AIMS

Computer Vision

Lecture 1: Matching, indexing, and retrieval

Dr Andrea Vedaldi

For lecture notes, tutorial sheets, and updates see
http://www.robots.ox.ac.uk/~vedaldi/teach.html

Structure of the course

Lecture 1: Matching, indexing, and search

- Practical 1: Recognition of object instances

Lecture 2: Object category detection

- Practical 2: Object category detection

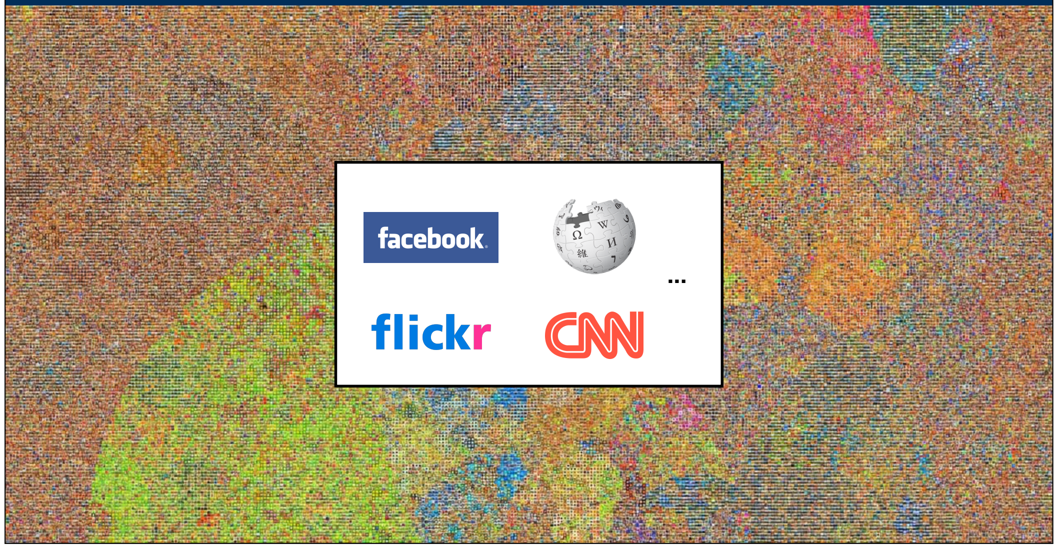
Lecture 3: Visual geometry 1/2: camera models and triangulation

Lecture 4: Visual geometry 2/2: reconstruction from multiple views

Lecture 5: Segmentation, tracking, and depth sensors

- Practical 3: Multiple view geometry

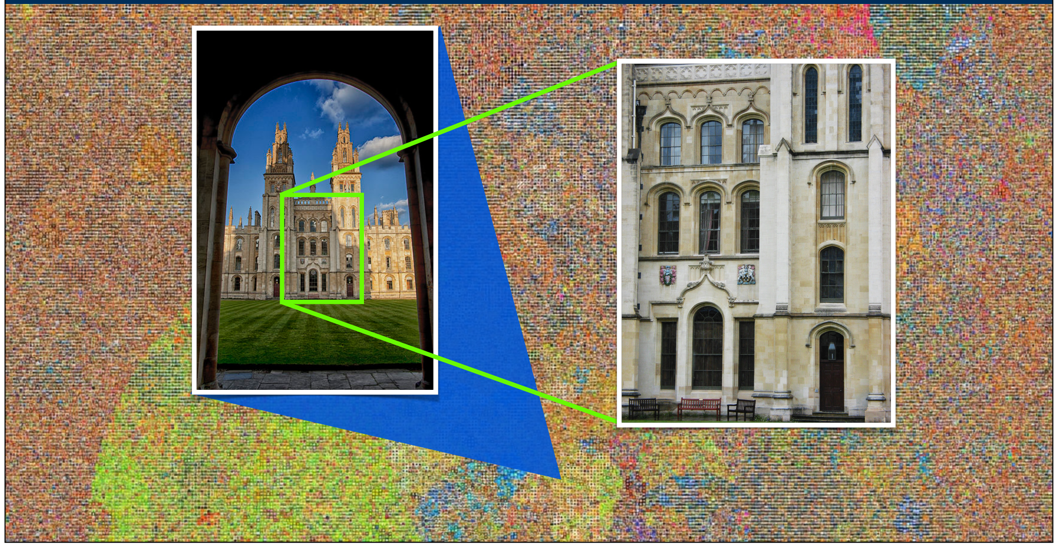
The Internet: 50 billion images and counting ...



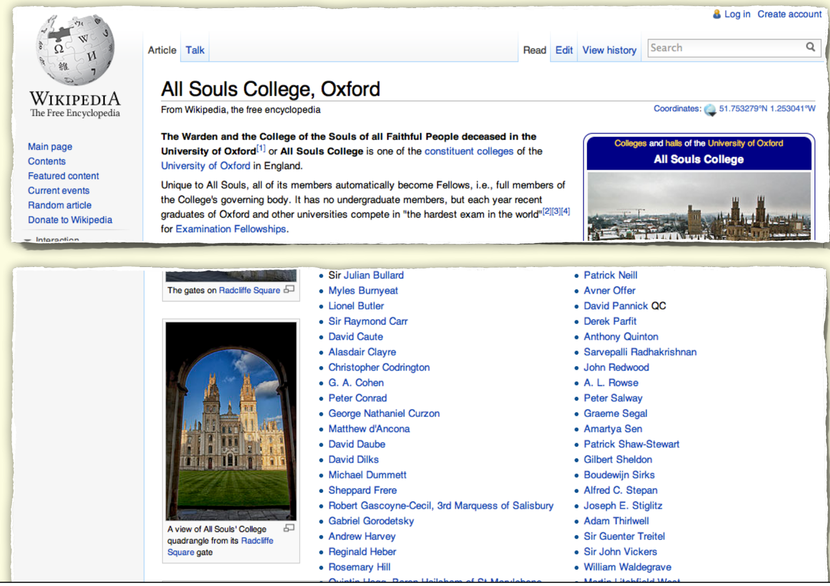
It may **not** contain the picture **you just took** …

.. but it likely contains a similar one!

5



6



Log in Create account

Wikipedia
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia

Article Talk Read Edit View history Search

All Souls College, Oxford

From Wikipedia, the free encyclopedia

Coordinates: 51.753279°N 1.253041°W

The Warden and the College of the Souls of all Faithful People deceased in the University of Oxford[1] or **All Souls College** is one of the constituent colleges of the University of Oxford in England.

Unique to All Souls, all of its members automatically become Fellows, i.e., full members of the College's governing body. It has no undergraduate members, but each year recent graduates of Oxford and other universities compete in "the hardest exam in the world"[2][3][4] for Examination Fellowships.

Colleges and halls of the University of Oxford
All Souls College

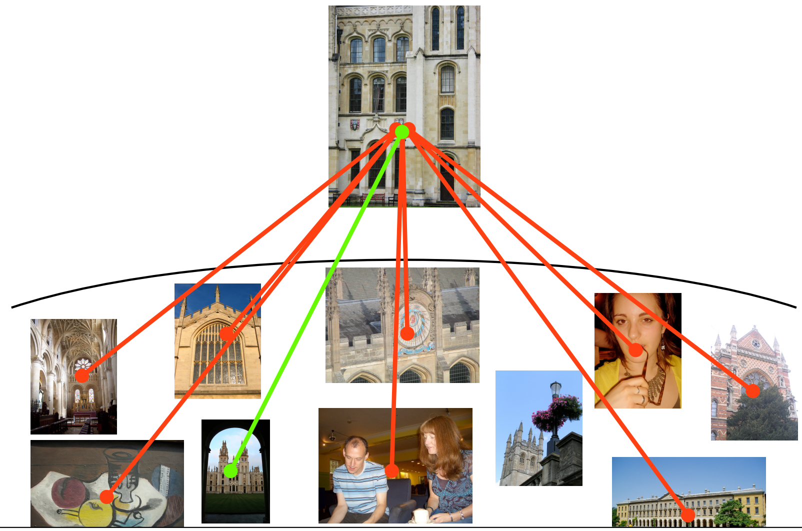
The gates on Radcliffe Square

A view of All Souls' College quadrangle from its Radcliffe Square gate

- Sir Julian Bullard
- Myles Burnyeat
- Lionel Butler
- Sir Raymond Carr
- David Caute
- Alasdair Clayre
- Christopher Codrington
- G. A. Cohen
- Peter Conrad
- George Nathaniel Curzon
- Matthew d'Ancona
- David Daube
- David Dilks
- Michael Dummett
- Sheppard Frere
- Robert Gascoyne-Cecil, 3rd Marquess of Salisbury
- Gabriel Gorodetsky
- Andrew Harvey
- Reginald Heber
- Rosemary Hill

- Patrick Neill
- Avner Offer
- David Pannick QC
- Derek Parfit
- Anthony Quinton
- Sarvepalli Radhakrishnan
- John Redwood
- A. L. Rowse
- Peter Salway
- Graeme Segal
- Amartya Sen
- Patrick Shaw-Stewart
- Gilbert Sheldon
- Boudewijn Sirks
- Alfred C. Stepan
- Joseph E. Stiglitz
- Adam Thirlwell
- Sir Guenter Treitel
- Sir John Vickers
- William Waldegrave

Goal: search a large collection for an image of the **same object**



8

Matching local features

Global geometric verification

Indexing using visual words

Evaluating retrieval systems

Matching local features

Global geometric verification

Indexing using visual words

Evaluating retrieval systems

Define a similarity function between images

$F(\mathbf{I}_1, \mathbf{I}_2)$ = confidence that the **object is the same**

$\mathbf{I}_1$

$\mathbf{I}_2$

Image similarity (I)

Compare images as vectors of pixels

$$F(\mathbf{I}_1, \mathbf{I}_2) = -\| \mathbf{I}_1 - \mathbf{I}_2 \|^2$$

$\mathbf{I}_1$

194	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
195	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
196	196	196	196	197	197	197	197	195
195	196	196	196	196	196	197	197	195
194	194	195	195	195	195	195	196	196
194	194	194	194	195	195	195	195	196
195	195	195	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
196	196	196	196	197	197	197	197	196
194	195	195	196	196	196	196	196	196
194	195	195	196	196	196	196	196	196
194	195	195	196	196	196	196	196	196
194	195	195	196	196	196	196	196	196
194	195	195	196	196	196	196	196	196
194	195	195	196	196	196	196	196	196
194	195	195	196	196	196	196	196	196
194	195	195	196	196	196	196	196	196
196	196	196	196	196	196	196	196	196
196	196	196	196	196	196	196	196	196
196	196	196	196	196	196	196	196	196

$(194 - 107)^2$

$(195 - 94)^2$

$(195 - 115)^2$

$\mathbf{I}_2$

107	131	132	118	126	130	129	134	140
94	104	126	120	124	125	129	129	133
89	94	116	123	117	123	131	134	136
96	101	111	119	121	128	136	141	143
110	108	120	118	133	130	134	134	132
110	106	122	120	126	125	123	129	128
[illegible]	[illegible]	115	118	115	123	125	132	139
126	137	118	117	122	127	135	133	137
100	107	116	116	116	121	128	132	143
120	91	112	114	123	129	139	142	147
148	135	99	110	119	126	136	142	145
106	139	124	110	117	125	133	143	139
103	112	130	118	122	133	141	137	148
105	106	118	131	124	132	142	134	144
97	101	115	119	117	129	136	135	135
95	88	110	136	125	128	140	148	144
91	98	119	132	125	131	138	136	149
97	100	123	129	131	131	135	136	150
94	106	113	127	127	131	136	136	148

Why do pixel values differ so much?

Nuisance factors

Viewpoint **Visibility** **Illumination** **Camera** **Noise**

$\mathbf{I}_1$

$\mathbf{I}_2$

Viewpoint and visibility

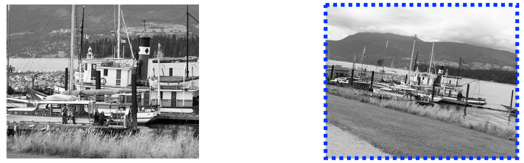
Handling a variable viewpoint

- As viewpoint changes pixels “move around” or even appear/disappear
- We need to **match corresponding pixels** before we can compare them

I_1 I_2

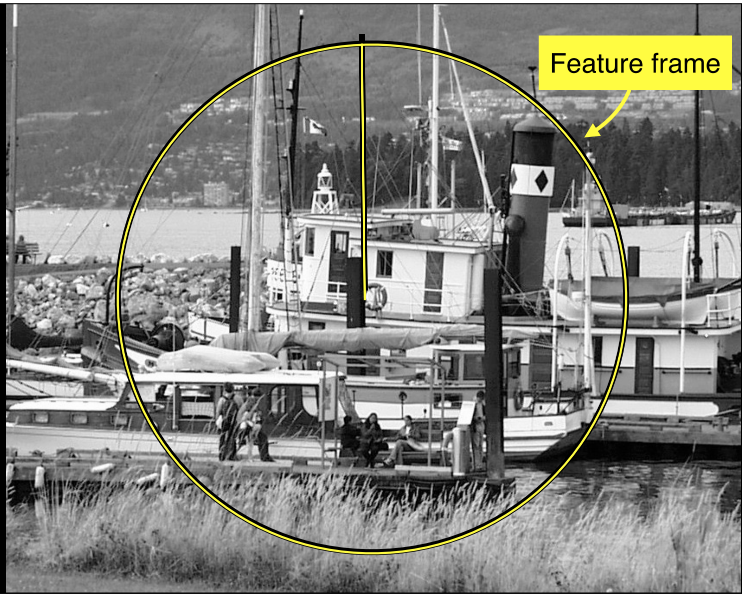
Matching and transformation

Matching can be seen as transforming or warping an image to another



Matching and transformation

Matching can be seen as transforming or warping an image to another



17

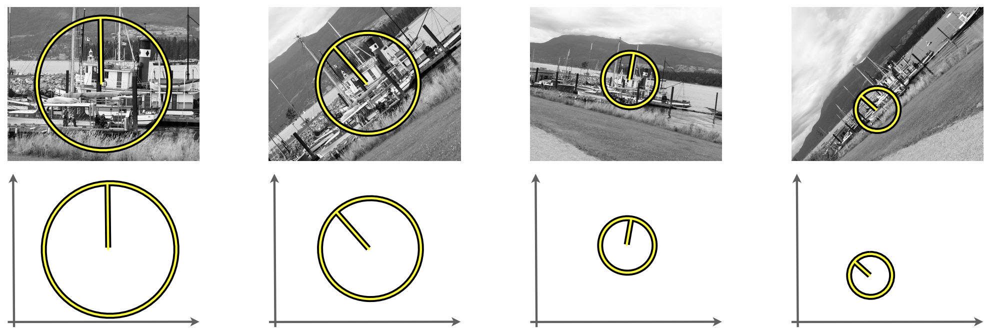
18

19

Similarity transformations

20

If the camera *rotates around* and *translates along* the *optical axis*, the image transforms according to a **similarity**: scale, rotation, and translation.

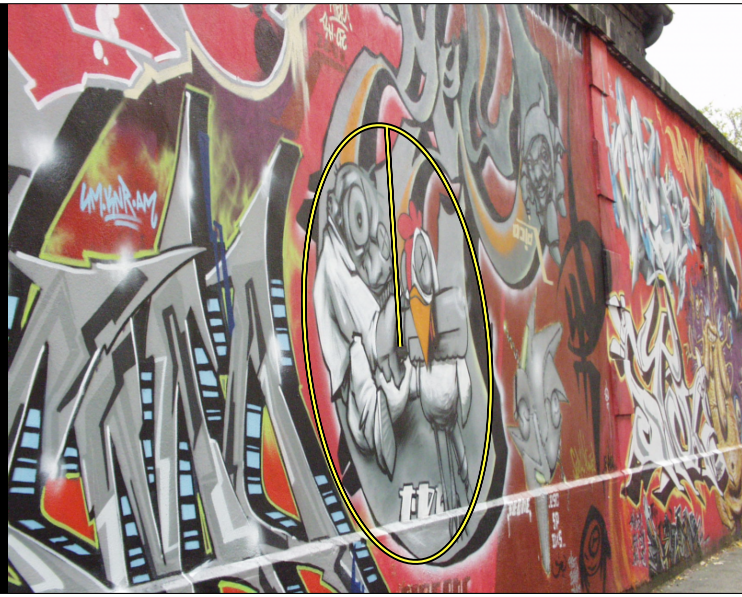


$$\begin{bmatrix}x'\\y'\end{bmatrix} = sR(\theta)\begin{bmatrix}x\\y\end{bmatrix} + \begin{bmatrix}t_x\\t_y\end{bmatrix} \qquad R(\theta) = \begin{bmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{bmatrix}$$

21

22

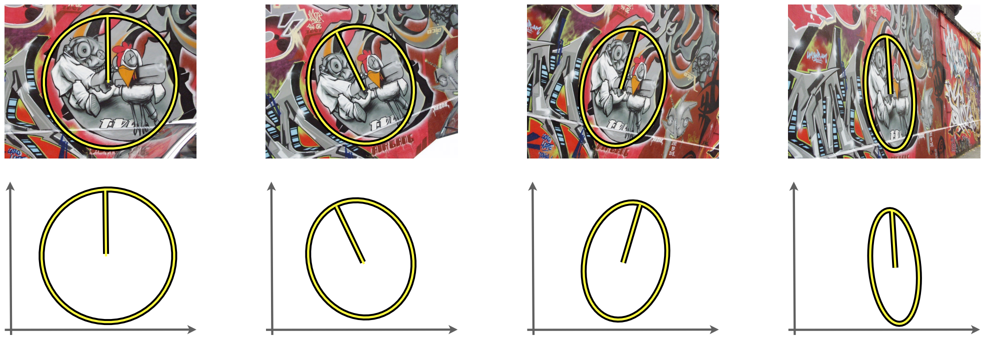
23



24

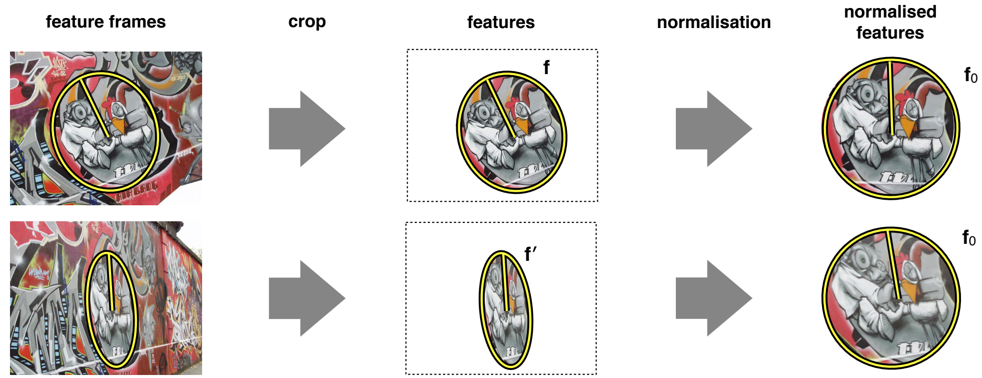
Homography/affine transformations

For *pure camera rotation* **or** if the *object is planar*, then the image transforms with an **homography** (approximated as an **affine transformation**).



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Comparing local features using normalisation

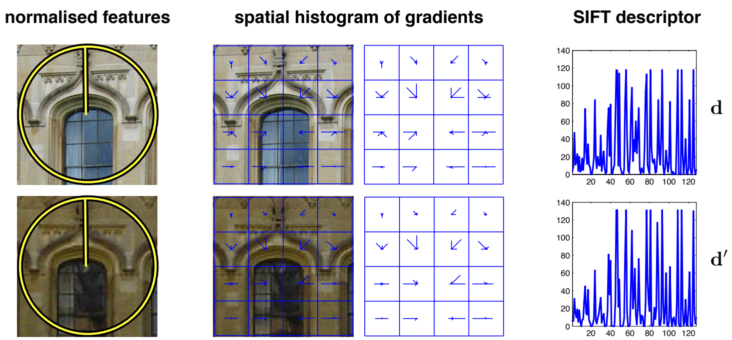


Consider corresponding feature frames **f** and **f′**.

Then **normalisation** *undoes the effect of a viewpoint change.*

After normalisation, pixels are in correspondence (matched) and can be compared directly.

Descriptors: SIFT

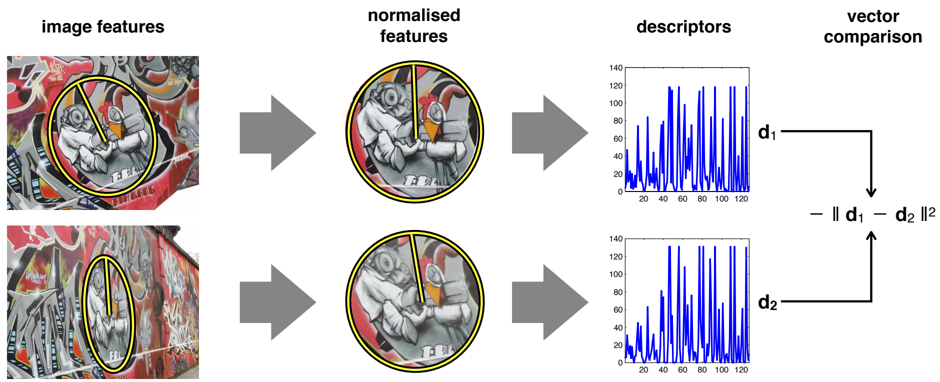


In practice, one **compares descriptors** rather than pixels. Descriptors:

- handle residual distortions, noise, illumination;
- make the representation more compact.

The most important example is the **SIFT descriptor**.

Summary: descriptors

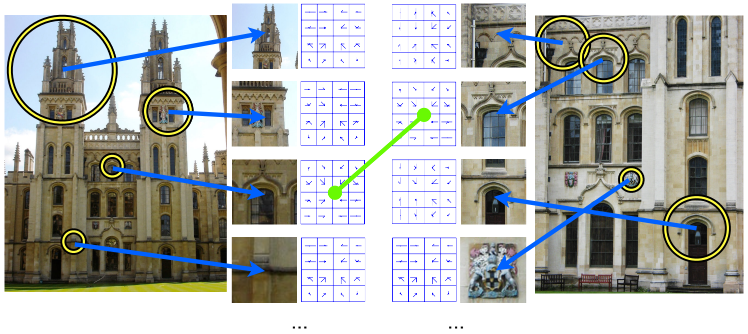


For each pair of image features

- Extract and normalize the corresponding image patches
- Compute their descriptor vectors
- Compare descriptors using the Euclidean distance

Question: how do we get the features in the first place?

Exhaustive matching

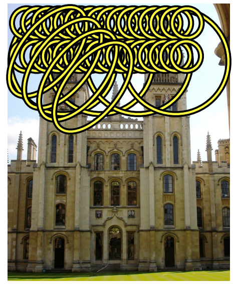


Exhaustive approach:

- Extract all possible features (all circles or all ellipses) from both images
- Test all feature pairs for possible matches

Testing all features guarantees that, if the “same feature” is visible in both images, then the corresponding patches are considered for matching.

Why exhaustive matching is unfeasible



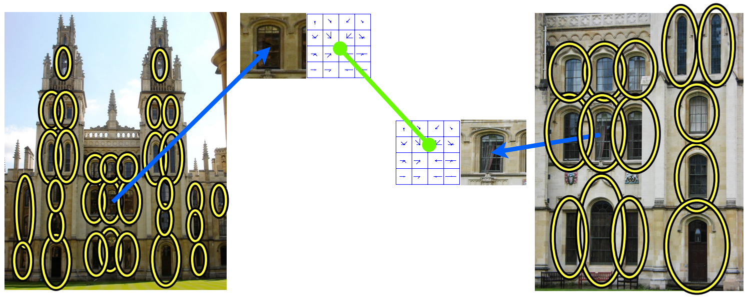
We need a method to **select** a small subset of features to match.

The cost of exhaustive matching is $O(N_1 N_2)$ where N_i is the number of features extracted from image $\mathbf{I}_i$.

Even after sampling the search space, the number of all possible features N_i is very large ($\sim 10^6$).

Exhaustive matching is just too expensive.

Co-variant feature detectors



A **detector** is a rule that **selects a small subset of features** for matching.

The key is **co-variance**: the selection mechanism must pick the “same” (i.e. corresponding) features after an image transformation.

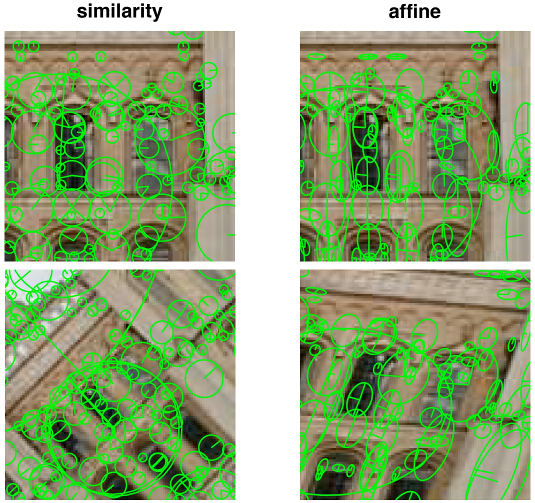
Example of a co-variant detection rule: “pick all the dark blobs”.

Co-variant detection, invariant descriptor

A feature extracted by the Harris-Affine detector independently from different frames of a video.

Note that the feature seems “glued on” the scene.

Co-variant detector types



Properties of a detector

- repeatability
- generality
- speed

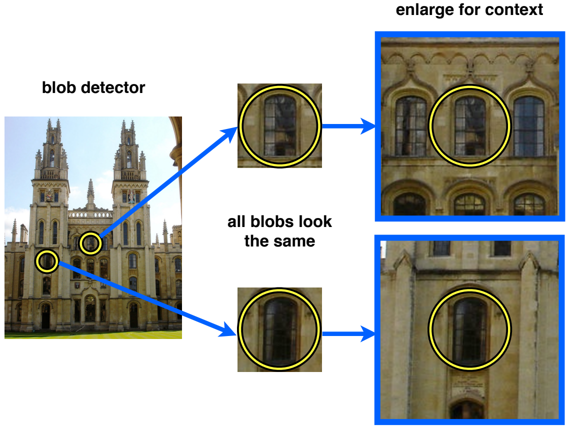
Benefits of increased covariance

- handle more general motions / objects

Cons of increased covariance

- less robust
- slower

Discriminability and support



In practice, descriptors are computed in a region **surrounding the feature**.

This is because the feature “visual anchors” (e.g. blobs) look the same and would be confused during matching.

Matching local features

Global geometric verification

Indexing using visual words

Evaluating retrieval systems

From local to global matching

Local matching

So far we have detected and then matched **local features**.

This is because normalisation is only possible if features are unoccluded and approximately planar.

Small features are much more likely to satisfy such assumptions.

On the contrary, the image as a whole is non-planar and contains plenty of self-occlusions.

Global matching

However, our goal is to compare images as a whole, not just individual patches.

Next, we will see how to build a **global similarity score** from patch-level local comparisons.

Matching all local features

Step 0: get an image pair

number of matches: 0

Matching all local features

Step 1: detect local features **f** and extract descriptors **d**

number of matches: 0

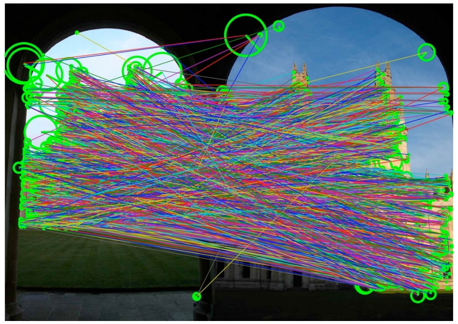
The left image has m features $(\mathbf{f}_1, \mathbf{d}_1), \ldots, (\mathbf{f}_m, \mathbf{d}_m)$

Right image has n feature $(\mathbf{f}'_1, \mathbf{d}'_1), \ldots, (\mathbf{f}'_n, \mathbf{d}'_n)$

Matching all local features

Step 2: match each descriptor to its closets one

number of matches: 2048



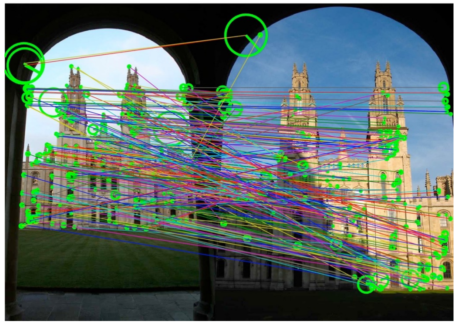
Match the i-th left feature to its right nearest-neighbour nn(i), where:

$$\mathrm{nn}(i) = \operatorname*{argmin}_{j=1,\ldots,m} \|\mathbf{d}_i - \mathbf{d}'_j\|^2$$

Matching all local features

Step 3: reject ambiguous matches using the **2nd-nn test**

number of matches: 293

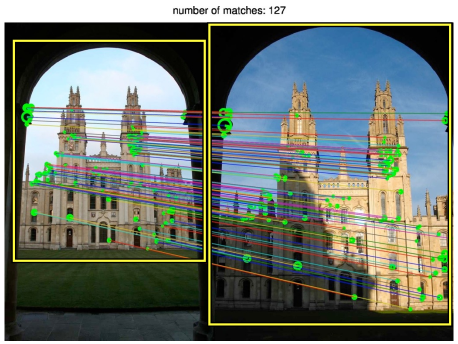


Accept a match $i \longmapsto \mathrm{nn}(i)$ only if it is at least a fraction $\tau = 0.9$ away from other possible matches:

$$\|\mathbf{d}_i - \mathbf{d}'_{\mathrm{nn}(i)}\|^2 < \tau \operatorname*{argmin}_{j \neq \mathrm{nn}(i)} \|\mathbf{d}_i - \mathbf{d}'_j\|^2$$

Matching all local features

Step 4: geometric verification

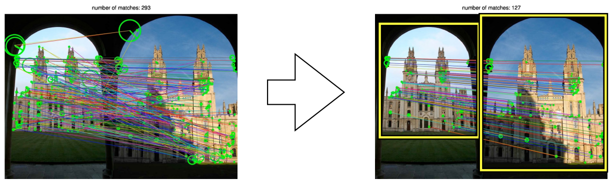


The final step is to test whether matches are consistent with an overall image transformation.

Inconsistent matches are rejected (see RANSAC).

RANSAC: optimization robust to outliers

(RANdom SAmple Consensus)



Input: M tentative feature matches $(\mathbf{x}_1, \mathbf{x}'_1), \ldots, (\mathbf{x}_M, \mathbf{x}'_M)$.

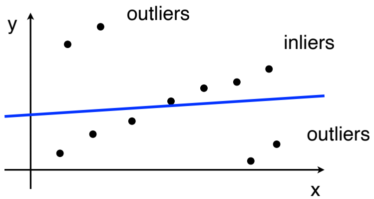
Output: affine transformation (A*,T*) with the largest number of inlier matches:

$$(A^*, T^*) = \operatorname{argmax}_{A,T} \left| \left\{ i : \|\mathbf{x}'_i - A\mathbf{x}_i - T\| < \epsilon \right\} \right|$$

1. Repeat a large number of times:
   A. Randomly sample a **minimal subset** of matches sufficient to estimate (A,T).
   B. Find **inliers**, i.e. other matches that are compatible with (A,T).
2. Return (A*,T*) as the pair (A,T) with the largest number of inliers.

The RANSAC Algorithm

RANdomized SAmples Consensus [Fishler & Bolles, 1981]



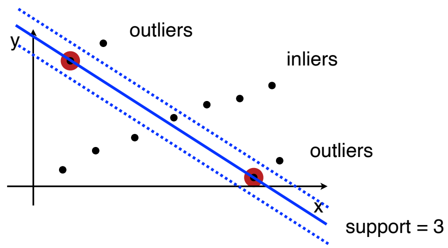
Consider the problem of fitting a line to a set of 2D points (xi,yi)

Often the data is contaminated by outliers, i.e. points that cannot be explained by the models

A method such as **least square** is heavily affected by outliers

The RANSAC Algorithm

RANdomized SAmples Consensus



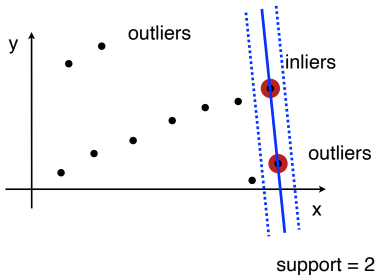
Pick two points at random instead, and fit the line

We may be unlucky, an pick two outliers

This can be detected by counting how many other points agree with the line

The RANSAC Algorithm

RANdomized SAmples Consensus

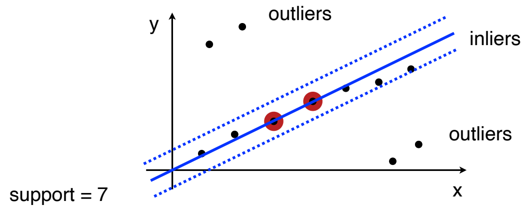


Play the game again

Once more we picked an outlier, so we obtained a small support

The RANSAC Algorithm

RANdomized SAmples Consensus

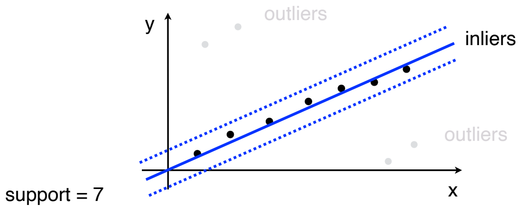


However, eventually we will be lucky, and pick two inliers

This can be detected because the support is much larger

The RANSAC Algorithm

RANdomized SAmples Consensus



Once the inlier set is identified, standard least square can be used to improve the solution

Why?

Image similarity (II)

By counting number of verified local feature matches

$F(\mathbf{I}_1, \mathbf{I}_2)$ = # of matches after geometric verification

$\mathbf{I}_1$ $\mathbf{I}_2$

Matching local features

Global geometric verification

Indexing using visual words

Evaluating retrieval systems

From image matching to image search

Our matching strategy can be used to search a handful of images exhaustively. However, this is far to slow to **search a database of a billion or more images** such as Flickr, FaceBook, or the Internet.

Example:

- L images in the database — e.g. 10^6 - 10^{10} (FaceBook)
- N features per image (incl. query) — e.g. 10^3 (~ SIFT detector)
- D dimensional feature descriptor — e.g. 10^2 (~ SIFT descriptor)
- Exhaustive search cost: $O(N^2 L D)$ — 10^{11} - 10^{15} ops = 100 days - 300 years
- Memory footprint: $O(NLD)$ — 1TB - 1PB

Goal: develop a method to search a million or more images on a single computer in under a second (and many more on computer clusters).

Issues:

- memory footprint
- matching cost (time)
- precision and recall

The inverted index

Used by Google to search the Web instantaneously

inverted index

term t	f_t	Inverted list for t
and	1	$\langle 6,2\rangle$
big	2	$\langle 2,2\rangle$ $\langle 3,1\rangle$
dark	1	$\langle 6,1\rangle$
did	1	$\langle 4,1\rangle$
gown	1	$\langle 2,1\rangle$
had	1	$\langle 3,1\rangle$
house	2	$\langle 2,1\rangle$ $\langle 3,1\rangle$
in	5	$\langle 1,1\rangle$ $\langle 2,2\rangle$ $\langle 3,1\rangle$ $\langle 5,1\rangle$ $\langle 6,2\rangle$
keep	3	$\langle 1,1\rangle$ $\langle 3,1\rangle$ $\langle 5,1\rangle$
keeper	3	$\langle 1,1\rangle$ $\langle 4,1\rangle$ $\langle 5,1\rangle$
keeps	3	$\langle 1,1\rangle$ $\langle 5,1\rangle$ $\langle 6,1\rangle$
light	1	$\langle 6,1\rangle$
never	1	$\langle 4,1\rangle$
night	3	$\langle 1,1\rangle$ $\langle 4,1\rangle$ $\langle 5,2\rangle$
old	4	$\langle 1,1\rangle$ $\langle 2,2\rangle$ $\langle 3,1\rangle$ $\langle 4,1\rangle$
sleep	1	$\langle 4,1\rangle$
sleeps	1	$\langle 6,1\rangle$
the	6	$\langle 1,3\rangle$ $\langle 2,2\rangle$ $\langle 3,3\rangle$ $\langle 4,1\rangle$ $\langle 5,3\rangle$ $\langle 6,2\rangle$
town	2	$\langle 1,1\rangle$ $\langle 3,1\rangle$
where	1	$\langle 4,1\rangle$

Inverted index

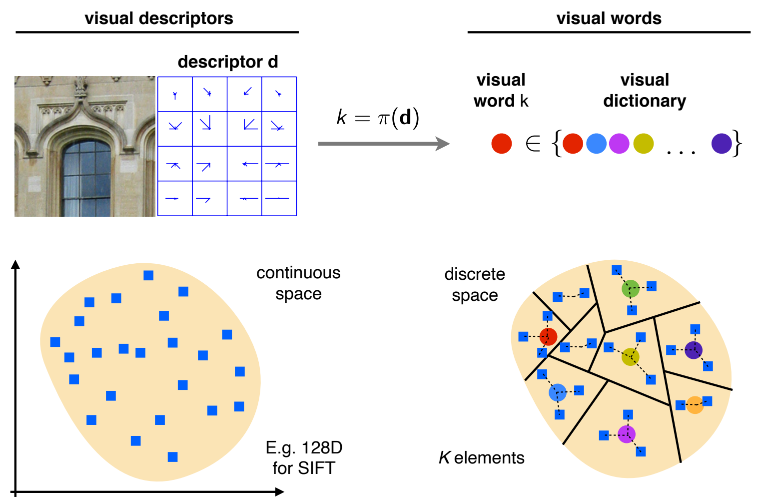
- For each word, lists all documents containing it as pairs ⟨`DocID, WordCount`⟩
- Efficient query resolution: given a word, return the corresponding list

Indexing images

- Image = document
- Word = ?

The key is to understand how to **extract "words" from images**

Visual words



The K-means algorithm

For learning a visual words vocabulary

The visual vocabulary is obtained by forming K **clusters** of example descriptors ($\mathbf{d}_1$, … $\mathbf{d}_M$). Here M may be in the order of a 1M, and K in the order of 10-100K.

The K cluster means ($\mu_1,\ldots,\mu_K$) are randomly initialised. Then the K-means algorithm alternates two steps:

- Find for each descriptor $\mathbf{d}_i$ the index $\pi(\mathbf{d}_i)$ of its closets mean:

$$\pi(\mathbf{d}_i) = \operatorname*{argmin}_{k=1,\ldots,K} \|\mathbf{d}_i - \mu_k\|^2$$

- Recompute each mean μ_k from the descriptor assigned to it:

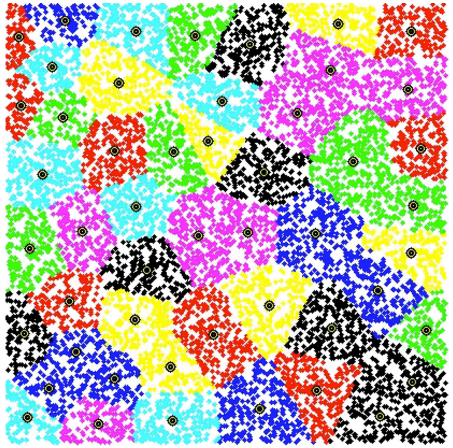
$$\mu_k = \text{average}\{\mathbf{d}_i : \text{nn}(\mathbf{d}_i) = k\}$$

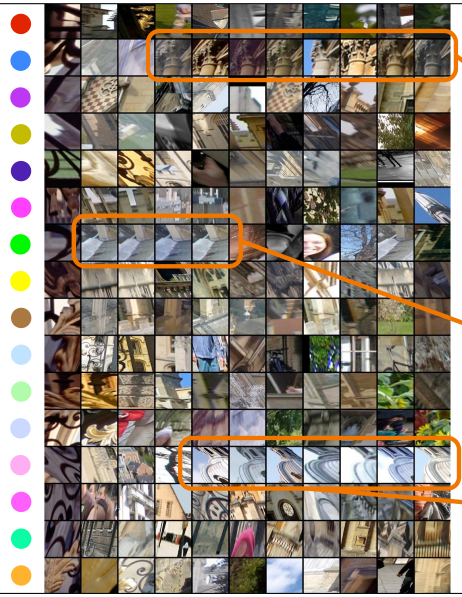
Once the means are trained, new descriptors d are quantised by mapping them to the closest mean:

$$\pi(\mathbf{d}) = \operatorname*{argmin}_{k=1,\ldots,K} \|\mathbf{d} - \mu_k\|^2$$

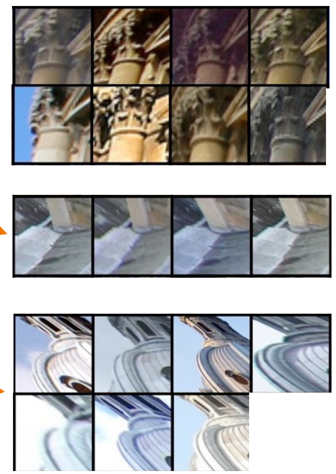
K-means example

Clustering a 2D dataset





Visual word examples. Each row is an equivalence class of patches mapped to the same cluster by K-means.



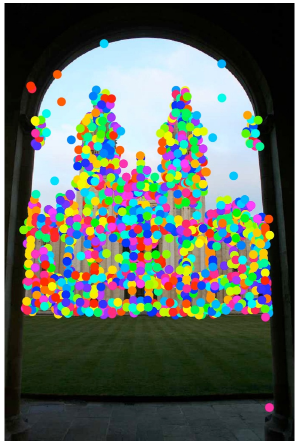
From local features to visual words

Two steps:

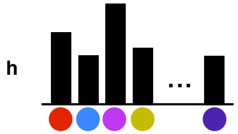
- **Extraction**. Extract local features and compute corresponding descriptors as before.
- **Quantisation**. Then map the descriptors to the K-means cluster centres to obtain the corresponding visual words.

Histogram of visual words

A simple but efficient global image descriptor



The **histogram of visual words** is the vector of the number of occurrences of the K visual words in the image:



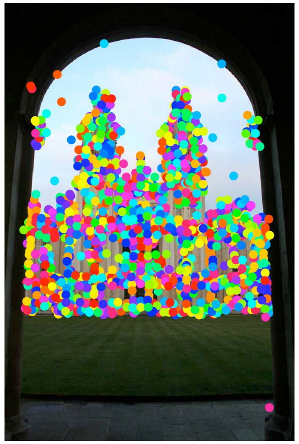
$$h_k = |\{\mathbf{d}_i : \pi(\mathbf{d}_i) = k\}|$$

If there are K visual words then **h** ϵ $\mathbb{R}^K$.

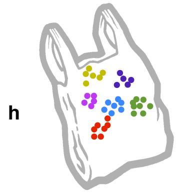
The vector **h** is a **global image descriptor**.

Histogram of visual words

A simple but efficient global image descriptor



This is also called a **bag of visual words** because it does not remember the relative positions of the features, just the number of occurrences.



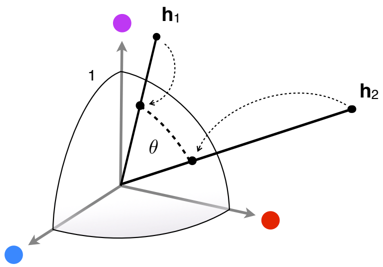
Hence, **h discards spatial information.**

Pros: more invariant to viewpoint changes and other nuisance factors.

Cons: less discriminative.

Comparing histograms

Cosine similarity



$$F(\mathbf{I}_1, \mathbf{I}_2) = \cos\theta = \left\langle \frac{\mathbf{h}_1}{\|\mathbf{h}_1\|}, \frac{\mathbf{h}_2}{\|\mathbf{h}_2\|} \right\rangle$$

Histogram of visual words can be compared as vectors.

The relative distribution of visual words is more informative than their absolute number of occurrences.

This intuition is captured by the **cosine similarity,** which computes the angle of the L^2-normalised histograms.

Image similarity (III)

By comparing bag-of-words descriptors

$$F(\mathbf{I}_1, \mathbf{I}_2) = \langle \mathbf{h}_1, \mathbf{h}_2 \rangle$$

$\mathbf{I}_1$

$\mathbf{I}_2$

Search as sparse matrix multiplication

Goal: given a query vector **h**, quickly compute its similarity with all the L vectors $\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3, \ldots, \mathbf{h}_L$ in the database (one per indexed image).

Express this as a vector-matrix multiplication:

$$\overset{\mathbf{h}}{\begin{bmatrix} 0 & 0.1 & 0.2 & 0 & \ldots & 0 & \ldots & 0.1 \end{bmatrix}} \times \begin{array}{ccccc} \mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_3 & \ldots & \mathbf{h}_L \\ \end{array} \begin{bmatrix} 0 & 0 & 0 & \ldots & 0.1 \\ 0 & 0.1 & 0 & \ldots & 0 \\ 0.2 & 0 & 0 & \ldots & 0 \\ 0.1 & 0 & 0.3 & \ldots & 0.1 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0.1 & \ldots & 0.2 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0.01 & 0.1 & 0 & \ldots & 0 \end{bmatrix}$$

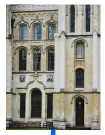
The naive **multiplication cost** is O(K L), where K is the number of visual words and L is the database size.

However, histograms are often highly sparse. If only a fraction $\rho \ll 1$ of entries is non-zero, then the cost reduces to $O(\rho K L)$ or even $O(\rho^2 K L)$.

The **space required** i is also only $O(\rho K L)$.

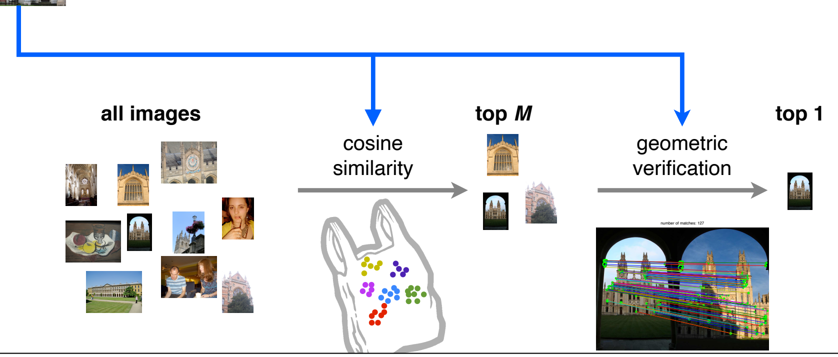
Summary: image indexing and retrieval

query **I**



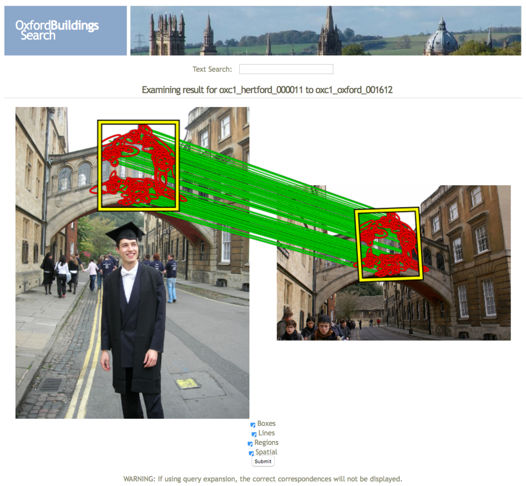
Given a query image **I**, we search the database by combining the two similarities:

1. The **fast but unreliable** cosine similarity to obtain a short list of $M \approx 100$ possible matches.
2. The **slow but reliable** geometric verification to rerank the top M matches.



Demo

http://www.robots.ox.ac.uk/~vgg/demo/



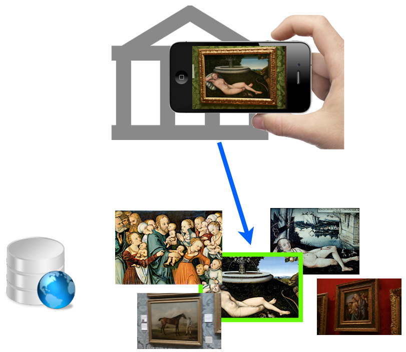
Matching local features

Global geometric verification

Indexing using visual words

Evaluating retrieval systems

Evaluating of a retrieval system



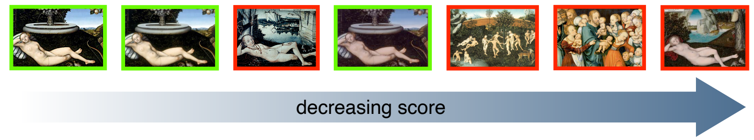
We now have a system that can match a given picture to a large database of images (e.g. Wikipedia).

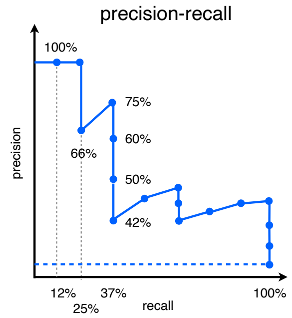
Besides speed, a **good retrieval system** must have two fundamental properties:

1. **Precision**, i.e. the ability to return **only** images that match the query.
2. **Recall**, i.e. the ability to return **all** the images that match the query.

Precision-recall curves

Assess the quality of a ranked result list





Consider all images up to rank *r* in the list:

- **Precision** @*r*: fraction of correct results in the top *r*.
- **Recall** @*r*: fraction of relevant database images that are contained in the top *r*.

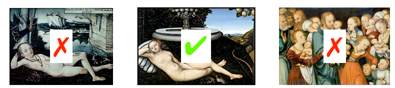
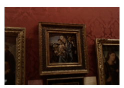
The **Average-Precision** (AP) is (roughly) the area under the PR curve.

AP is a single number summarising the overall quality of the result list.

Evaluating an image retrieval system

A benchmark usually has 1) a large image database and 2) a number of test queries for which the correct answer (relevant/irrelevant images) is known.

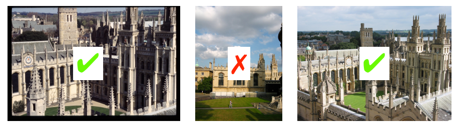
The retrieval system is evaluated in term of **mean average precision** (mAP), which is the mean AP of the test queries.

query	retrieval results	AP
		35%
		100%
		75%
...	... ...	...
	mean average precision (mAP)	53%

Example benchmark: Oxford 5K

http://www.robots.ox.ac.uk/~vgg/data/oxbuildings/

Query **Retrieved Images**



...

Dataset content

- ~ 5K images of Oxford
- An optional additional set of confounder (irrelevant) images
- 58 test queries