AIMS Computer Vision
Lecture 1: Matching, indexing, and search
HT 2017
Andrea Vedaldi

For slides and up-to-date information:
http://www.robots.ox.ac.uk/~vedaldi/teach.html

Structure of the course

An overview

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

Making sense of trillions of images

Organising, searching and distilling visual information

10% of all the images in the world have been collected this year

BBC: 2.4M videos
Facebook: 140B images
~100M monitoring, safety & surveillance cameras

Trillions of images.

Thousand of potential applications.
The Internet: 50 billion images and counting...

It may not contain the picture you just took...

.. but it likely contains a similar one!

All Souls College, Oxford

The Warden and the College of the Souls of all Faithful People deceased in the University of Oxford. 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 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" for Examination Fellowships.
Goal: search a large collection for an image of the **same object**

Define a similarity function between images

\[ F(I_1, I_2) = \text{confidence that the object is the same} \]

**Local image similarity**
by matching features

**Global image similarity** with geometric verification

Indexing and searching large image collections using visual words

Evaluating image retrieval systems

SIFT
Nuisance factors

Why do pixel values differ so much?

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

Matching and transformation

Matching can be seen as transforming or warping an image in another
Similarity transformations

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

\[
\begin{bmatrix} x' \\ y' \end{bmatrix} = sR(\theta) \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \quad R(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}
\]
Homography/affine transformation

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

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

More precisely, a flat surface transforms with an homography

Note: the effect of an homography applied to an ellipse can always be emulated by an affine transformation.

Comparing local features using normalisation

Suppose corresponding features \( f \) and \( f' \) are given to us.
Then **normalisation** undoes the effect of a viewpoint change.
After normalisation, pixels are in correspondence (matched) and can be compared.
In practice, one compares descriptors rather than pixels. Descriptors:
- handle residual distortions, noise, illumination;
- make the representation more compact.

Example: SIFT descriptors

**Proposal: exhaustive matching**

For the list of all possible transformations of a feature (unit circle)
- similarity + unit circle → all possible circles
- affine + unit circle → all possible ellipses

If the “same feature” is visible in both images, it must be somewhere in such lists.
Thus, try to match everything with everything.

**Why exhaustive matching is unfeasible**

The cost of exhaustive matching is \( O(N_1 N_2) \) where \( N_i \) is the number of features extracted from image \( i \).

\( N_i \) is a very large number:
- there is one feature for each transformation
- in principle, \( N_i = \infty \)
- even with a sufficiently fine discretisation, \( N_i \) is in the order of millions
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” (corresponding) features after an image transformation. Example of a co-variant detection rule: “pick all the dark blobs.”

A feature extracted by the Harris-Affine detector independently from different frames of a video. Note that the feature seems “glued on” the scene.

In practice, descriptors are computed in a region surrounding the feature. This is because all feature “anchors” (e.g. blobs) look the same and would be confused.
Feature locality vs objects

Did we succeed?
▶ Goal: match objects
▶ Achieved: match patches
Not there yet!

We did this to achieve invariance
▶ possible only for (simple) invertible transformation classes e.g., similarity, affinity
▶ cannot work with occlusions

Local image similarity by matching features

Global image similarity with geometric verification

Indexing and searching large image collections using visual words
Evaluating image retrieval systems
SIFT

Local matching
▶ So far we have detected and then matched local features.
▶ This is because normalisation is only possible for unoccluded, approximately planar structures.
▶ Small enough image fragments tend to satisfy such assumptions, but not the image as a whole.

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.

From local to global matching

Matching all local features

Step 0: get an image pair

Matching all local features

number of matches: 0
Step 1: detect local features $f$ and extract descriptors $d$

Matching all local features

The left image has $m$ features $(f_1, d_1), \ldots, (f_m, d_m)$
Right image has $n$ features $(f'_1, d'_1), \ldots, (f'_n, d'_n)$

Step 2: match each descriptor to its closest one

Matching all local features

Match the $i$-th left feature to its right nearest-neighbour $\text{nn}(i)$, where:

$$\text{nn}(i) = \arg \min_{j=1,\ldots,m} \| d_i - d'_j \|^2$$

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

Matching all local features

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

$$\| d_i - d'_{\text{nn}(i)} \|^2 < \tau \min_{j \neq \text{nn}(i)} \| d_i - d'_j \|^2$$

Step 4: geometric verification

Matching all local features

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

Inconsistent matches are rejected (see RANSAC).
Image similarity (II)
By counting number of **verified** local feature matches

\[ F(I_1, I_2) = \text{# of matches after geometric verification} \]

For geometric verification

**RANSAC**

Input: \( M \) tentative feature matches \((x_1, x'_1), \ldots, (x_M, x'_M)\).

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

\[
(A^*, T^*) = \arg\max_{A, T} \{ i : \| x_i - Ax_i - T \| < \epsilon \} 
\]

1. Repeat a large number of times:
   A. Randomly sample a minimal subset of matches sufficient to estimate \((A, T)\).
   B. Compute how many other inlier matches are compatible with \((A, T)\).
2. Return the estimated \((A, T)\) that has the largest number of inliers.

The RANSAC Algorithm

**RANdomized SAmple Consensus** [Fishler & Bolles, 1981]

Consider the problem of fitting a line to a set of 2D points \((x_i, y_i)\)

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

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
Play the game again
Once more we picked an outlier, so we obtained a small support

However, eventually we will be lucky, and pick two inliers
This can be detected because the support is much larger

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

Local image similarity
by matching features

Global image similarity with geometric verification

Indexing and searching large image collections using visual words

Evaluating image retrieval systems

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

Example:
- \( L \) images in the database
- \( N \) features per image (incl. query)
- \( D \) dimensional feature descriptor
- Exhaustive search cost: \( O(N^2L D) \)
- Memory footprint: \( O(NLD) \)

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

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).

The inverted index

I.e. how Google can search the Web so fast

Inverted index
- For each word, lists all documents containing it as pairs \((\text{DocID, WordCount})\)
- 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

The K-means algorithm

For learning a visual words vocabulary

The visual vocabulary is obtained by forming \( K \) clusters of example descriptors \( d_1, \ldots, d_M \). 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:

1. Assign each example descriptor \( d_i \) to the closest mean \( \pi(d_i) \):
   \[
   \pi(d_i) = \arg\min_{k=1,\ldots,K} \|d_i - \mu_k\|^2
   \]

2. Recompute each mean \( \mu_k \) from the descriptor assigned to it:
   \[
   \mu_k = \text{average}\{d_i : \pi(n(d_i)) = k\}
   \]

Once trained, new descriptors \( d \) are quantised by mapping them to the closest mean:
\[
\pi(d) = \arg\min_{k=1,\ldots,K} \|d - \mu_k\|^2
\]
Clustering a 2D dataset

K-means example

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.

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:

\[ h_k = |\{d_i : \pi(d_i) = k\}| \]

If there are \( K \) visual words then \( h \in \mathbb{R}^K \).

The vector \( h \) is a global image descriptor.
A simple but efficient global image descriptor

Histogram of visual words

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

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

$$F(I_1, I_2) = \langle h_1, h_2 \rangle$$
Search as sparse matrix multiplication

**Goal:** given a query vector $h$, quickly compute its similarity with all the $L$ vectors $h_1$, $h_2$, $h_3$, ..., $h_L$ in the database (number of images).

Express this as a vector-matrix multiplication:

$$
\begin{bmatrix}
0 & 0.1 & 0.2 & 0 & \ldots & 0 & \ldots & 0.1 \\
\end{bmatrix}
\times
\begin{bmatrix}
h_1 & h_2 & h_3 & \ldots & h_L \\
0 & 0 & 0 & \ldots & 0.1 \\
0 & 0.1 & 0 & \ldots & 0 \\
0.2 & 0 & 0 & \ldots & 0 \\
0.1 & 0 & 0 & \ldots & 0.1 \\
\ldots & \ldots & \ldots & \ldots & \ldots \\
0 & 0 & 0.3 & \ldots & 0.1 \\
\ldots & \ldots & \ldots & \ldots & \ldots \\
0.01 & 0.1 & 0 & \ldots & 0 \\
\end{bmatrix}
$$

The naive **multiplication cost** is $O(KL)$, 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 $O(\rho KL)$ or even $O(\rho^2 KL)$.

The **space required** is also only $O(\rho KL)$.

Overview of fast image retrieval

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.

Local image similarity by matching features

Global image similarity with geometric verification

Indexing and searching large image collections using visual words

Evaluating image retrieval systems

SIFT
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 of returning only images that match the query.
2. **Recall**, i.e. the ability of returning all the images that match the query.

**Precision-recall curves**

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.

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 terms of **mean average precision** (mAP), which is the mean AP of the test queries.

<table>
<thead>
<tr>
<th>query</th>
<th>retrieval results</th>
<th>AP</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image 1" /></td>
<td><img src="checkmark.png" alt="Checkmark" /> <img src="x.png" alt="X" /></td>
<td>35%</td>
</tr>
<tr>
<td><img src="image2.png" alt="Image 2" /></td>
<td><img src="checkmark.png" alt="Checkmark" /> <img src="x.png" alt="X" /></td>
<td>100%</td>
</tr>
<tr>
<td><img src="image3.png" alt="Image 3" /></td>
<td><img src="checkmark.png" alt="Checkmark" /> <img src="x.png" alt="X" /></td>
<td>75%</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Local image similarity by matching features**

**Global image similarity with geometric verification**

Indexing and searching large image collections using visual words

Evaluating image retrieval systems

**SIFT**
SIFT detector

More details on Wednesday

The SIFT detector anchors key points to image **blobs**

Blobs are defined as local maxima/minima of the Laplacian operator

Intuition: the Laplacian operator looks like a blob

Gaussian scale space

Blobs are searched at multiple scale by resizing the image using a Gaussian filter

Laplacian of Gaussian scale space

Given the Gaussian scale space, we can quickly compute the corresponding Laplacian of Gaussian scale space, and search for blobs of all sizes.
SIFT detector
Orientation assignment

SIFT key points have a location, a scale, as well as an orientation

The orientation is determined by looking at the dominant direction of the gradient in the patch

Algorithm:
- Compute the histogram of oriented gradients (crf. previous term)
- Set the angle of the patch to the histogram peak

SIFT descriptor
Histogram of oriented gradients

Image gradients → Keypoint descriptor