by the same transformation from one view to another).

Bag-of-words



Analogous to the method used for document recognition by Google.

Bag-of-words (for documents)

Representation:

- Documents are parsed into words
  - e.g. "the cat sat on the mat, a dog is sitting on the cat"
- Common words are ignored (the, a, on, it, he, etc.)
  - e.g. "cat sat mat dog sitting cat"
- Words are represented by their stems (e.g. 'sitting', 'sat', 'sits' all become 'sit')
  - e.g. "cat sit mat dog sit cat"
- Each word is assigned a unique identifier
  - e.g. 1 = "cat", 2= "sit", 3= "mat" 4= "dog" 5="fish"
- Each document is represented by a K components vector of words frequencies (where K is the total size of the vocabulary extracted from all documents)
  - e.g. (2, 2, 1, 1, 0, ...)

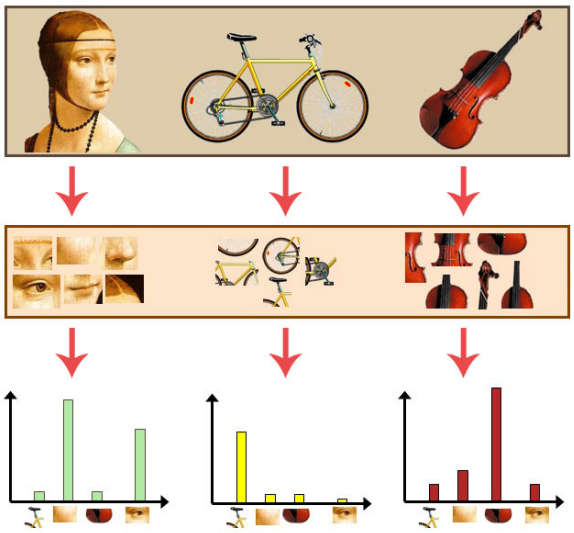
Bag-of-words (for documents)

Matching:

- The query is represented in the same format.
  - e.g. “cats and dogs” → (1, 0, 0, 1, ...)
- For all documents containing at least one of the query words, calculate the angle (i.e. $\cos^{-1}$NCC) between the query and document vectors
- Rank the results

Bag-of-words (for images)

Different objects have distinct sets of features that occur in different frequencies:

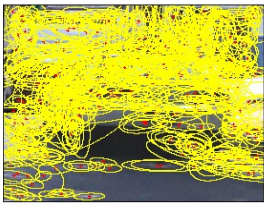


Represent objects as a distribution (histogram) of feature occurrences.

Bag-of-words

Choosing features:

Regular grid



Interest point detector

Randomly

Bag-of-words

Encoding features:

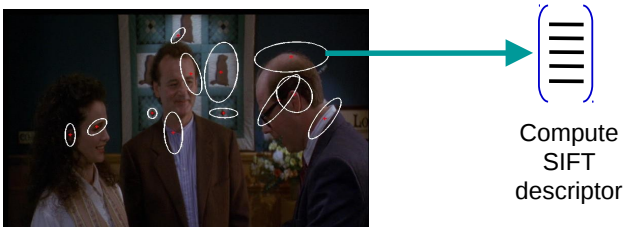
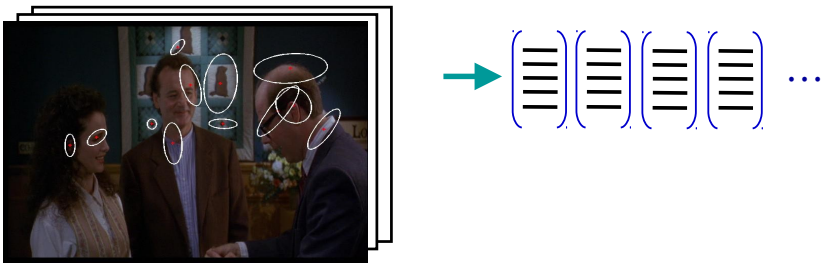


Image patches around the chosen locations are encoded using descriptor (analagous to a word)

Bag-of-words

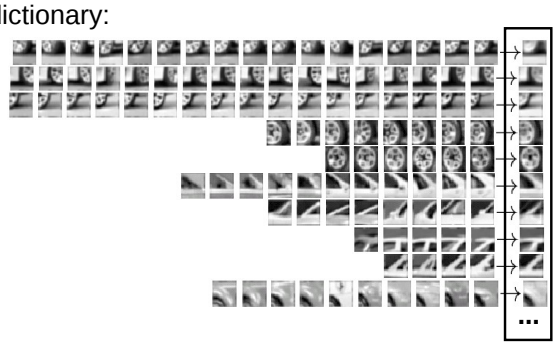
Creating dictionary:



Encode many features taken from many images

Bag-of-words

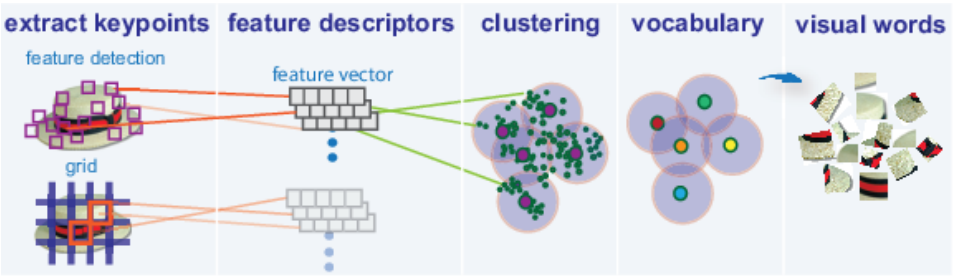
Creating dictionary:



- cluster feature descriptors into K groups using K-means clustering algorithm (analogous to representing words by their stems)
- each cluster represents a “visual word” or codeword.
- the whole set of clusters represents a “visual vocabulary” or codeword dictionary
- The most frequent codewords that occur in almost all images are removed from the dictionary (analogous to ignoring common words)

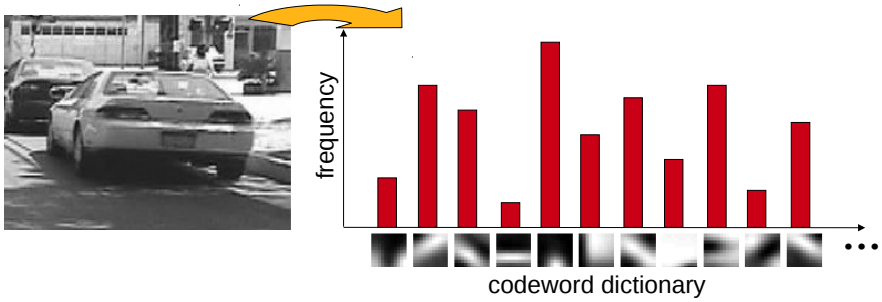
Bag-of-words

Creating dictionary (summary):



Bag-of-words

Representing images:



Representation:

Each image is represented as a histogram showing the frequency of appearance of each of the codewords in the dictionary.

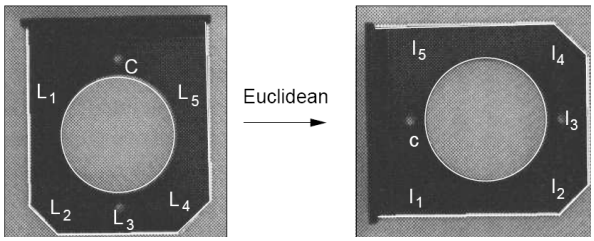
Matching:

Images compared (and hence a match between an input image and a training image) is found by calculating the distance between histograms.

Geometric invariants

A geometric invariant is a property of an object in the scene, which does not vary with viewpoint.

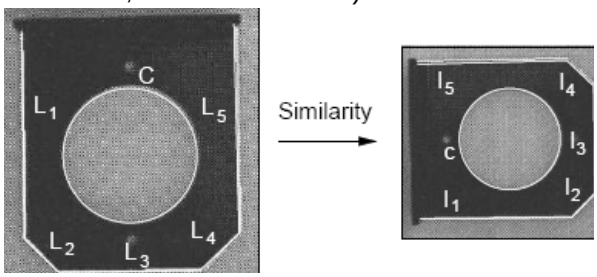
e.g. if we consider viewpoint changes in Euclidean space (i.e. translation and rotation)



invariant properties:
- lengths
- angles
- areas

Geometric invariants

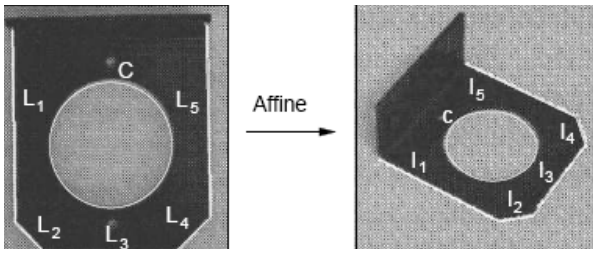
e.g. if we consider viewpoint changes in Similarity space (i.e. translation, rotation and scale)



invariant properties:
- *ratios* of lengths
- angles

Geometric invariants

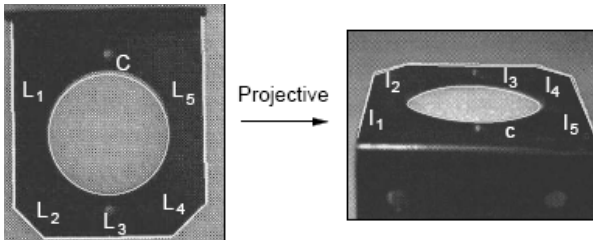
e.g. if we consider viewpoint changes in Affine space (i.e. translation, rotation, scale and shear)



invariant properties:
- parallelism
- *ratios* of lengths *along lines*
- *ratio* of areas

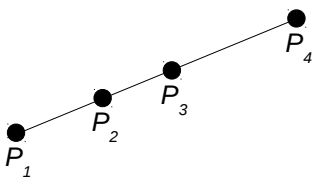
Geometric invariants

e.g. if we consider viewpoint changes in Projective space (i.e. translation, rotation, scale, shear and foreshortening)



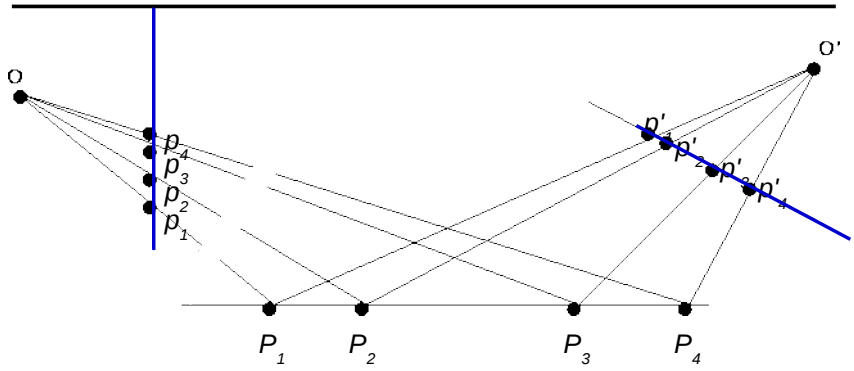
invariant properties:
- *cross-ratio*

The *cross-ratio* is the ratio of ratios of lengths on a line, e.g.



$$\frac{\|P_3-P_1\|\,\|P_4-P_2\|}{\|P_3-P_2\|\,\|P_4-P_1\|}$$

Geometric invariants



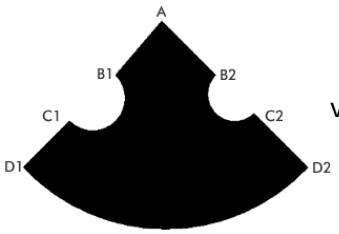
$$\frac{\|P_3-P_1\|\,\|P_4-P_2\|}{\|P_3-P_2\|\,\|P_4-P_1\|}=\frac{\|p_3-p_1\|\,\|p_4-p_2\|}{\|p_3-p_2\|\,\|p_4-p_1\|}=\frac{\|p'_3-p'_1\|\,\|p'_4-p'_2\|}{\|p'_3-p'_2\|\,\|p'_4-p'_1\|}$$

Under perspective projection, the cross ratio remains constant from any viewpoint.

Geometric invariants

Representation: value of cross-ratio
Matching: compare value of cross-ratio measured in image with database of cross-ratios measured in training images.

Problems: occlusion, availability and distinctiveness of the points, e.g.:



VS

Summary

- template matching
- sliding window
- edge matching
- model-based
- intensity histograms
- implicit shape model
- SIFT feature matching
- bag-of-words
- geometric invariants

These are just a small selection of methods – many, many, more methods have been tried!

There are various ways of classifying methods...

Classification of Object Recognition Methods

Matching procedure:

- Top-down (generative)
  - e.g. model-based, templates, edge matching
    - Hypothesise that image contains a certain object and look for it in the image. Expectation-driven. (= “fitting”, see lecture on segmentation).
- Bottom-up (discriminative)
  - e.g. SIFT, bag-of-words, intensity histograms, ISM
    - Extract description and match to descriptions in database. Stimulus-driven.

Classification of Object Recognition Methods

Representation used:

- pixel intensities vs feature vectors vs geometry
  - e.g. templates, edge matching, intensity histograms, ISM
  - e.g. SIFT feature matching, bag-of-words
  - e.g. model-based, geometric invariants
- 2D (image-based) vs 3D (object-based)
  - e.g. templates, sliding window, ISM, bag-of-words
  - e.g. 3D models, geometric invariants, SIFT (check of consistency)
- local features vs global features
  - e.g. SIFT (bag-of-words), ISM, part templates
  - e.g. whole object templates, sliding window, 3D models

Local vs global representation

Local (each object is broken down into simple features)

e.g. A = / + \ + –

Advantages: tolerant to viewpoint, within class variation, occlusion.

Problem: Many objects consist of the same collection of features, and hence can not be distinguished

e.g. T = | + –
L = | + –
\+ = | + –

Problem: If many objects in image, then there is no information about which feature comes from which object

e.g. TA = | + – + / + \ + –
IV = | + – + / + \ + –

Local vs global representation

Gobal (each object is represented by a description of its overall shape)

e.g. A = A

Advantage: can distinguish similar objects

Problems:

- sensitive to viewpoint and within class variation (due to need for exact alignment of representations)
- sensitive to occlusion (due to all parts of description being relevant for matching)

Local vs global representation

Local and global representations have complementary advantages and disadvantages

- local features generate many false positives
- global features generate many false negatives

Solutions:

- use features of intermediate complexity
- use a hierarchy of features with a range of complexities