Specific Object Recognition

Specific objects vs. class-level objects

A *specific object* = an instance of an object class
e.g. “my car” instead of “a car”

Specific objects vs. class-level objects

Traditionally specific object recognition was easier than class recognition

Because there is much more variability between the views of class members

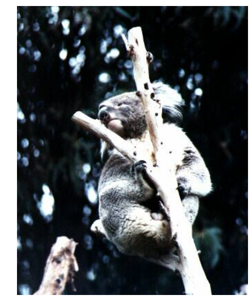
Specific objects vs. class-level objects

Illumination

Object pose

Clutter

Occlusions



Viewpoint

Computer Vision

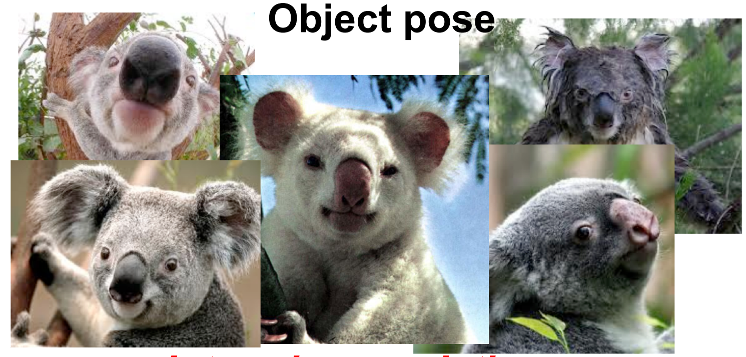
Specific objects vs. class-level objects

Illumination

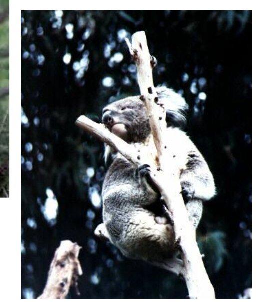
Object pose

Clutter

Occlusions



Intra-class variation



Viewpoint

On top of factors affecting specific object recognition, there is added complexity of intra-class variation… i.e. differences between koala's in this case

Specific objects vs. class-level objects

Intra-class and inter-class variation

The difference between classes can be as small as that between instances of the same class …
yet the distinction needs to be made

Specific objects vs. class-level objects

Traditionally specific object recognition was easier than class recognition

Because there is much more variability between the views of class members

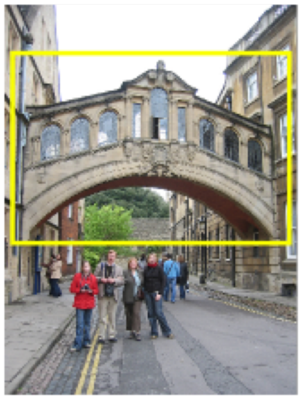
The first reasonably successful class recognition methods were being developed when deep learning made its large-scale entry

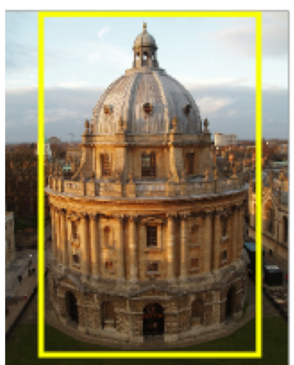
The capability of deep networks to generalize is so good that in deep learning class recognition now dominates.

For deep methods specific object recognition is the more difficult task (fine-grained classification…)

Example app

search photos on the web for particular places



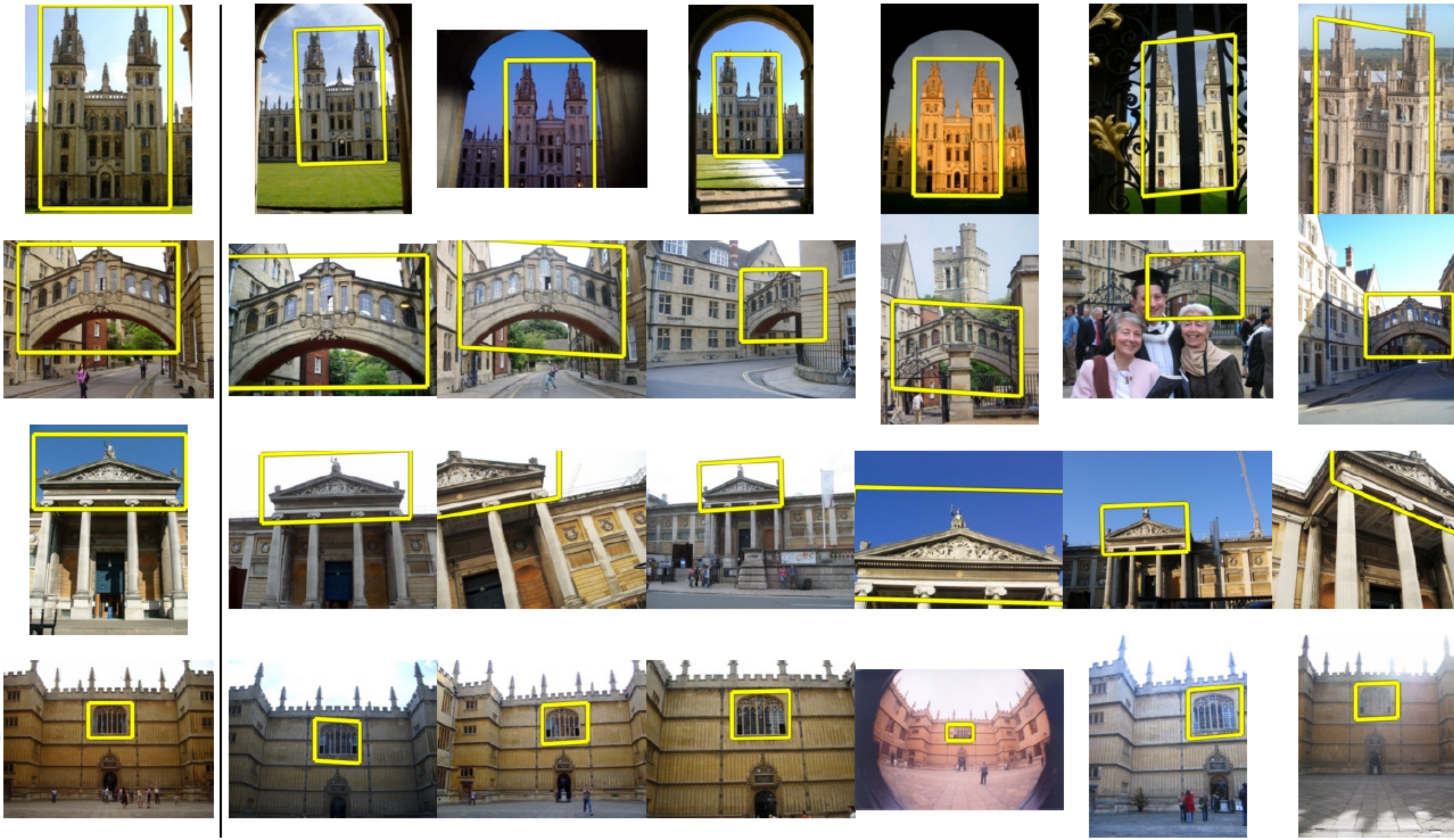


Find these landmarks ...in these images and 1M more

Slide credit: J. Sivic

Computer Vision

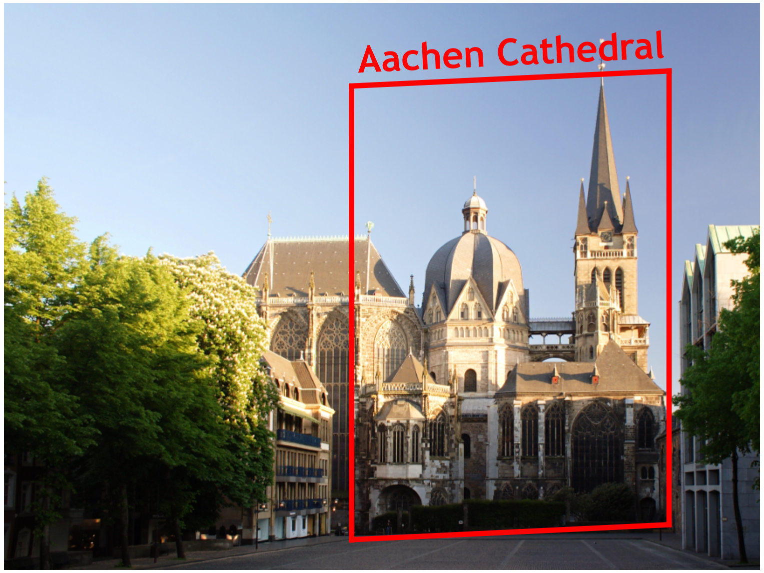
Application: Large-Scale Retrieval



Query Results from 5k Flickr images (demo available for 100k set)

[Philbin CVPR'07]

Example Applications



Mobile tourist guide

- Self-localization
- Object / landmark recognition
- Augmented reality
- Wine label rec.

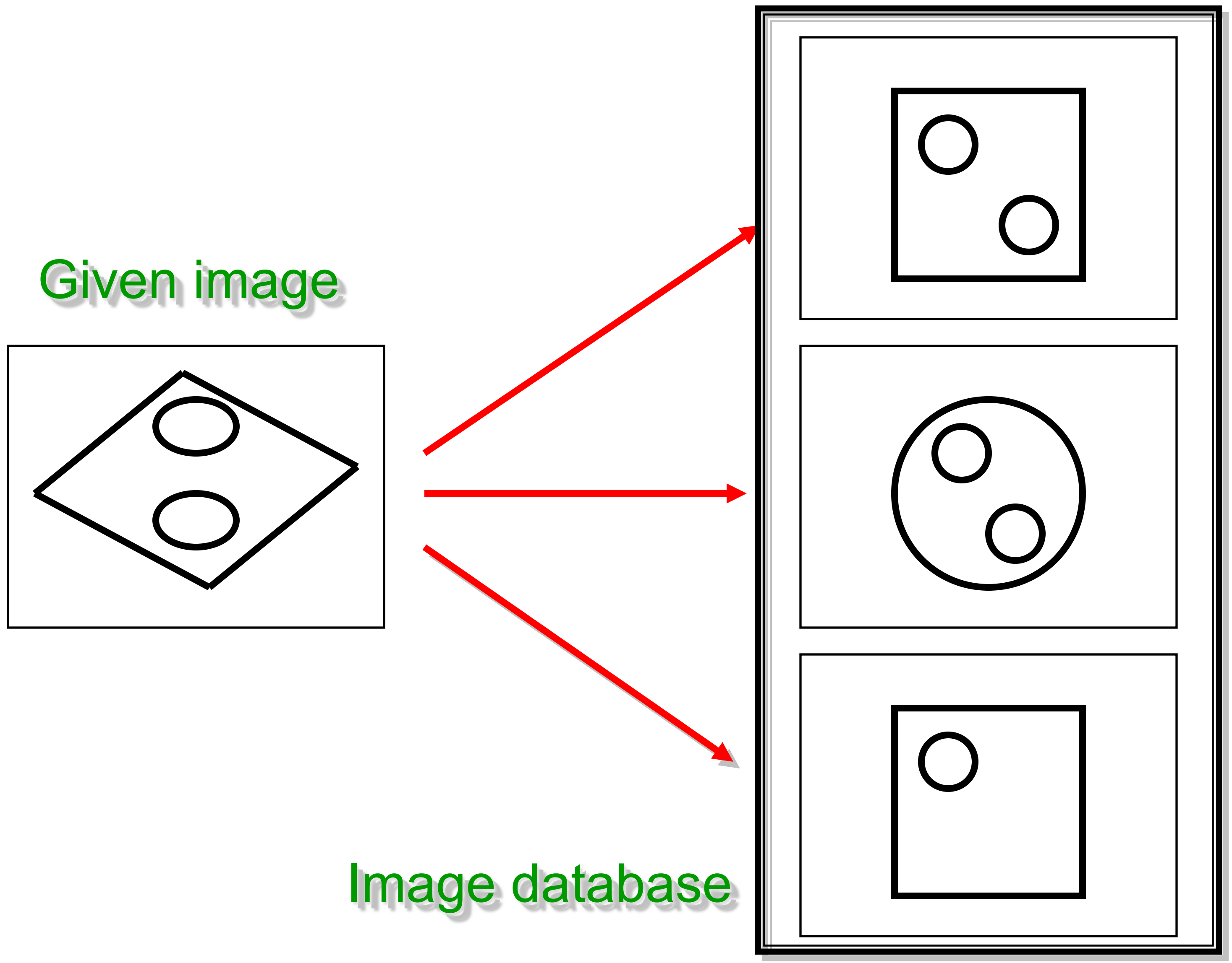
(Vivino, 1st CV app in Samsung SmartWatch powered by kooaba

[Quack, Leibe, Van Gool, CIVR'08]

Model-based

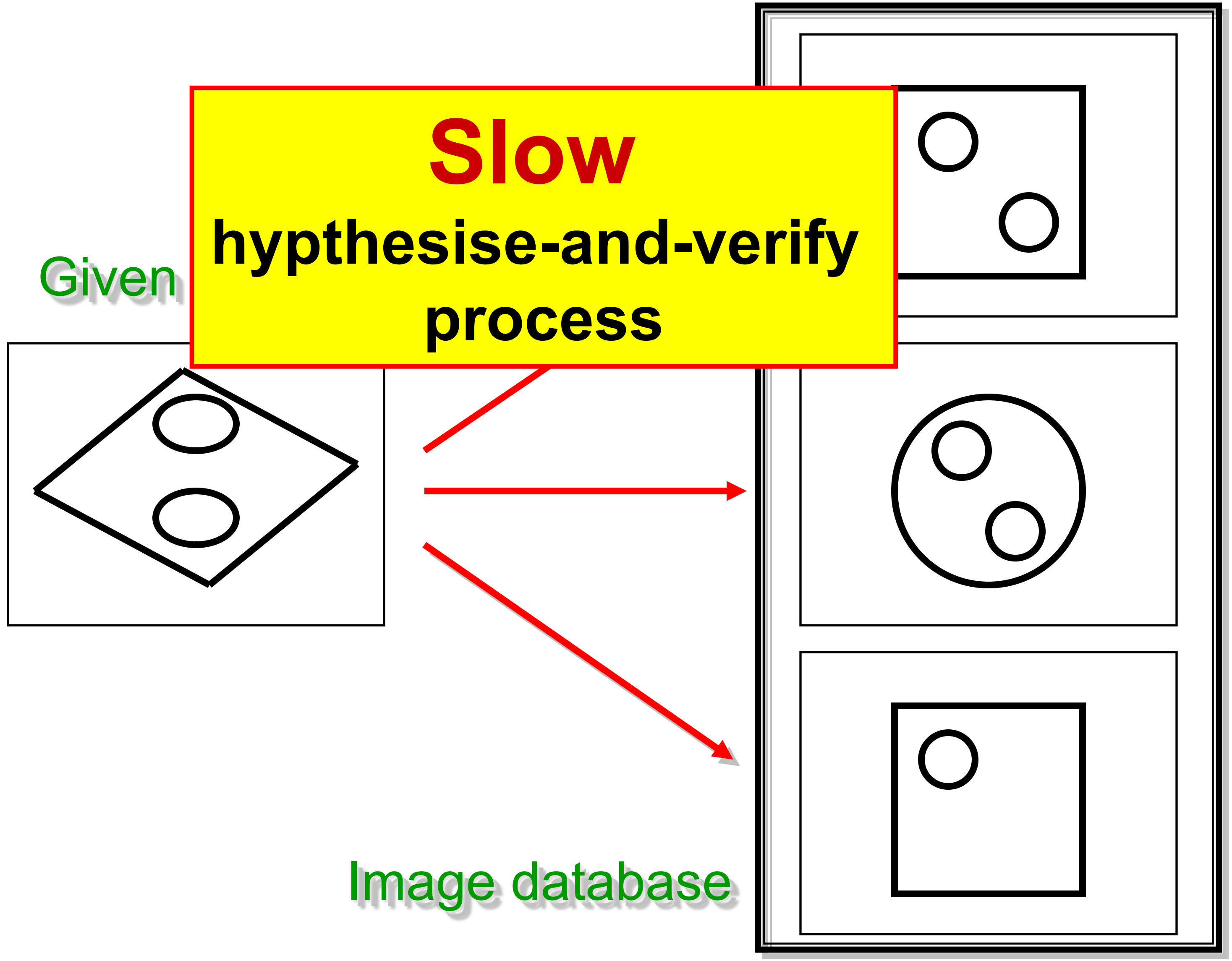
Once upon a time...

comparing image features with features of objects in a database, trying to figure out type + pose



Once upon a time...

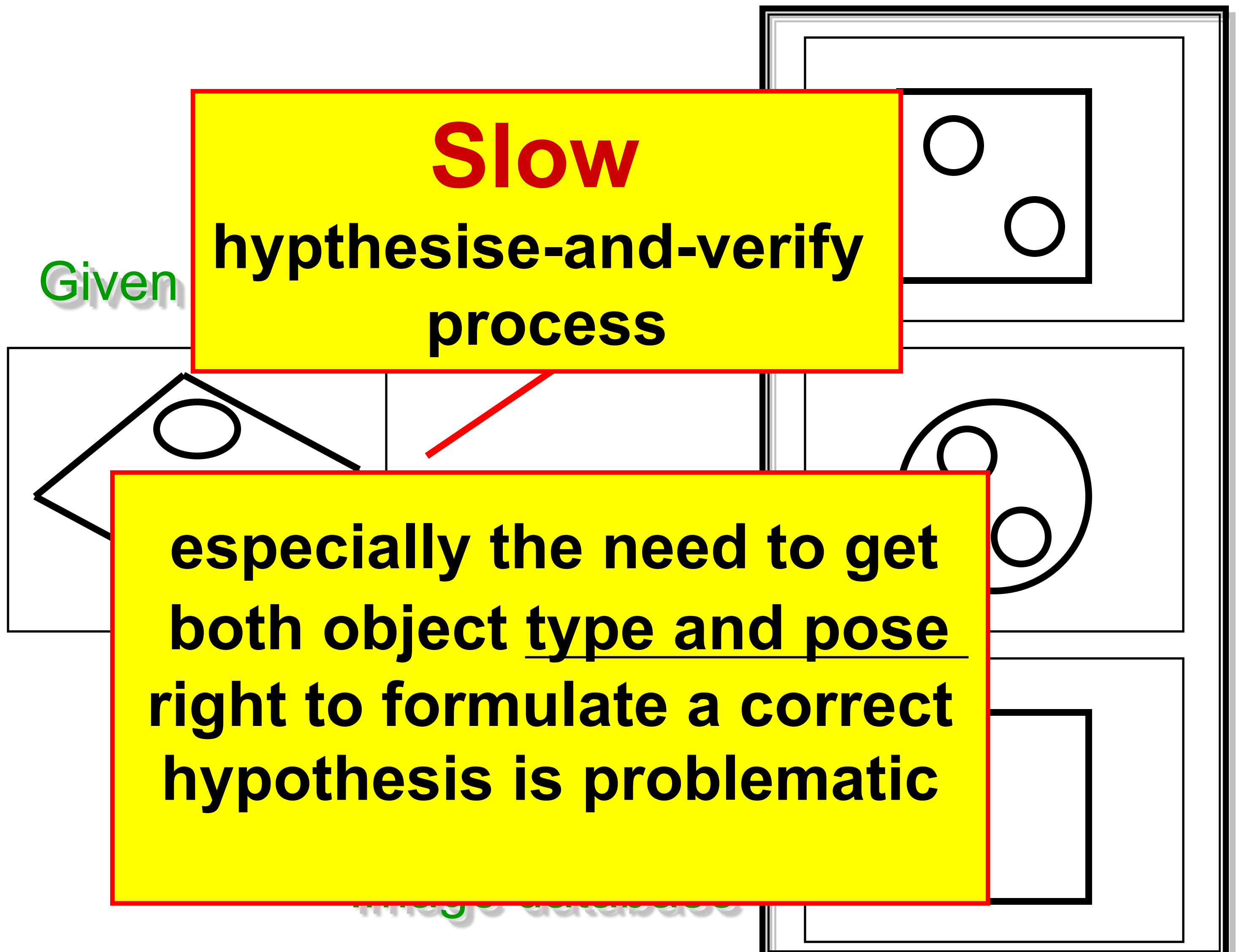
comparing image features with features of objects in a database, trying to figure out type + pose



Computer Vision

Once upon a time...

comparing image features with features of objects in a database, trying to figure out type + pose



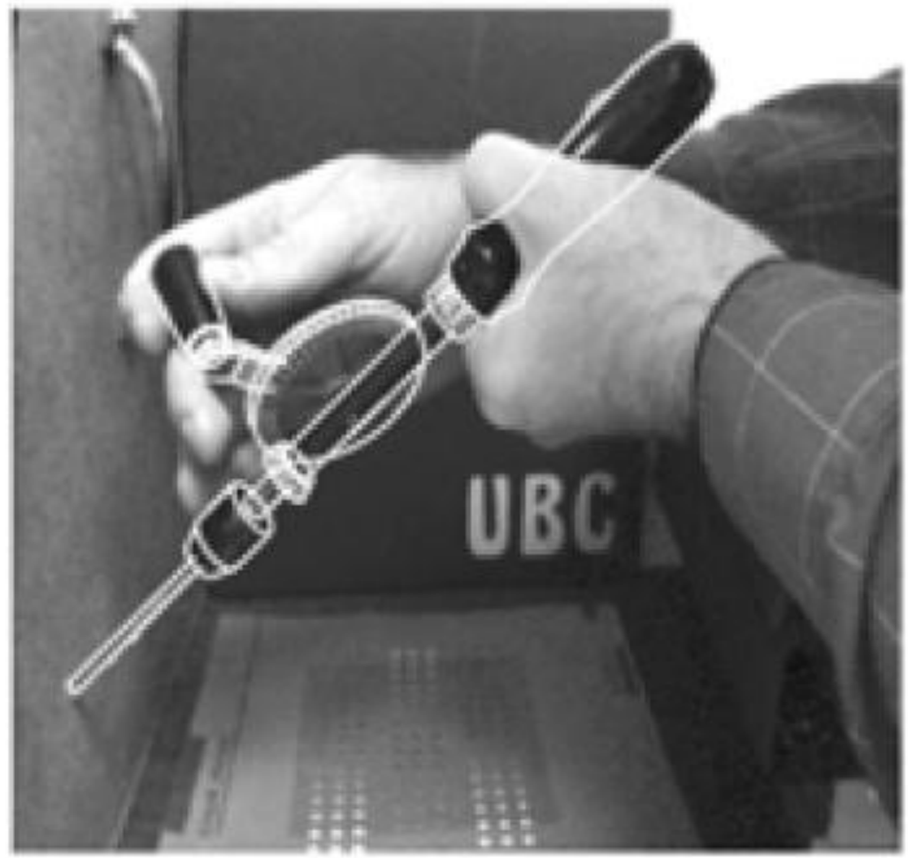
Slow

hypthesise-and-verify process

especially the need to get both object type and pose right to formulate a correct hypothesis is problematic

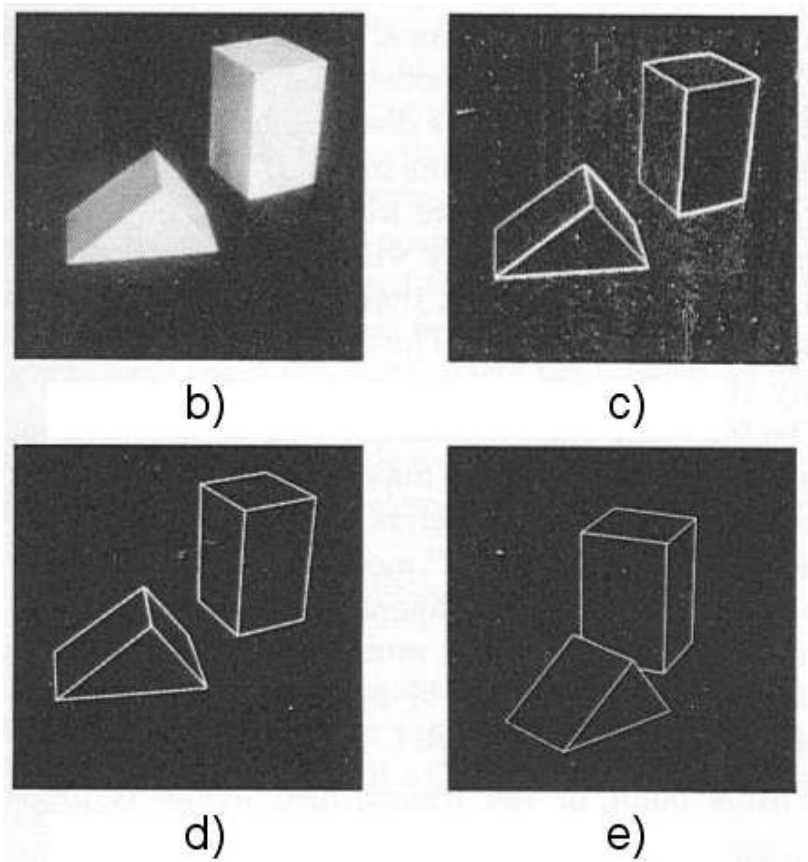
Computer Vision

Model-based approaches



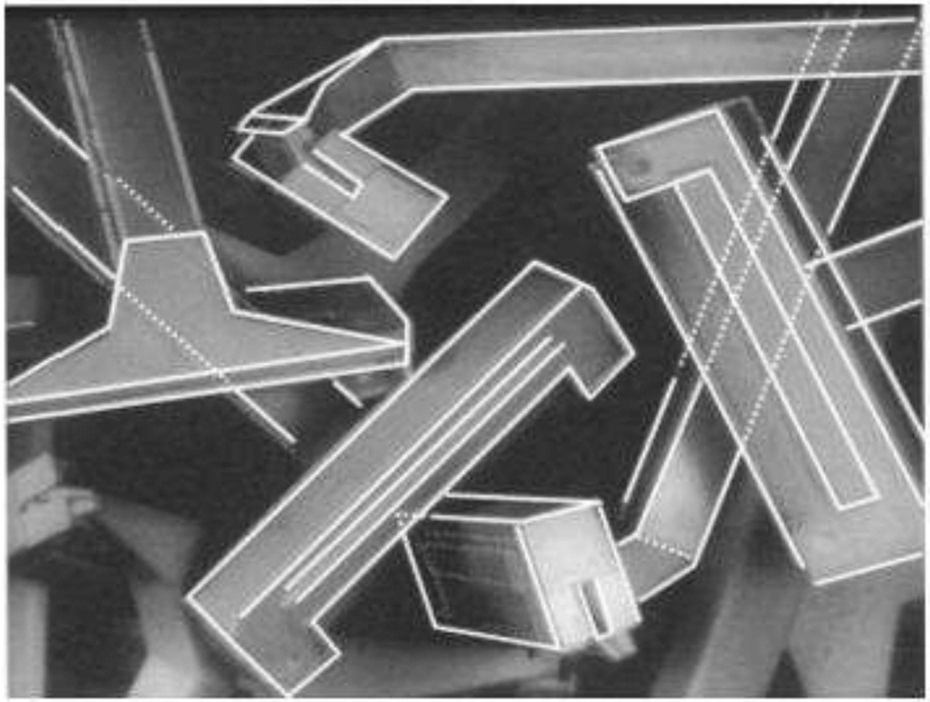
Wireframe model for 3D objects

Early attempts…1965



Blocks world model, Roberts et al., 1965

Early attempts…1985



Dealing with occlusions, Lowe, 1985

More recent Example: Invariant-based recognition of planar shapes

The crucial advantage of invariants is that they decouple object type and pose issues

Computer Vision

Invariant-based recognition of planar shapes

Ex. given here for completely visible planar shapes
- under affine distortions
- using invariant signatures of the outlines

Image on the left is compared against database images of various animals like that of the matching swan on the right

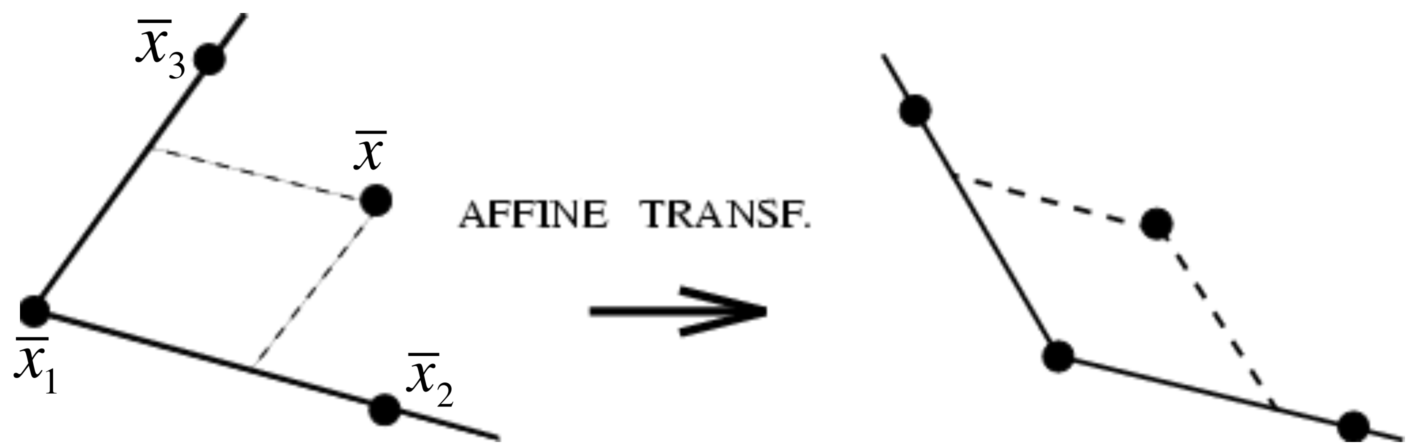
Invariants under affine transf. : ex 1

ratios of areas are affine invariant and the following invariants are based on this

8 *(x,y)* point coordinates – 6 parameters affine transf. → 2 invariants

affine *invariant coordinates* (x_A, y_A) :

$$x_A = \frac{\begin{vmatrix} \bar{x} - \bar{x}_2 & \bar{x} - \bar{x}_3 \end{vmatrix}}{\begin{vmatrix} \bar{x}_1 - \bar{x}_3 & \bar{x}_2 - \bar{x}_3 \end{vmatrix}} \qquad y_A = \frac{\begin{vmatrix} \bar{x} - \bar{x}_3 & \bar{x} - \bar{x}_1 \end{vmatrix}}{\begin{vmatrix} \bar{x}_1 - \bar{x}_3 & \bar{x}_2 - \bar{x}_3 \end{vmatrix}}$$

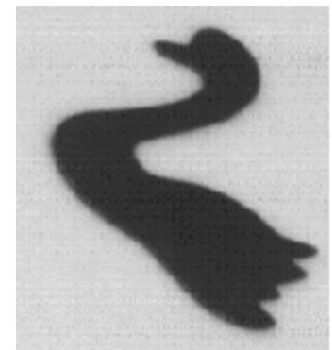


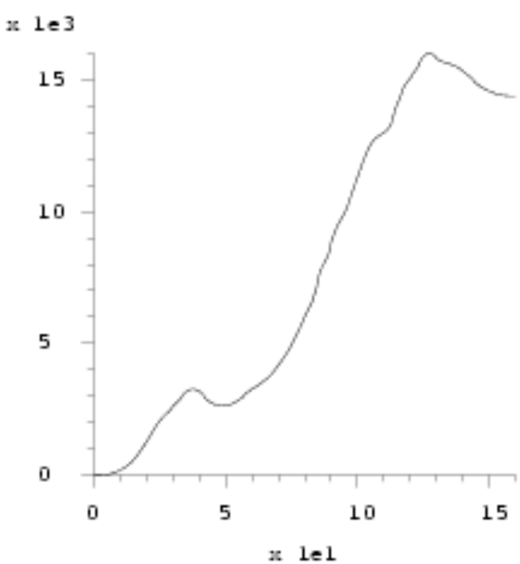
(Rel.) inv. under affine transf. : ex 2

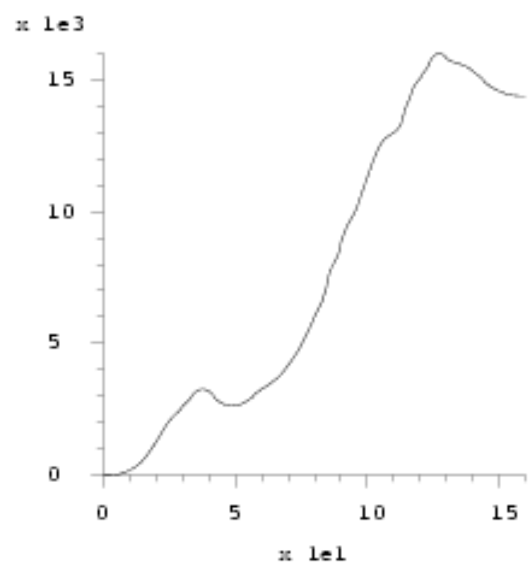
$$\int_{\text{start pt}} \left| \bar{x} - \bar{x}_1 \quad \bar{x}^{(1)} \right| dt$$

As a function of

$$\int_{\text{start pt}} abs \left(\left| \bar{x}^{(1)} \quad \bar{x}^{(2)} \right|^{\frac{1}{3}} \right) dt$$







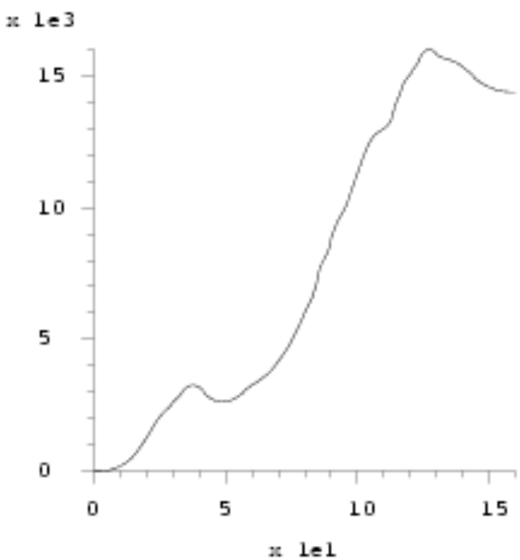
Computer Vision

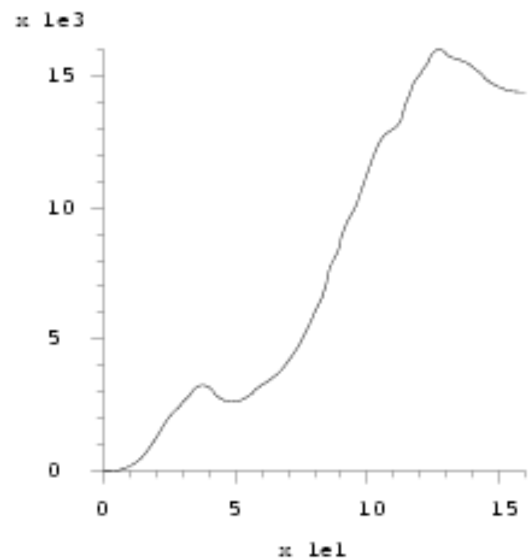
(Rel.) inv. under affine transf. : ex 2

$$\int_{\text{start pt}} \left|\bar{x}-\bar{x}_1 \quad \bar{x}^{(1)}\right| dt$$ as a function of $$\int_{\text{start pt}} abs\left(\left|\bar{x}^{(1)} \quad \bar{x}^{(2)}\right|^{1}\right) dt$$

$$\left(\frac{dx}{dt}, \frac{dy}{dt}\right)$$

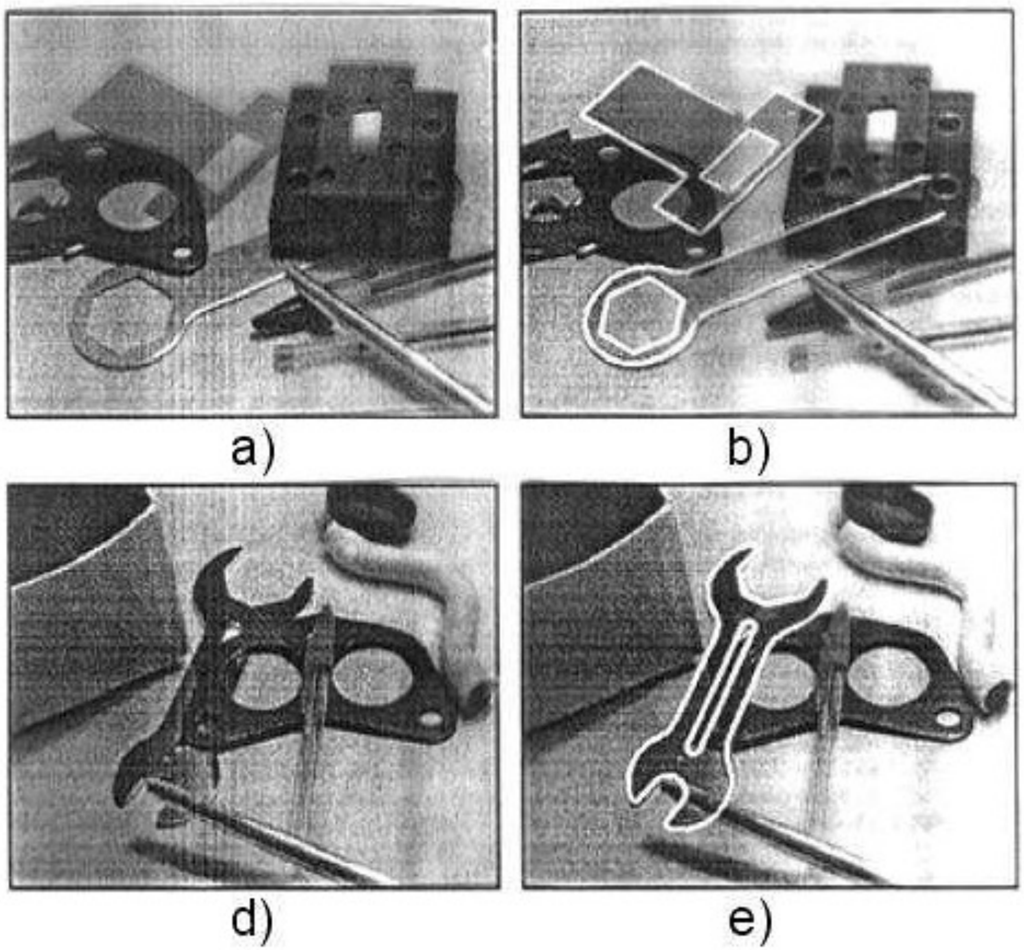
$$\left(\frac{d^2x}{dt^2}, \frac{d^2y}{dt^2}\right)$$





Computer Vision

Early attempts…1992



Projective invariance, Rothwell, 1992

Computer Vision

Image-based

Image-based

Appearance based methods

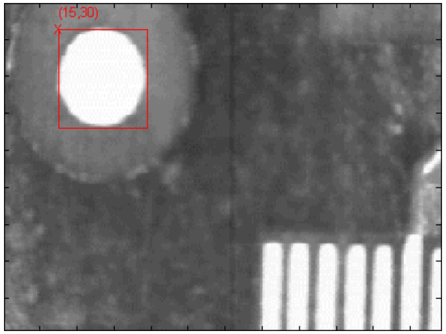
The model of an object is simply its image(s).

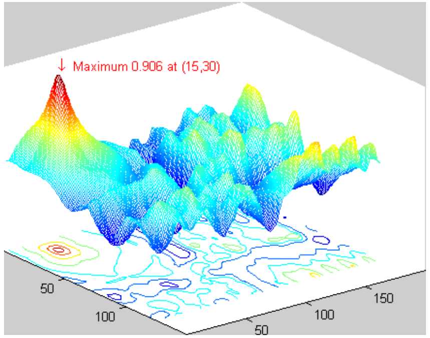
A simple example: Template matching

Shift the template over the image and compare

(e.g. Normalized Cross-Correlation or Sum of Squared Diff.)

Template





The problem is variation in the appearance because of changes in viewpoint / lighting

Zillions of templates!

The power of Principal Component Anal.

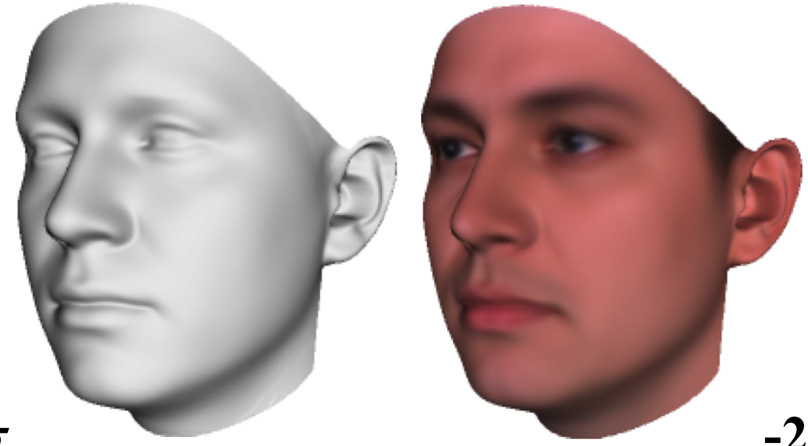
You remember PCA?

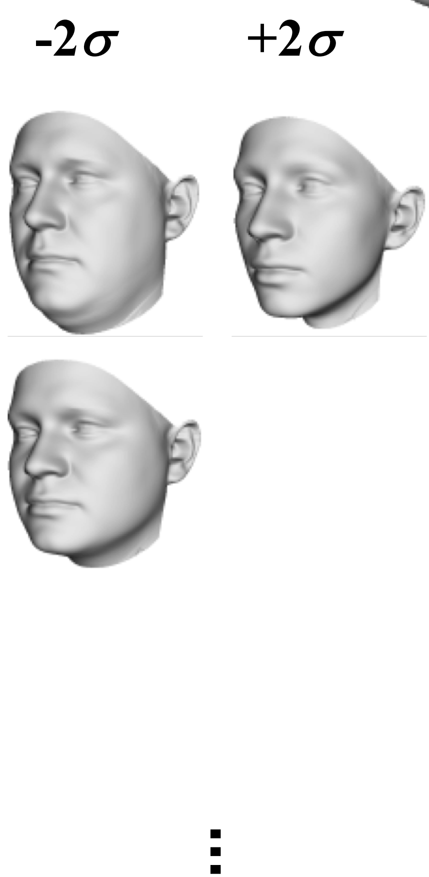
(… or the Karhunen-Loeve transform ?)

PCA represents data in a lower-dimensional space keeping most of the variance

It was seen to be powerful for similar patterns like faces, that exhibit a lot of redundancy

Eigenfaces for compact face representation





$$\mathbf{S} = \overline{\mathbf{S}} + \sum_{j=1}^{m} \alpha_j \mathbf{S}_j$$

$$\mathbf{T} = \overline{\mathbf{T}} + \sum_{j=1}^{m} \beta_j \mathbf{T}_j$$

Eigenfaces for compact face representation

Modes

Morphs

Computer Vision

Eigenfaces for compact face representation

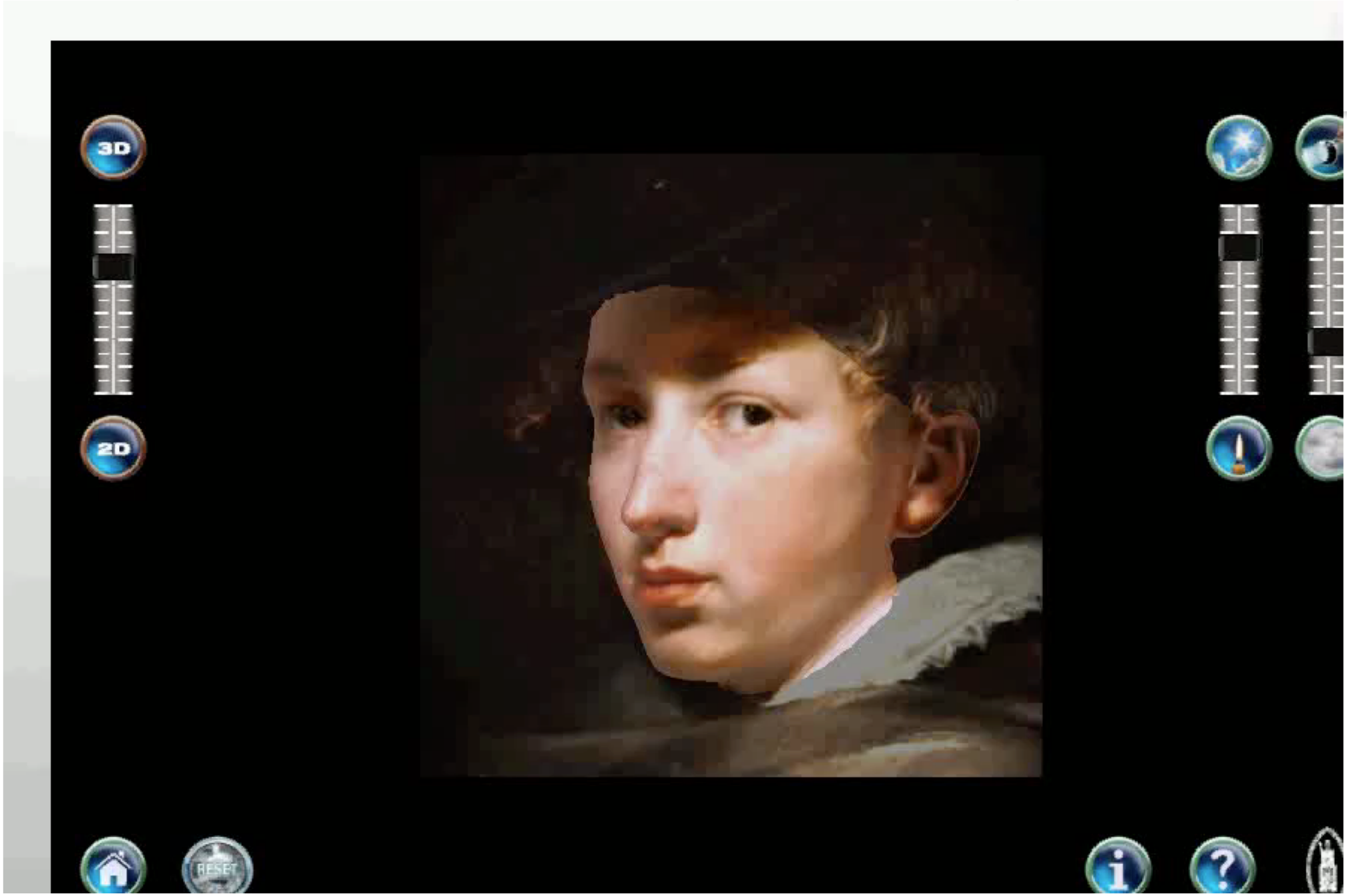
(self?-) portrait of the

young

Anthony Van Dijck

Eigenfaces for compact face representation

3D PCA-based reconstruction



Appearance manifold approach

(Nayar et al. '96)

Appearance manifold approach

Training

for every object :

- sample the set of viewing conditions
  (mainly viewpoints in this ex.)
- use these images as feature vectors
  (after manual bounding-box fitting around the object, rescaling, brightness normalization)

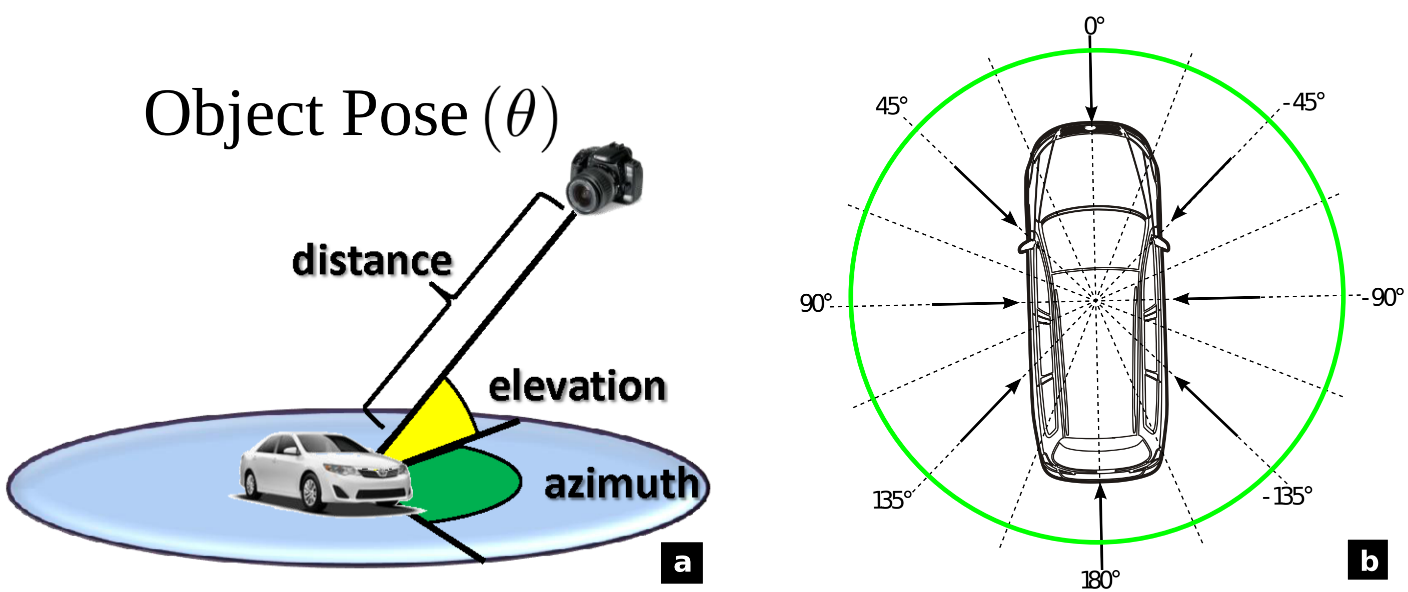
over all objects:

- apply a PCA over all the images of all objects
  (directly on the images)
- keep the dominant PCs
  (10-20 enough already)
- sequence of views for 1 object represent a manifold in the space of projections
  (fit splines to manifolds + resample if desired)

Appearance manifold approach

The objects were put on a turntable, and imaged from a fixed distance and under a fixed elevation angle; also the illumination remained fixed

hence the manifolds of appearances are simplified to a 1D, closed curve, but only considering the elevation angle will normally not suffice…

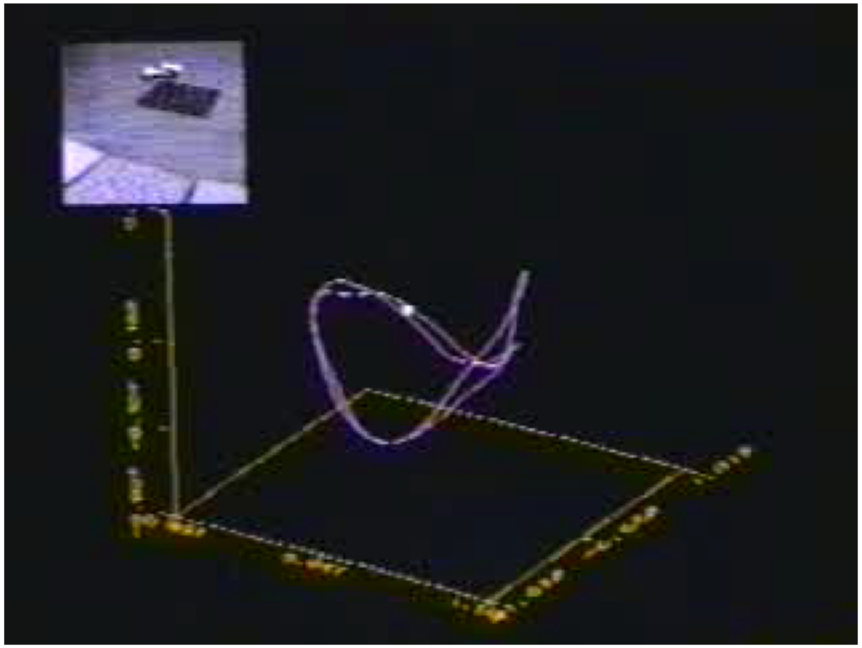


Computer Vision

Appearance manifold approach

For the illustration below, the images are shown in only a 3D space, as only 3 PCs are used in this case – for reasons of visualization

Sufficient characterization for recognition and pose estimation



Appearance manifold approach

Recognition stage (aka `Testing')

Represent the incoming image as a point in the same PC space

Type: what is the nearest manifold to the point ?

Pose: what is the closest point on that closest manifold ?

Real-time system (Nayar et al. ‘96)

Comparison between model-based and appearance-based techniques

Pure model-based	Pure appearance-based
Compact model	Large models
Can deal with clutter	Cannot deal with clutter
Slow analysis-by-synthesis	Efficient
Models difficult to produce	Models easy to produce
For limited object classes	For wide classes of objects

Computer Vision

Hybrid techniques

Euclidean invariant feature

(Schmid and Mohr ‘97)

Training

- look for corners
  (with the Harris corner detector)

- take circular regions around these points,
  of multiple radii
  (cope a bit with scale changes)

- calculate from the intensities in the circular regions
. invariants under planar rotation -> feature vectors

- do this from different viewpoints, where the
  invariance cuts down on the number of views needed
. (here no in-plane rotations necessary)

- put for every object and for each of its viewpoints the
. list of corner positions and their invariant feature
. vectors (descriptors) in a database

Euclidean invariant features

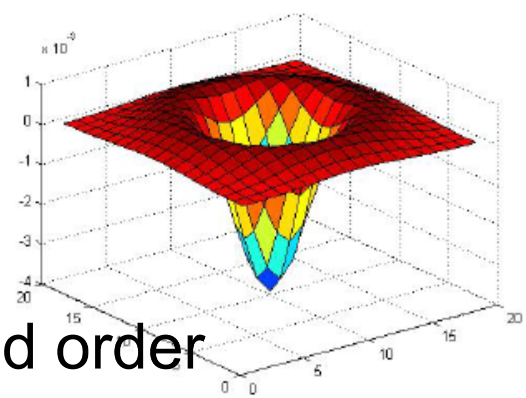
Example (rotation) invariant gradient:

$$G_x G_x + G_y G_y$$

Where G_x and G_y represent horizontal and vertical derivatives of intensity weighted by a Gaussian profile *(`Gaussian derivatives')*

2nd example invariant:

$$G_{xx} + G_{yy}$$



Where G_{xx} and G_{yy} represent 2nd order Gaussian derivatives

Note 1: several other invariants measured, then all put in a vector
Note 2: compute features for circles at different scales, (i.e. take scale into account explicitly) and each scale gets its own vector

Euclidean invariant feature

(Schmid and Mohr ‘97)

Testing

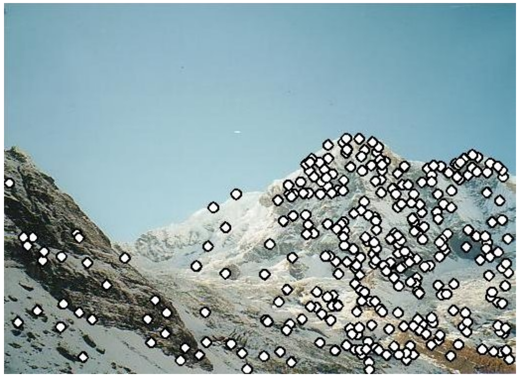
- extract corners and their invariant descriptors from the incoming image
- compare these invariants with those stored in the database -> find matches
- look for consistent placement of candidate matching corner points (e.g. using epipolar geometry)
- decide which object based on the number of remaining matches (i.e. consistently placed matches)
  (the best matching image yields the object type+appr.pose)

Local features: main components

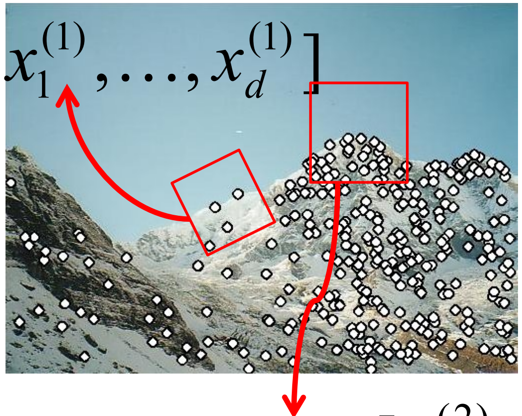
1) Detection: Identify interest points

2) Description: Extract feature vector descriptors around them

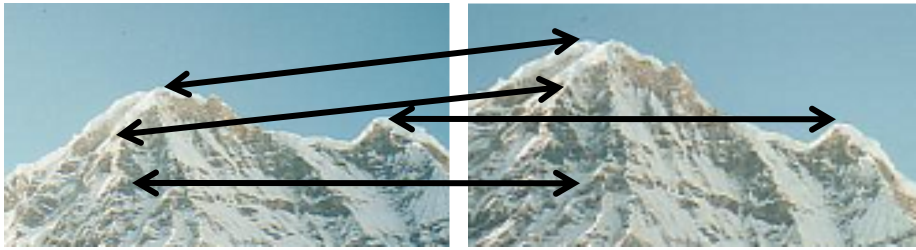
3) Matching: Determine correspondence between descriptors in two views



$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$



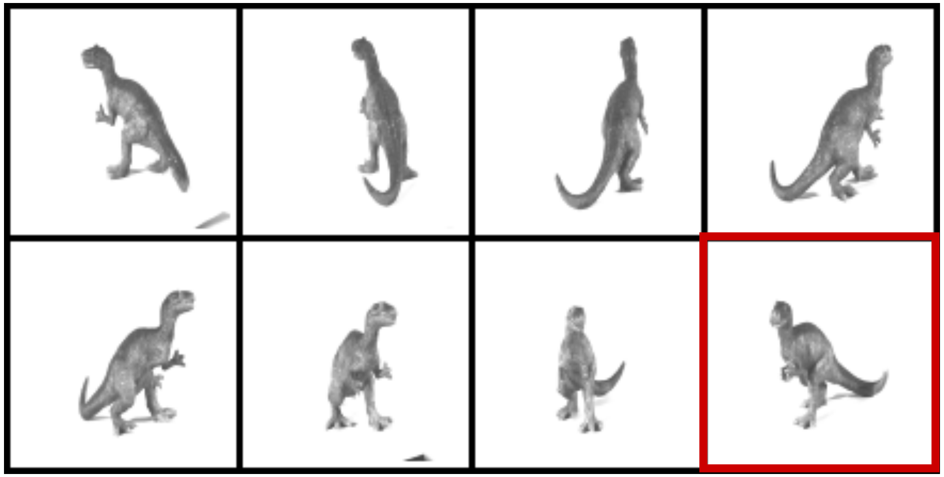
$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$



Kristen Grauman

Example

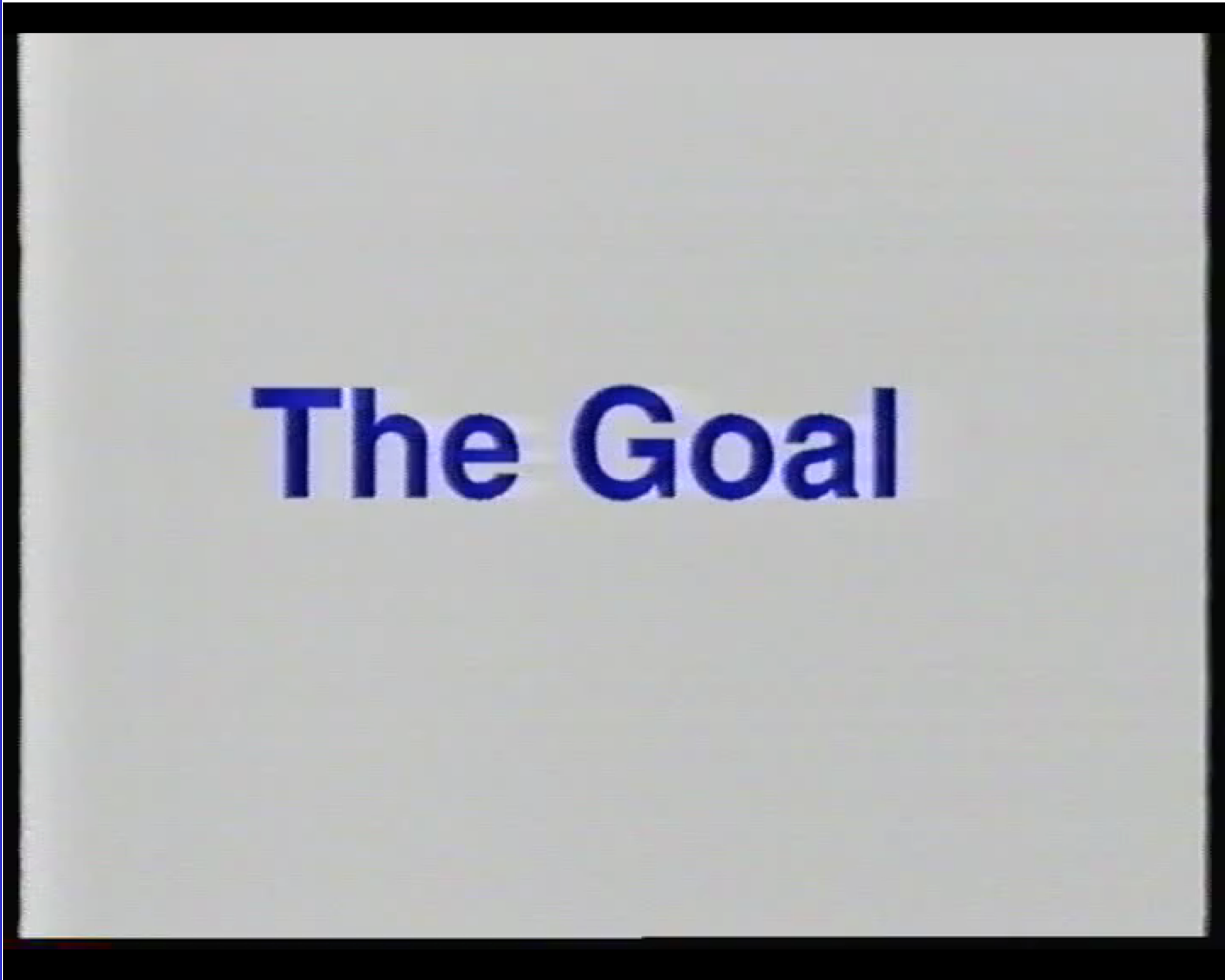
Training examples for one object in the database



Test image

+ deal with cluttered background
+ need less training images
~ problems with uniform objects

Example



Hybrid techniques

+ Rather compact model

+ Can deal with clutter and partial occlusion

+ Efficient

+ Models easy to produce

(take images, fewer than in pure appearance-based method)

+ For rather wide class of objects

(almost as wide as in pure appearance based,
but there is a problem with untextured objects)

Hybrid techniques

The idea of using these local interest points, with their surroundings characterized by a vector of features (`descriptor'), became very popular after Schmid introduced her method.

The invariance of Schmid's point descriptors was still quite limited though. Increasing the level of invariance (larger groups of transformations under which the descriptor remains unchanged) would further reduce the number of images that need to be taken as reference images (fewer viewpoints, for instance)

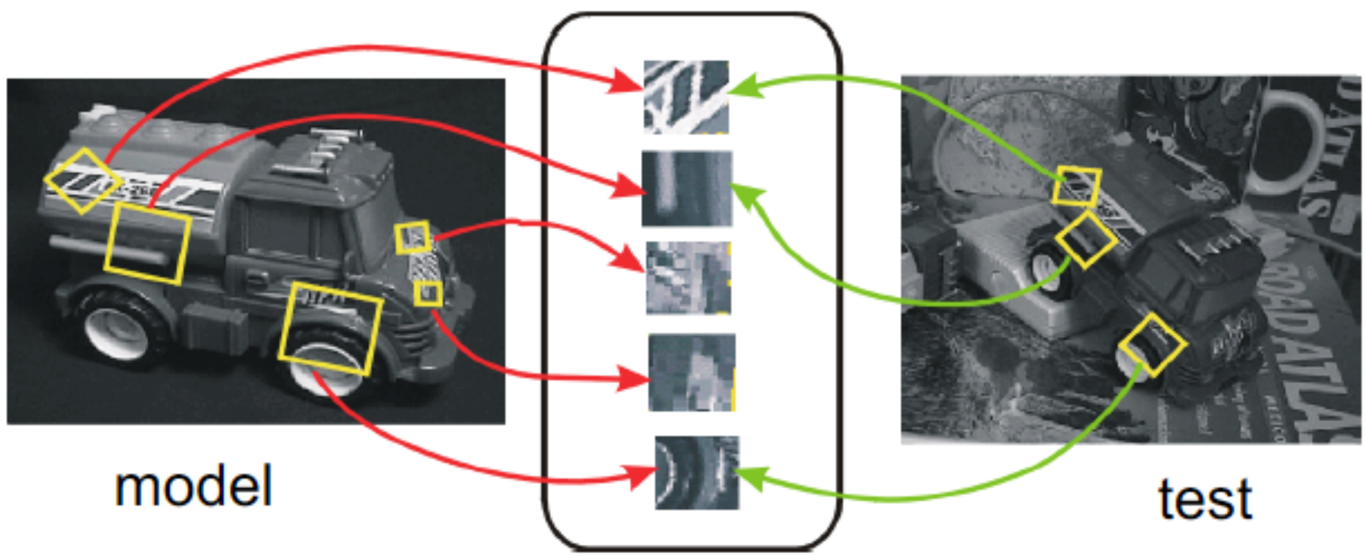
The descriptors could also be made invariant under changes of illumination, for instance…

Next we consider affine + photometric invariance

Computer Vision

Matching with local features, what follows

- Detect and describe features in model image(s) (done: scale invariant features; next: affine invariant ones)
- Detect and describe features for test image
- Match features (including geometric verification, e.g. RANSAC)
- Count matches
  → many? → object recognized and localized
  → few? → object not present in test image



Recognition using local affine and photometric invariant features

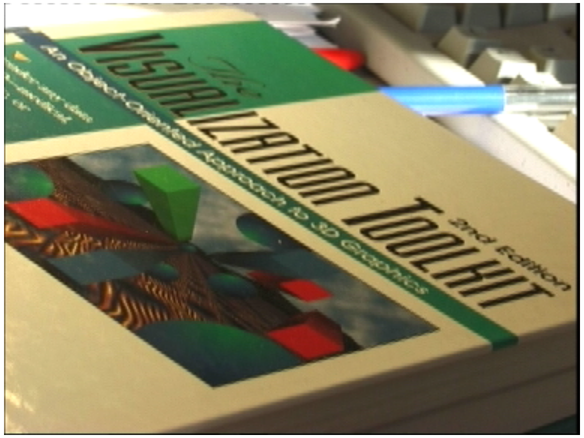
Hybrid approach that aims to deal with
large variations in

Viewpoint

Recognition using local affine and photometric invariant features

Hybrid approach that aims to deal with **large variations** in

- Viewpoint
- Illumination



Recognition using local affine and photometric invariant features

Hybrid approach that aims to deal with
large variations in

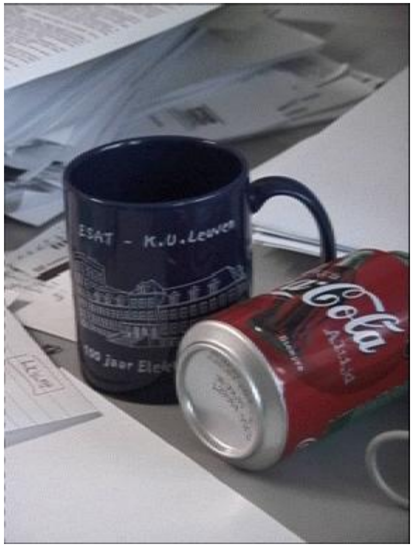
- Viewpoint
- Illumination
- Background

Computer Vision

Recognition using local affine and photometric invariant features

Hybrid approach that aims to deal with **large variations** in

- Viewpoint
- Illumination
- Background
- and Occlusions



Recognition using local affine and photometric invariant features

Hybrid approach that aims to deal with **large variations** in

- Viewpoint
- Illumination
- Background
- and Occlusions

⇒ Use local invariant features

Invariant features
= features that are preserved under a specific group of transformations

Recognition using local affine and photometric invariant features

Hybrid approach that aims to deal with **large variations** in

Viewpoint

Illumination

Background

and Occlusions

⇒ Use local invariant features

Robust to changes in viewpoint and illumination

Recognition using local affine and photometric invariant features

Hybrid approach that aims to deal with **large variations** in

Viewpoint

Illumination

Background

and Occlusions

⇒ Use local invariant features

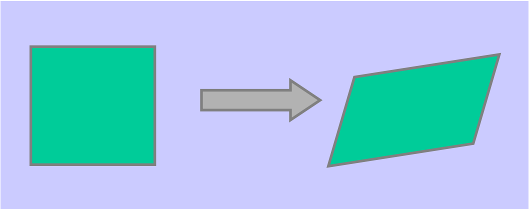
Robust to changes in viewpoint and illumination

Robust to occlusions and changes in background

Transformations for planar objects

Affine geometric deformations

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix}$$



Linear photometric changes

$$\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = \begin{bmatrix} s_R & 0 & 0 \\ 0 & s_G & 0 \\ 0 & 0 & s_B \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} o_R \\ o_G \\ o_B \end{bmatrix}$$

Local features: desired properties

Repeatability

The same feature can be found in several images despite geometric and photometric transformations

Distinctiveness

Each feature has a distinctive descriptor

Thus, we can go further with invariance than similarities (as in the current example of affine + photometric), to increase repeatability, but we risk to reduce distinctiveness doing so

Local invariant features

We glossed over another important issue…

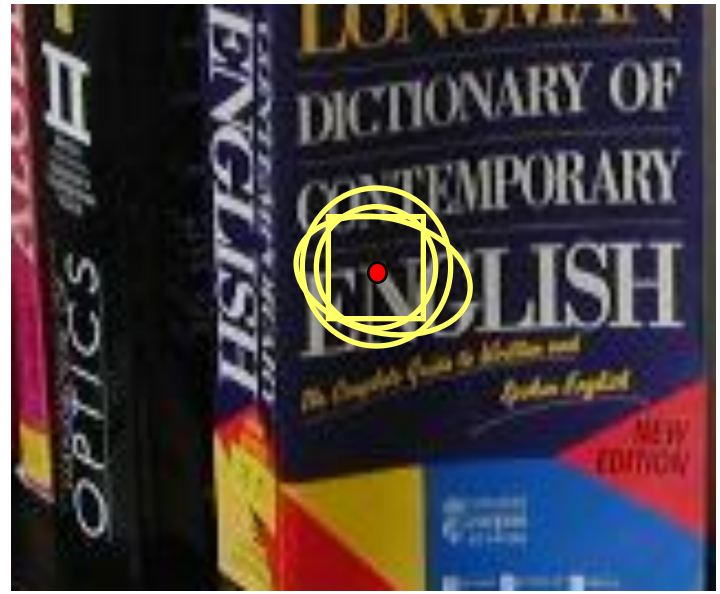
Interest points -> neighborhood -> descriptor

The neighborhood should cover the same part of the scene in the given image and the reference image that we want to match against… but changing viewpoint then also changes neighbourhood shape

In Schmid’s method a circle was OK, because only invariance under in-plane rotation was considered

But how about affine invariance? Eg a circle would turn into an ellipse under a general affine change

Local invariant features



… e.g. by going for invariance under affinities rather than similarity

Computer Vision

The need for variable patch shape

The important thing is to achieve such change in patch shape without having to compare the images, i.e. this should happen on the basis of information in one image only !

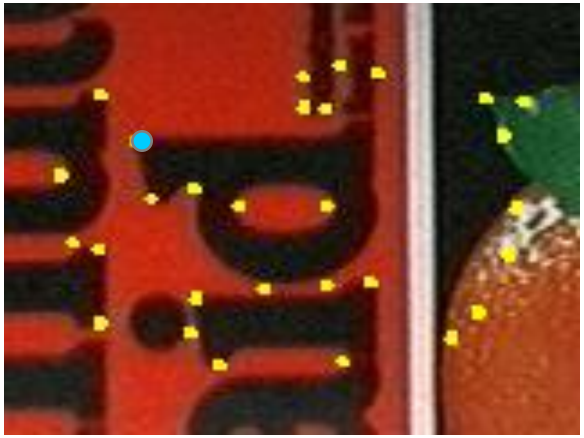
As in this ex: if the circle would be selected as neighbourood for the image on the left, the ellipse should be selected for the image on the right, without any knowledge of the image on the left

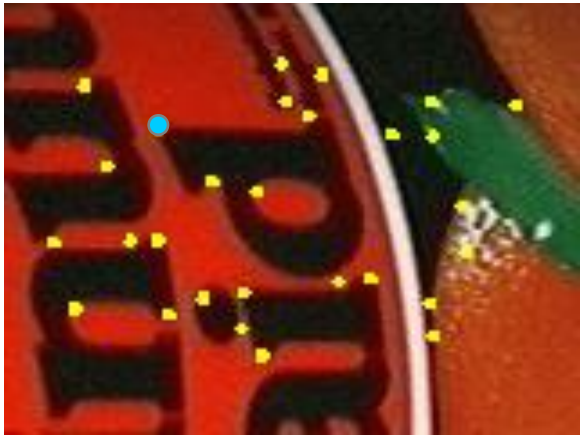
Computer Vision

Example: starting from edge corners

Example: starting from edge corners

1. Harris corner detection





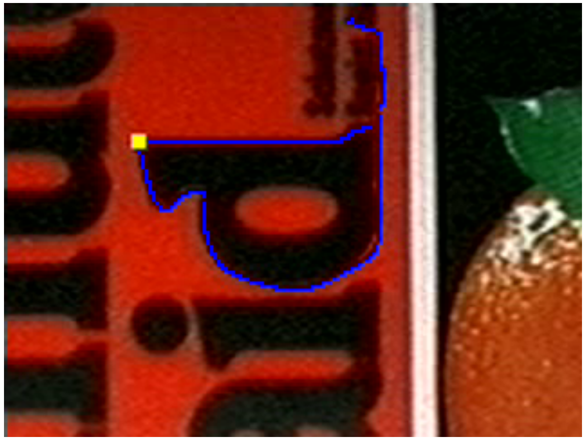
Example: starting from edge corners

2. Canny edge detection

Computer
Vision

Example: starting from edge corners

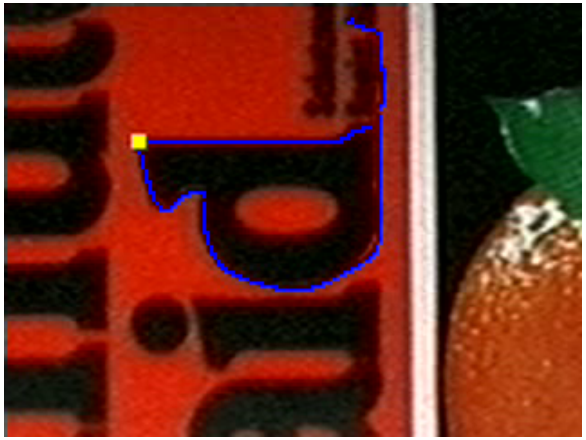
3. Evaluation relative affine invariant parameter along two edges



Moving away from the corner, consider point pairs that yield equal areas between the curve and the straight joint between the pts -> 1D family of pairs

Example: starting from edge corners

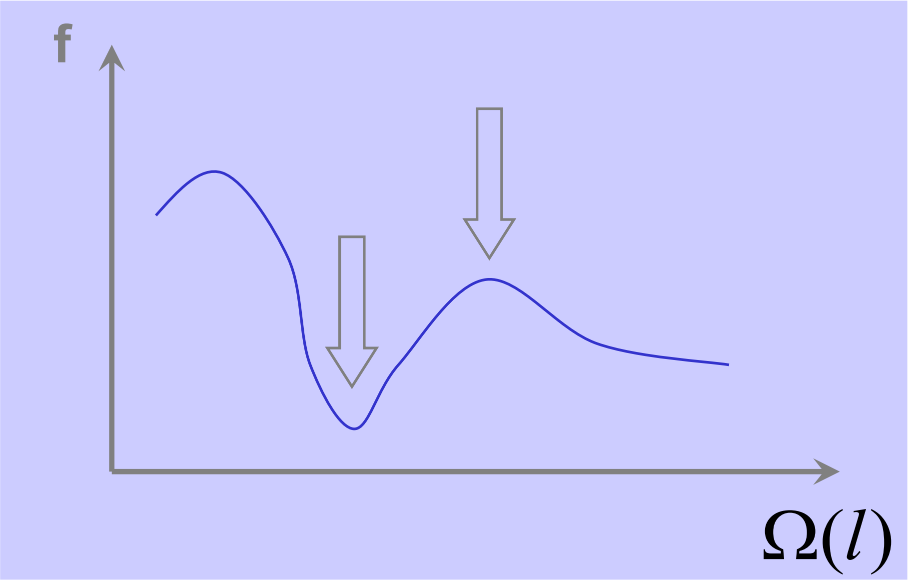
4. Construct 1-dimensional family of parallelogram shaped regions



Example: starting from edge corners

5. Select parallelograms based on invariant extrema of function

For instance: extrema of average value of a color band within the patch



Example: starting from edge corners

5. Select parallelograms based on local extrema of invariant function

Increasing the level of invariance: `Invariant Neighbourhoods' are needed

note regions are extracted based on local info only

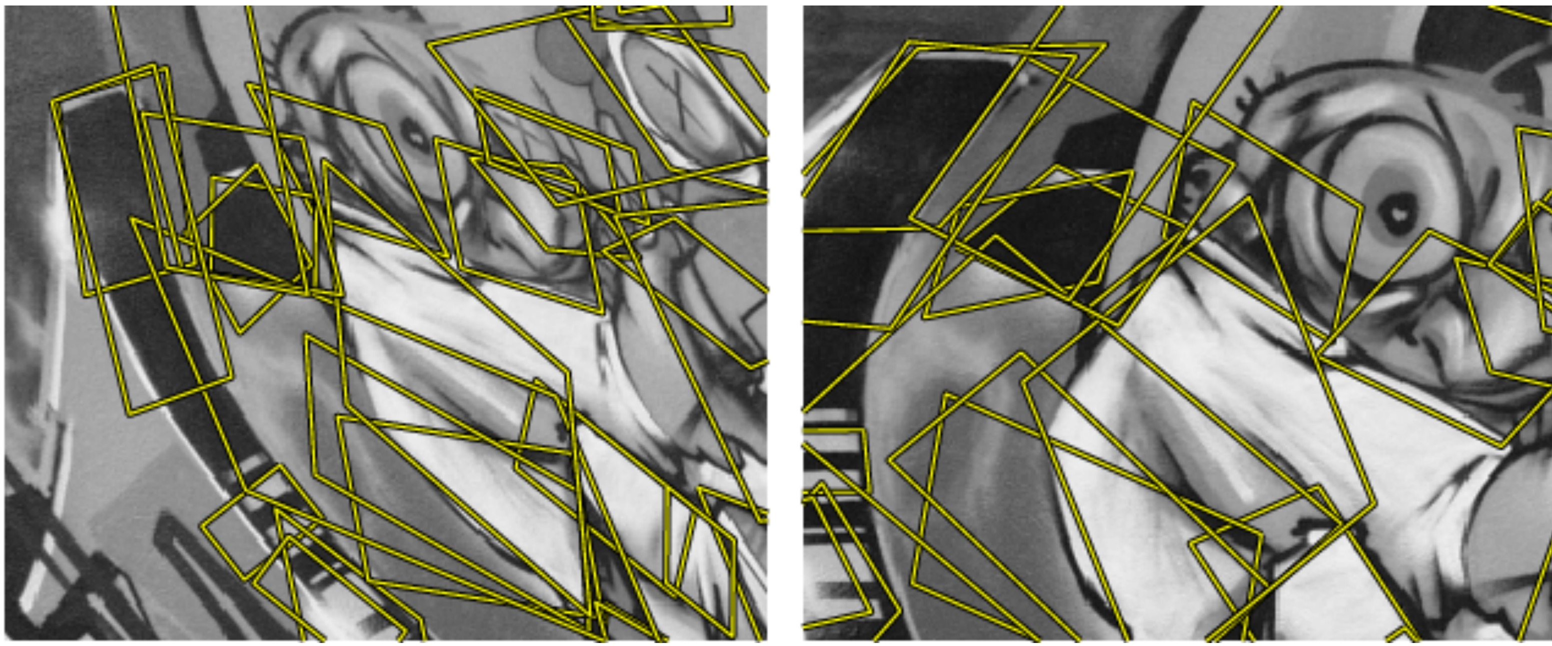
This method started from corners on edge strings

The need for variable patch shape

Another example

Note the global perspective/projective distortion, dealt with rather well with the local affine patches that we use !

Example 1: edge corners + affine moments



Other approach yielding invariant neighbourhoods (around intensity extrema)

Local invariant features

Once we have such affinely invariant neighbourhoods, we again characterize them by extracting descriptors from them – e.g. affine - photometric invariant ones – that we match

Next we show results for a specific object recognition system that uses affine invariant regions

Some extra tricks are used to increase the success of affine region matching, that we do not discuss here (Ferrari, Tuytelaars, and Van Gool, 2006)

As to the choice of affine-photometric invariants we refer to the literature…

Results: model objects (planar)

1 model view each

Results: model objects (curved)

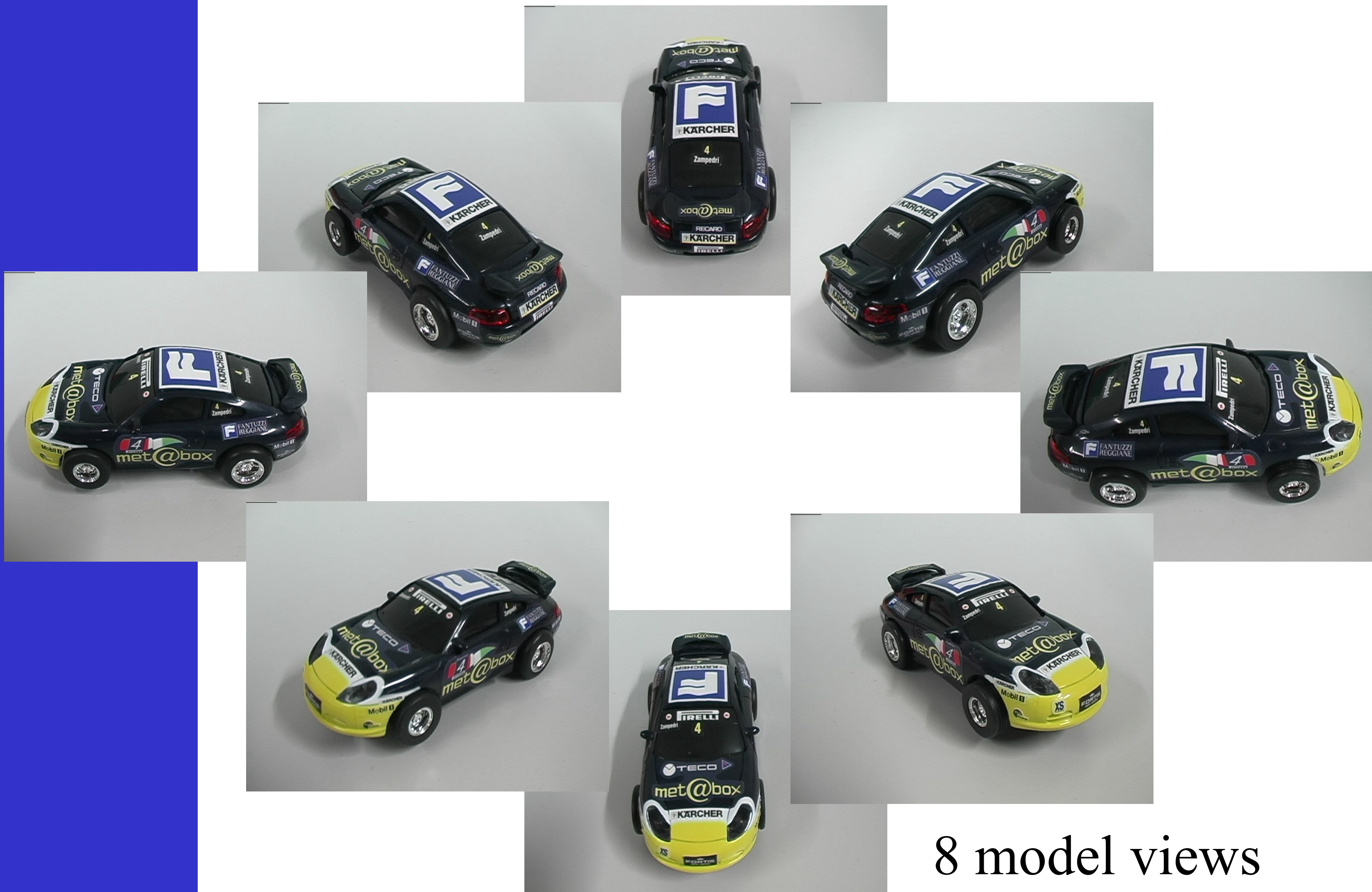
6 model views

Computer Vision

Results: model objects (curved)

6 model views

Results: model objects (3D)



8 model views

Computer Vision

Results

Computer Vision

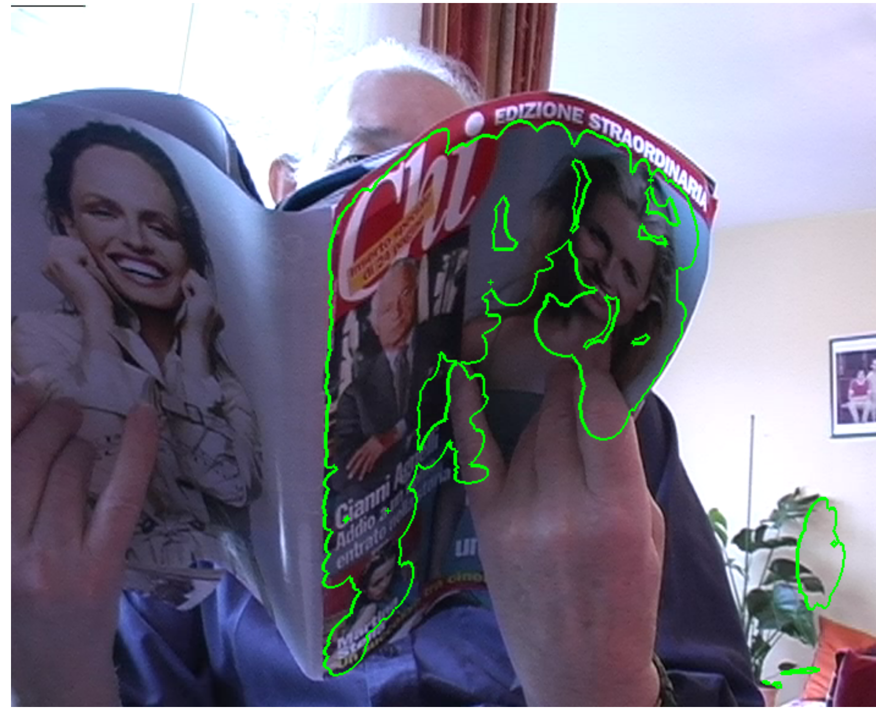
Results

Computer Vision

Results

Large scale change, heavy occlusion

Results



Deformation, illumination change, occlusion

Results

Large scale change, perspective deformation, extensive clutter

Results

Extensive clutter, scale, occlusion, blur

Results

Extensive clutter, scale, occlusion, blur

Computer Vision

Robustifying Hybrid techniques

Supporting the matching step

1) Too slow if naively done

2) Will often fail when only based on descriptor matching

Supporting the matching step

1) Too slow if naively done

2) Will often fail when only based on descriptor matching

Supporting the matching step

1) Hierarchical vocabulary tree for speed-up

Indexing local features

With potentially thousands of interest pts + their descriptors per image,
and hundreds to millions of images to search,
how to efficiently find those relevant to a new test image?

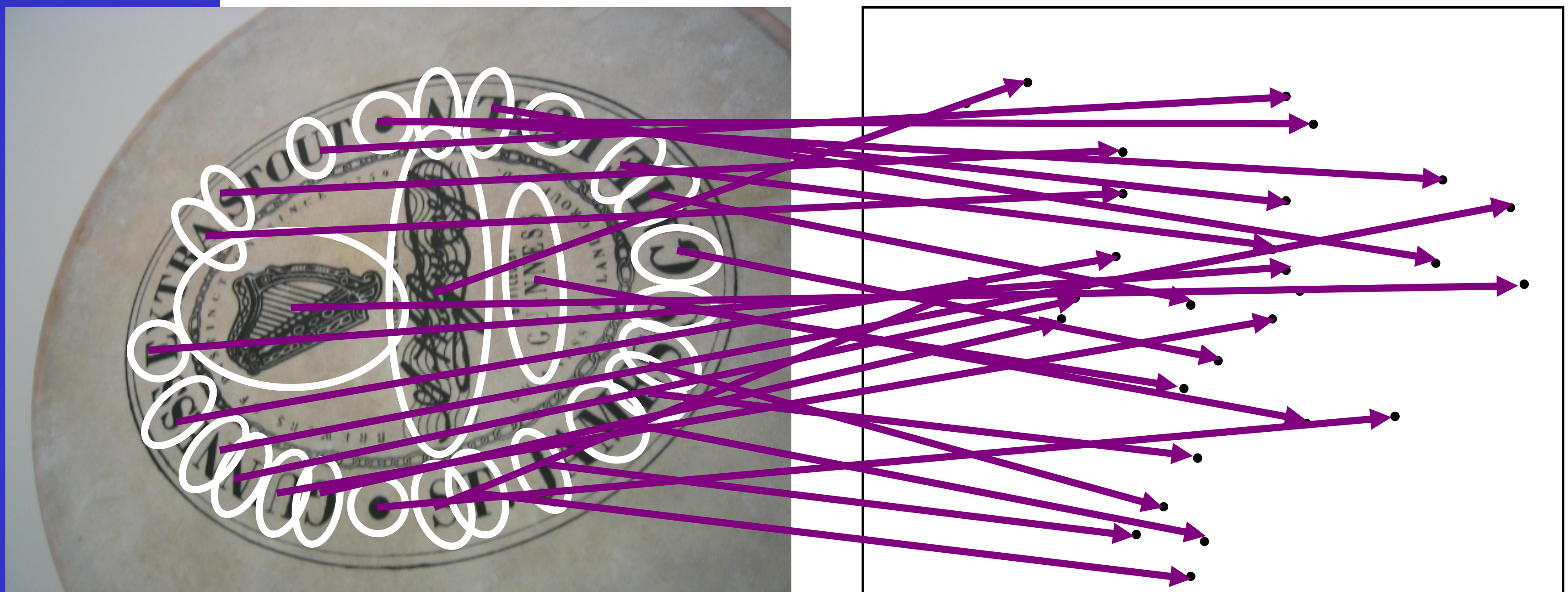
Quantize/cluster the descriptors into *`visual words'*

And match words hierarchically: *vocabulary tree*

Use *Inverted file indexing schemes*

Visual words: main idea

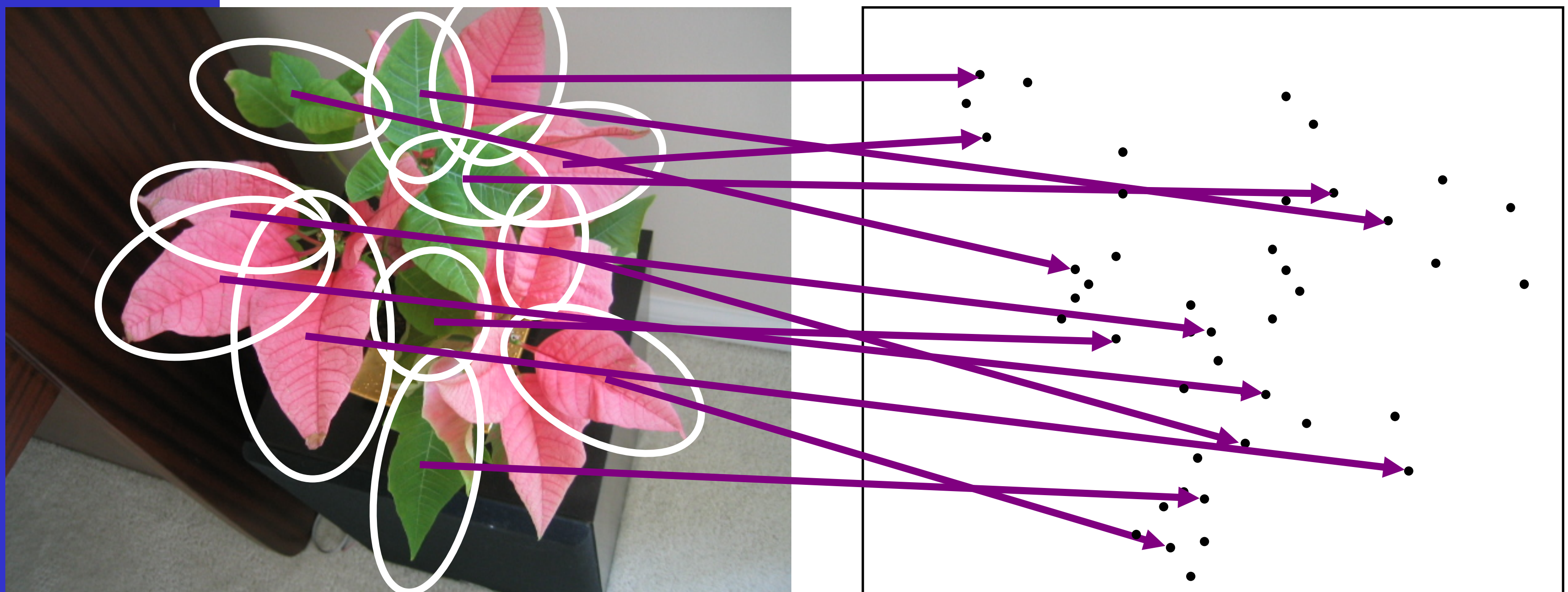
Extract some local features from a number of images



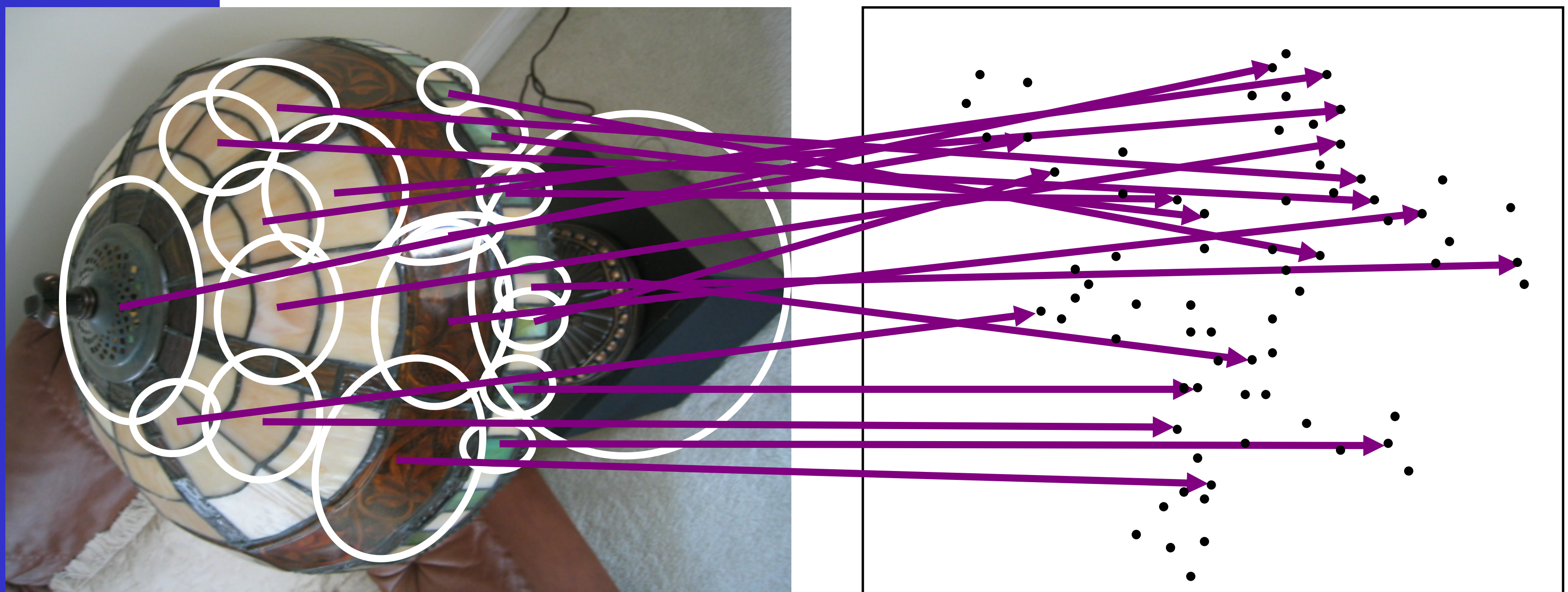
e.g., SIFT descriptor space: each point is 128-dimensional

Slide credit: D. Nister, CVPR 2006

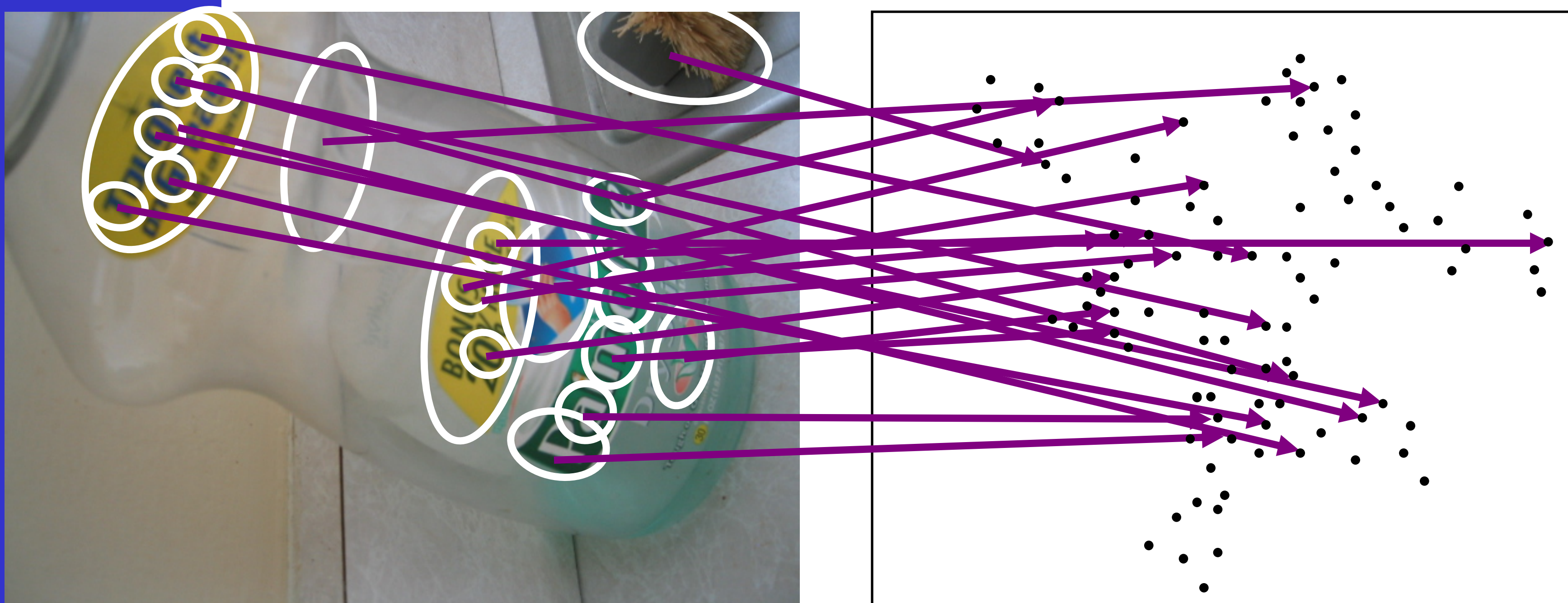
Visual words: main idea



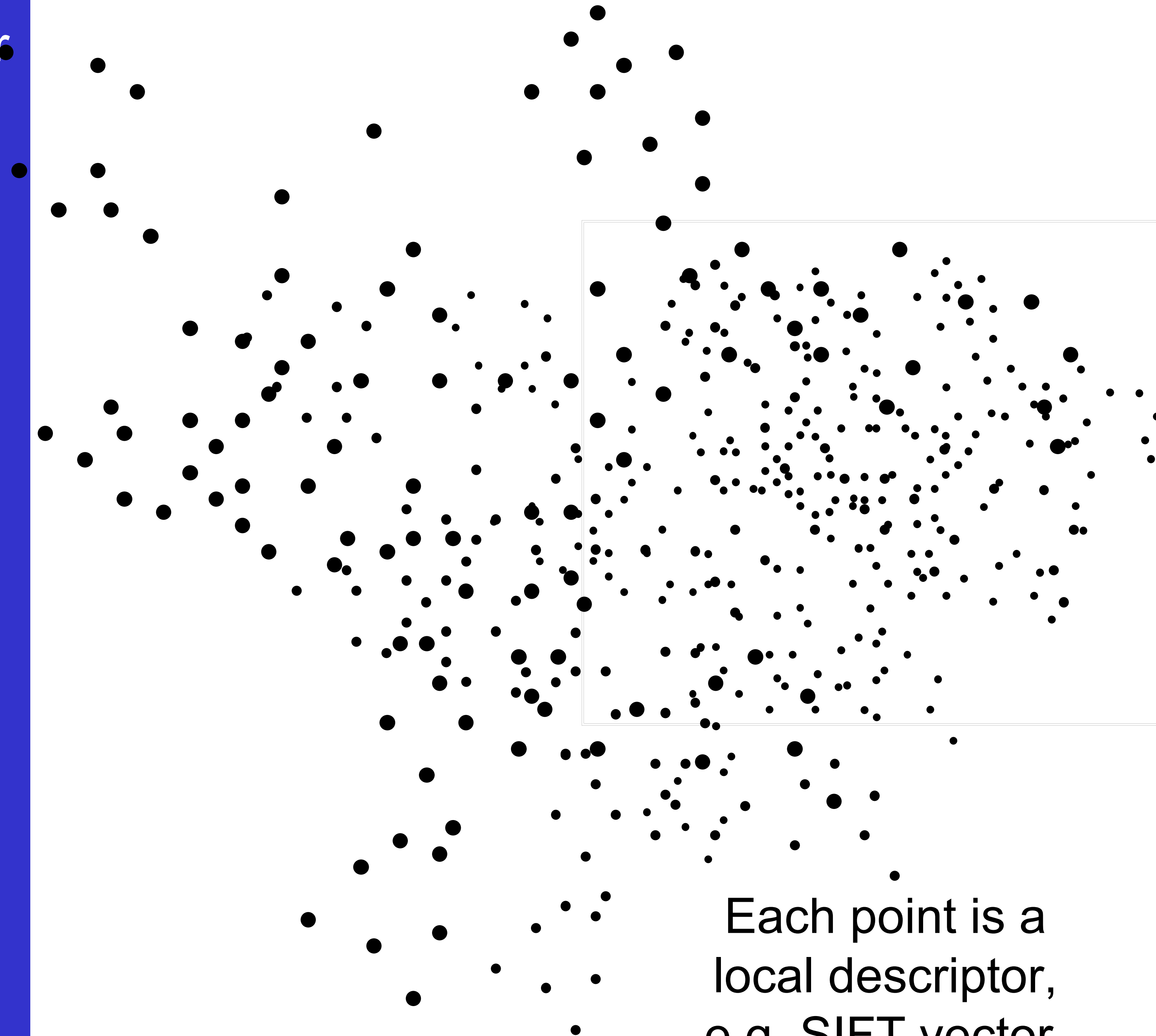
Visual words: main idea



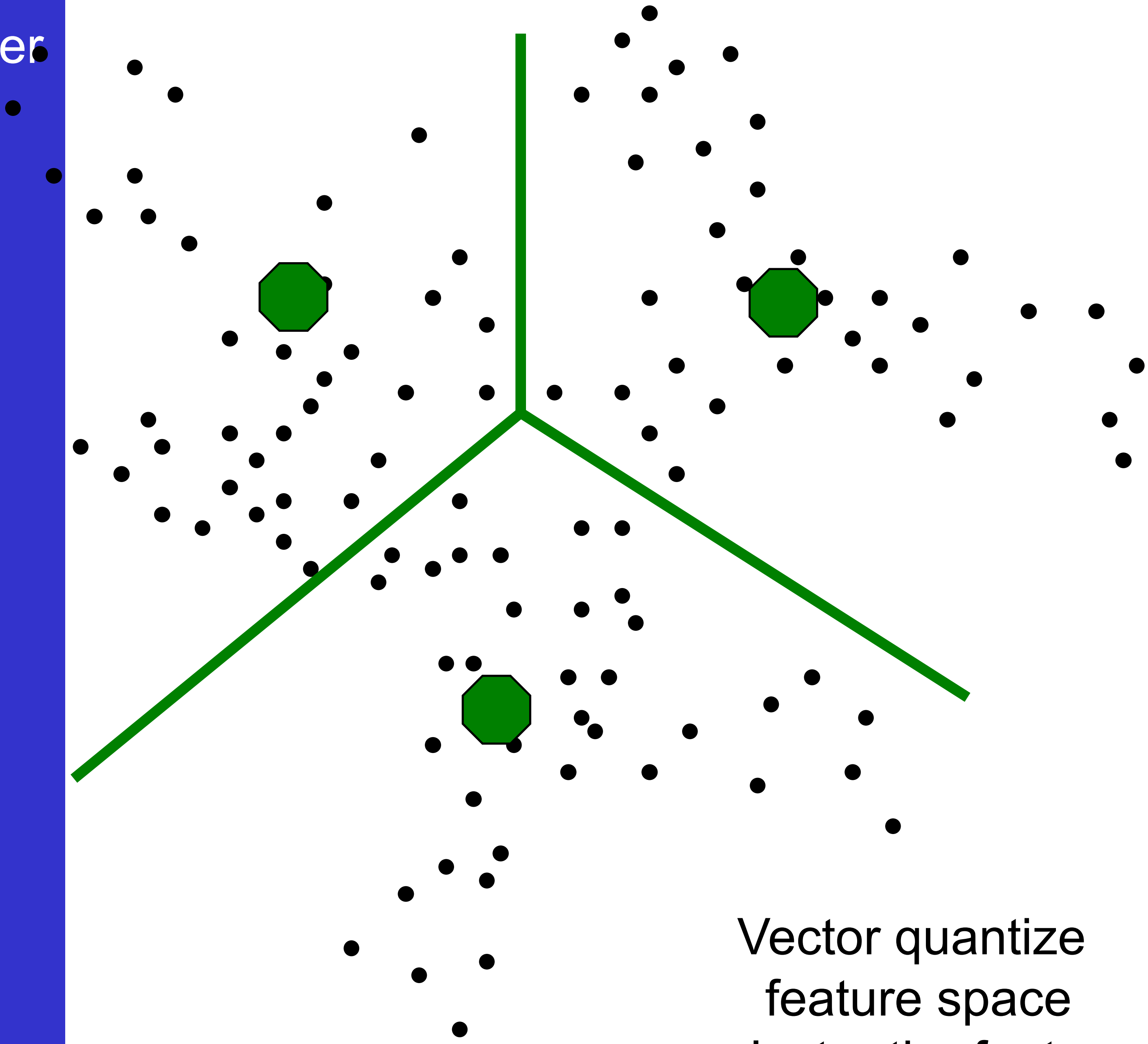
Visual words: main idea



Computer Vision



Each point is a local descriptor, e.g. SIFT vector.

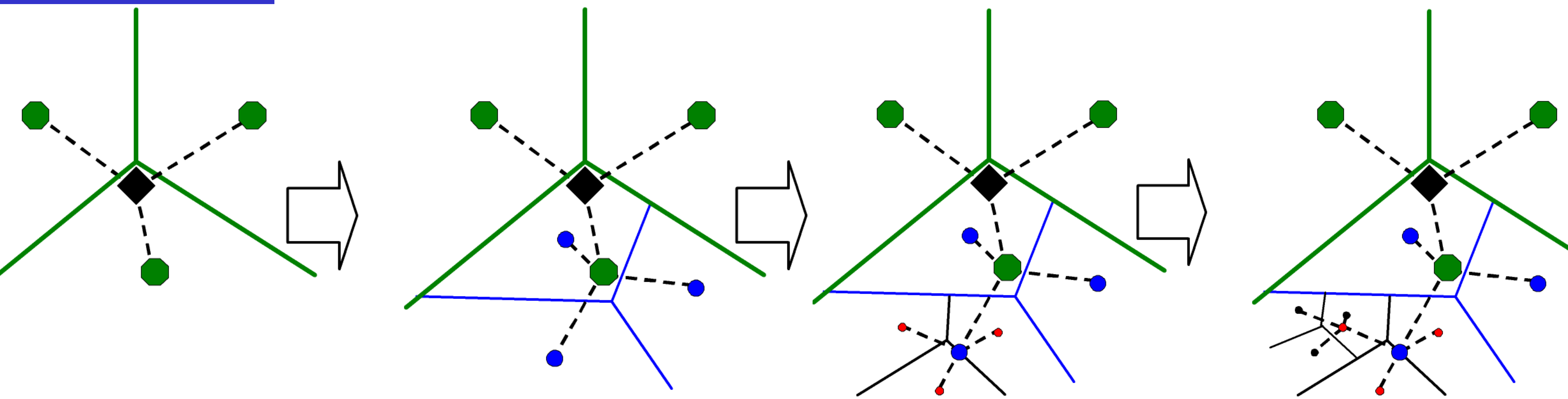


Vector quantize
feature space
= cluster the features

K-means clustering

1. randomly initialize K cluster centers
2. Assign each feature to nearest cluster center
3. Recompute cluster center (mean)
4. Iterate from 2, until convergence

Hierarchical K-means clustering

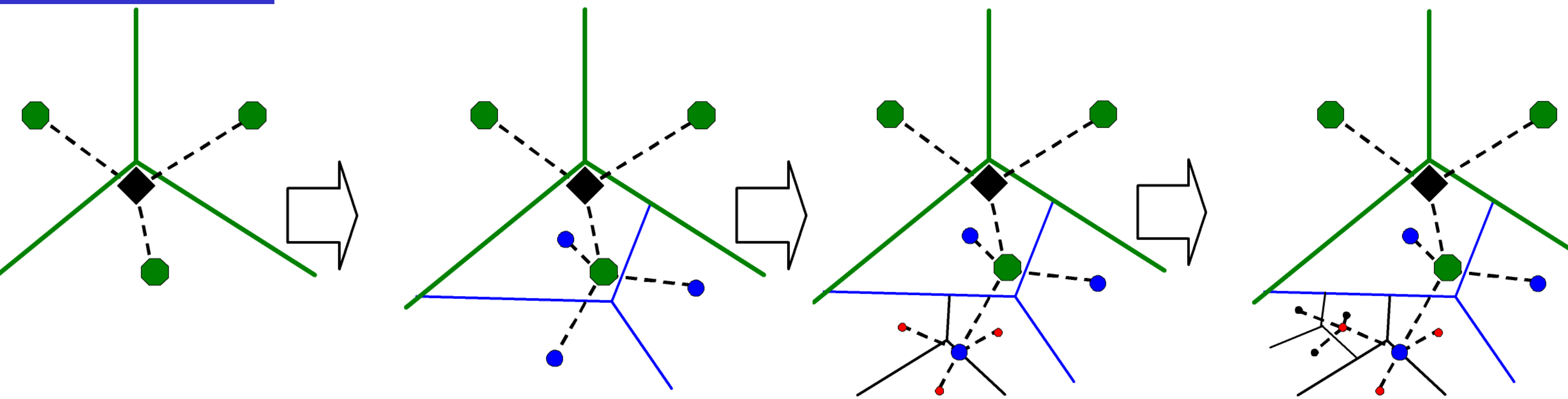


Allows to use larger vocabularies and thereby yields better results

In the example k=3, but typically it is chosen higher, e.g. k=10 and 6 layers could be used for search in about 1M images

Hierarchical K-means clustering

Here subdivisions for only one cluster at each layer… actually done for all



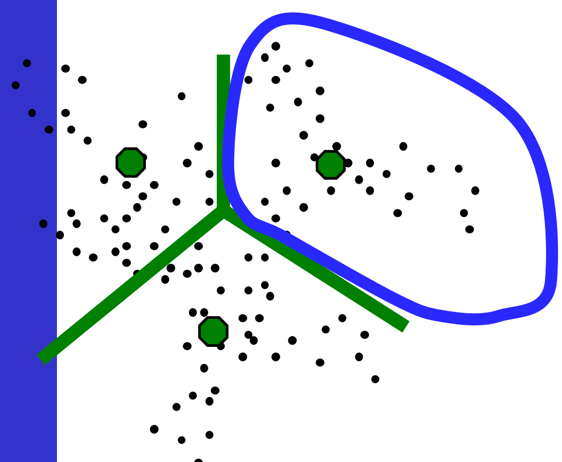
Allows to use larger vocabularies and thereby yields better results

In the example k=3, but typically it is chosen higher, e.g. k=10 and 6 layers could be used for search in about 1M images

Computer Vision

Visual words

Ex: each group of patches belongs to same visual word



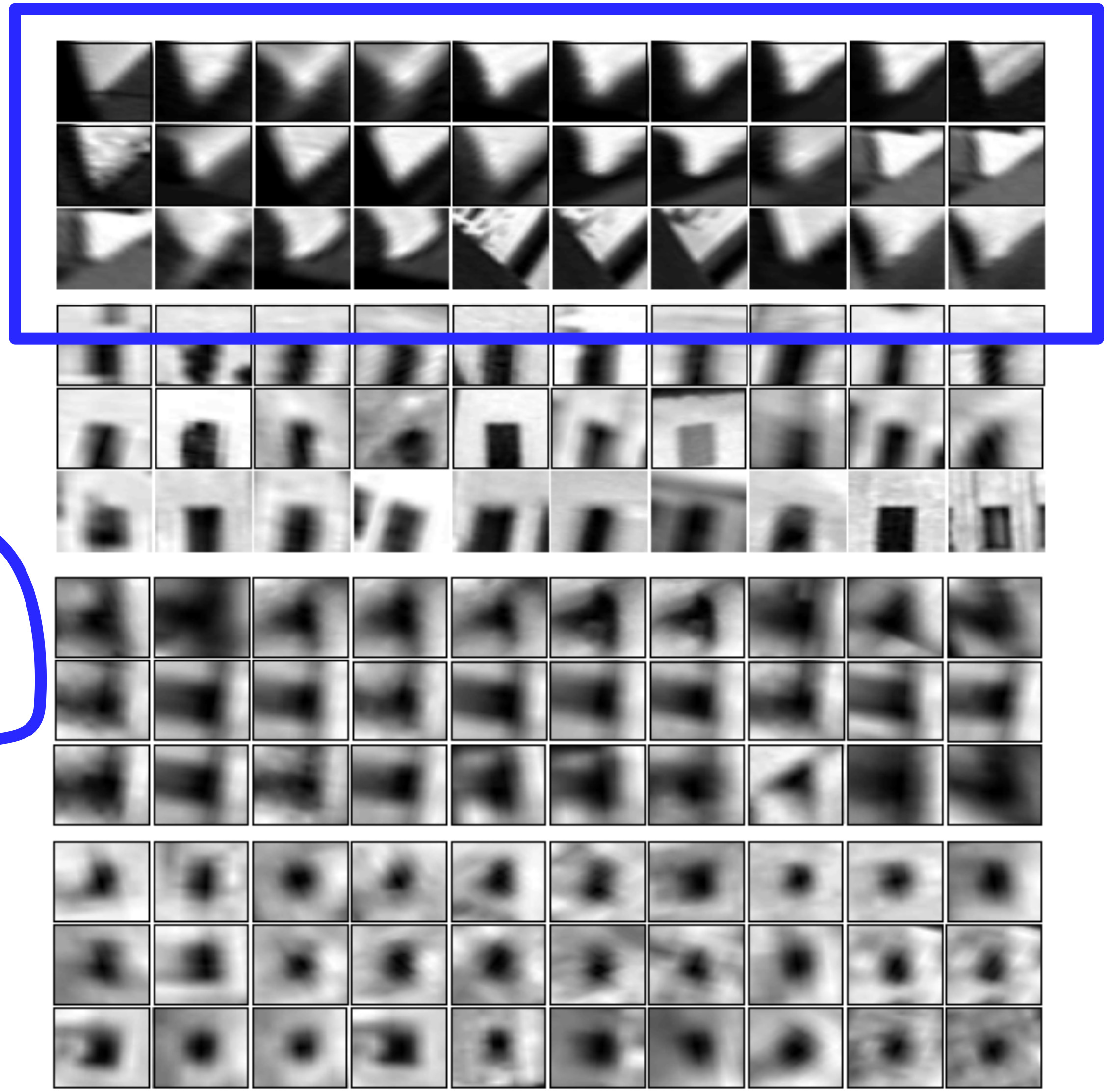


Figure from Sivic & Zisserman, ICCV 2003

Kristen Grauman

Indexing local features: inverted file index

Computer Vision

Index

For text documents, an efficient way to find all *pages* on which a *word* occurs is to use an index…

We want to find all *images* in which a *visual word* occurs.

Kristen Grauman

Inverted file index

Database images

Image #1

Image #2

Image #3

⋮

Word #	Image #
1	3
2 …	
7	1, 2
8	3
9	
10 …	
91	2
⋮	⋮

…

Database images are loaded into the index, mapping words to image numbers

Kristen Grauman

Inverted file index

New query image

Word #	Image #
1	3
2 …	
7	1, 2
8	3
9	
10 …	
91	2
⋮	⋮

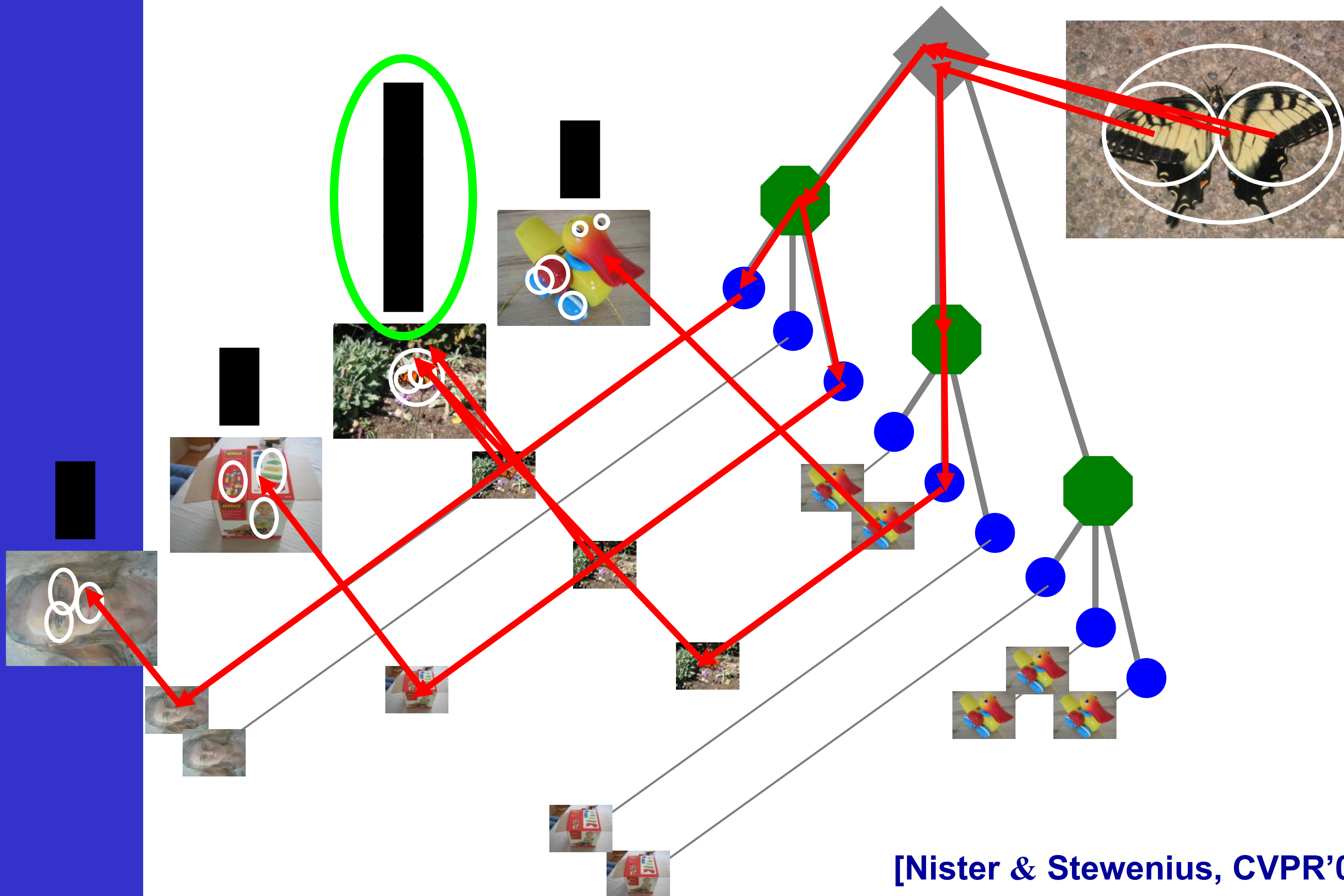
…

New query image is mapped to indices of database images that share a word.

Kristen Grauman

Computer Vision

Retrieval with vocabulary tree + inverted file index

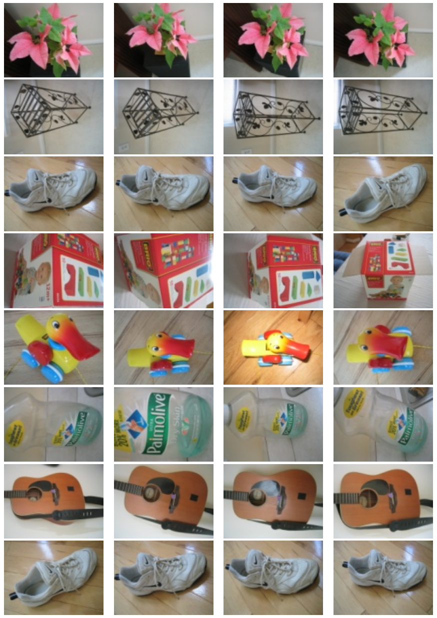


[Nister & Stewenius, CVPR'06]

Slide credit: David Nister

Computer
Vision

Performance



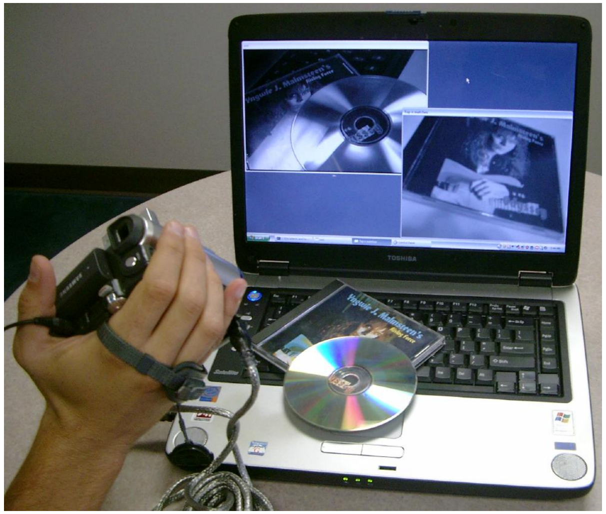
Evaluated on large databases

- Indexing with up to 1M images

Online recognition for database of 50,000 CD covers

- Retrieval in ~1s

Best with very large visual vocabularies



NOTE: object class recognition typically done with smaller vocabularies

Supporting the matching step

1) Too slow if naively done

2) Will often fail when only based on descriptor matching

RANSAC - intermezzo

Matching can start from interest points and their descriptors, but such matching is rather fragile.

Typically, several `matches' are wrong, so-called ***outliers***, and one needs to add a test on the configuration of the matches in order to remove the outliers and keep the correct ***inliers***.

Epipolar geometry and projective matching are often used tests, using RANSAC to withstand unavoidable mismatches.

We describe RANSAC <u>after the next slide</u>

RANSAC

The RANSAC test on **epipolar geometry** assumes that there is a fundamental matrix that matches are in agreement with, and

The RANSAC test on **projectivities** that there is a projectivity that maps points in the first image onto the matching points in the second

Such tests allow for the elimination of many outliers

but these tests make strong assumptions about the scene:

Epipolar geometry: rigidity of the scene (i.e. objects in the scene do not move with respect to each other)

Projectivity: the scene is not only rigid, but also (largely) planar

Nonetheless such tests help !

RANSAC

algorithm (full name RANdom SAmple Consensus) that assumes the data consists of "inliers", i.e. correct matches, and "outliers", i.e. incorrect matches.

From a set of match candidates, RANSAC

1. randomly select the minimal nmb of matches to formulate an initial test hypothesis (e.g. 7 for epipolar geometry or 4 for a projectivity; this nmb better be small since the selected tuple must not contain any outlier match for it to work)
2. check how consistent other matches are with this hypothesis, i.e. in how far it is supported
3. use all supporting matches to refine the hypothesis and discard the rest

Finally, RANSAC selects the hypothesis with maximal support after a fixed number of trials or after sufficient support was reached

RANSAC

How often should we draw?.... Suppose

n - minimum number of data required to fit the model

k - nmb of iterations / trials performed by the algorithm

t – threshold to determine when a match fits a model

d - nmb of `inliers' needed for a model to be OK

t and *d* are typically chosen beforehand. The nmb of iterations *k* can then be calculated. Let *p* be the probability that RANSAC only selects inliers for the *n* data units generating a valid test at least once, i.e. the probability that the algorithm gets a good output. When *w* is the proportion of inliers (estimated),

$$1 - p = \left(1 - w^n\right)^k$$

is the probability that NO good hypothesis is selected

Supporting the matching step

Ex. cleaning matches based on RANSAC-Ep.Geom.

Matches are sought between the left and the right image. On the left one sees all matches found by matching corner descriptors only… on the right after RANSAC check spatial consistency; quite some pruning !

[Chum, Werner, Matas]

Supporting the matching step

… but remember that these tests make quite strong assumptions like rigidity (epipolar geometry) or planarity (proj.) – even if they tend to work quite well also in conditions where they hold only partially

RISK!

There are alternative schemes like topological filtering that do not have these issues,
but the large majority of systems are RANSAC-based.

Computer Vision