Image alignment
Alignment applications

Panorama stitching

http://matthewalunbrown.com/autostitch/autostitch.html
Alignment applications

- A look into the past
Alignment applications

- A look into the past
Alignment applications

• Cool video
### Alignment applications

#### Instance recognition

<table>
<thead>
<tr>
<th>Query</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Query Image" /></td>
<td><strong>notre dame</strong></td>
</tr>
<tr>
<td><img src="image2" alt="Query Image" /></td>
<td><strong>ecole militaire et tour eiffel</strong></td>
</tr>
<tr>
<td><img src="image3" alt="Query Image" /></td>
<td><strong>notre dame de paris choir wall</strong></td>
</tr>
<tr>
<td><img src="image4" alt="Query Image" /></td>
<td><strong>avenue de wagram</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Query</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image5" alt="Query Image" /></td>
<td><strong>notre dame</strong></td>
</tr>
<tr>
<td><img src="image6" alt="Query Image" /></td>
<td><strong>eiffel tower</strong></td>
</tr>
<tr>
<td><img src="image7" alt="Query Image" /></td>
<td><strong>jef aerosol : quartier rue mouffetard</strong></td>
</tr>
<tr>
<td><img src="image8" alt="Query Image" /></td>
<td><strong>arc de triomphe et avenue des champs-elysees vu du 3eme etage de la tour eiffel</strong></td>
</tr>
</tbody>
</table>

T. Weyand and B. Leibe, *Visual landmark recognition from Internet photo collections: A large-scale evaluation*, CVIU 2015
Alignment challenges

- Small degree of overlap
- Intensity changes

- Occlusion, clutter, viewpoint change
Feature-based alignment

• Search sets of feature matches that agree in terms of:
  a) Local appearance
  b) Geometric configuration
Feature-based alignment: Overview

- Alignment as fitting
  - Affine transformations
  - Homographies
- Robust alignment
  - Descriptor-based feature matching
  - RANSAC
- Large-scale alignment
  - Inverted indexing
  - Vocabulary trees
- Application: searching the night sky
Alignment as fitting

- Previous lectures: fitting a model to features in one image

Find model $M$ that minimizes

$$\sum_i \text{residual}(x_i, M)$$
Alignment as fitting

- Previous lectures: fitting a model to features in one image

- Find model $M$ that minimizes
  \[ \sum_i \text{residual}(x_i, M) \]

- Alignment: fitting a model to a transformation between pairs of features (matches) in two images

- Find transformation $T$ that minimizes
  \[ \sum_i \text{residual}(T(x_i), x'_i) \]
2D transformation models

- Similarity (translation, scale, rotation)
- Affine
- Projective (homography)
Let’s start with affine transformations

- Simple fitting procedure (linear least squares)
- Approximates viewpoint changes for roughly planar objects and roughly orthographic cameras
- Can be used to initialize fitting for more complex models
Fitting an affine transformation

- Assume we know the correspondences, how do we get the transformation?

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} = \begin{bmatrix}
  m_1 & m_2 \\
  m_3 & m_4
\end{bmatrix} \begin{bmatrix}
  x_i \\
  y_i
\end{bmatrix} + \begin{bmatrix}
  t_1 \\
  t_2
\end{bmatrix}
\]

Want to find \( M, t \) to minimize

\[
\sum_{i=1}^{n} \left\| x'_i - Mx_i - t \right\|^2
\]
Fitting an affine transformation

- Assume we know the correspondences, how do we get the transformation?

\[
\begin{bmatrix}
x_i' \\
y_i'
\end{bmatrix} = \begin{bmatrix}
m_1 & m_2 \\
m_3 & m_4
\end{bmatrix} \begin{bmatrix}
x_i \\
y_i
\end{bmatrix} + \begin{bmatrix}
t_1 \\
t_2
\end{bmatrix}
\]

\[
\begin{bmatrix}
m_1 \\
m_2 \\
m_3 \\
m_4
\end{bmatrix} = \begin{bmatrix}
x_i' \\
y_i'
\end{bmatrix} + \begin{bmatrix}
t_1 \\
t_2
\end{bmatrix}
\]
Fitting an affine transformation

\[
\begin{bmatrix}
    x_i & y_i & 0 & 0 & 1 & 0 \\
    0 & 0 & x_i & y_i & 0 & 1 \\
 \end{bmatrix}
\begin{bmatrix}
    m_1 \\
    m_2 \\
    m_3 \\
    m_4 \\
    t_1 \\
    t_2 \\
\end{bmatrix}
= 
\begin{bmatrix}
    \cdots \\
    x_i' \\
    y_i' \\
    \cdots \\
\end{bmatrix}
\]

- Linear system with six unknowns
- Each match gives us two linearly independent equations: need at least three to solve for the transformation parameters
Fitting a plane projective transformation

- **Homography**: plane projective transformation (transformation taking a quad to another arbitrary quad)
Homography

- The transformation between two views of a planar surface
- The transformation between images from two cameras that share the same center
Application: Panorama stitching

Source: Hartley & Zisserman
Fitting a homography

- Recall: homogeneous coordinates

\[(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \text{Converting to homogeneous image coordinates}\]

\[\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w) \quad \text{Converting from homogeneous image coordinates}\]
Fitting a homography

- Recall: homogeneous coordinates

\[(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}\]

Converting to homogeneous image coordinates

\[\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)\]

Converting from homogeneous image coordinates

- Equation for homography:

\[
\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}
\]
Fitting a homography

- Equation for homography:

\[
\begin{bmatrix}
  x'_i \\
  y'_i \\
  1
\end{bmatrix}
= \begin{bmatrix}
  h_{11} & h_{12} & h_{13} \\
  h_{21} & h_{22} & h_{23} \\
  h_{31} & h_{32} & h_{33}
\end{bmatrix}
\begin{bmatrix}
  x_i \\
  y_i \\
  1
\end{bmatrix}
\]

\[\lambda x'_i = H x_i\]

\[x'_i \times H x_i = 0\]

\[
\begin{bmatrix}
  x'_i \\
  y'_i \\
  1
\end{bmatrix}
\times
\begin{bmatrix}
  h_1^T x_i \\
  h_2^T x_i \\
  h_3^T x_i
\end{bmatrix}
=
\begin{bmatrix}
  y'_i h_3^T x_i - h_2^T x_i \\
  h_1^T x_i - x'_i h_3^T x_i \\
  x'_i h_2^T x_i - y'_i h_1^T x_i
\end{bmatrix}
\]

\[
\begin{bmatrix}
  0^T & -x_i^T & y_i^T x_i^T \\
  x_i^T & 0^T & -x'_i x_i^T \\
  -y'_i x_i^T & x'_i x_i^T & 0^T
\end{bmatrix}
\begin{bmatrix}
  h_1 \\
  h_2 \\
  h_3
\end{bmatrix}
= 0
\]

3 equations, only 2 linearly independent
Fitting a homography

\[
\begin{bmatrix}
0^T & x_1^T & -y_1' x_1^T \\
x_1^T & 0^T & -x_1' x_1^T \\
\vdots & \vdots & \vdots \\
0^T & x_n^T & -y_n' x_n^T \\
x_n^T & 0^T & -x_n' x_n^T
\end{bmatrix}
\begin{bmatrix}
h_1 \\
h_2 \\
h_3
\end{bmatrix} = 0 
A \mathbf{h} = 0
\]

- \( H \) has 8 degrees of freedom (9 parameters, but scale is arbitrary)
- One match gives us two linearly independent equations
- Homogeneous least squares: find \( \mathbf{h} \) minimizing \( \| A \mathbf{h} \|^2 \)
  - Eigenvector of \( A^T A \) corresponding to smallest eigenvalue
  - Four matches needed for a minimal solution
Robust feature-based alignment

• So far, we’ve assumed that we are given a set of “ground-truth” correspondences between the two images we want to align
• What if we don’t know the correspondences?
Robust feature-based alignment

- So far, we’ve assumed that we are given a set of “ground-truth” correspondences between the two images we want to align
- What if we don’t know the correspondences?
Robust feature-based alignment
Robust feature-based alignment

- Extract features
Robust feature-based alignment

- Extract features
- Compute *putative matches*
Robust feature-based alignment

- Extract features
- Compute putative matches
- Loop:
  - Hypothesize transformation $T$
Robust feature-based alignment

- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation $T$
  - *Verify* transformation (search for other matches consistent with $T$)
Robust feature-based alignment

- Extract features
- Compute *putative matches*
- Loop:
  - Hypothesize transformation $T$
  - Verify transformation (search for other matches consistent with $T$)
Generating putative correspondences
Generating putative correspondences

- Need to compare feature descriptors of local patches surrounding interest points
Feature descriptors

• Recall: feature detection and description
Feature descriptors

- Simplest descriptor: vector of raw intensity values
- How to compare two such vectors?
  - Sum of squared differences (SSD)
    \[
    SSD(u, v) = \sum_i (u_i - v_i)^2
    \]
    - Not invariant to intensity change

- Normalized correlation
  \[
  \rho(u, v) = \frac{(u - \bar{u}) \cdot (v - \bar{v})}{\|u - \bar{u}\| \|v - \bar{v}\|} = \frac{\sum_i (u_i - \bar{u})(v_i - \bar{v})}{\sqrt{\left(\sum_j (u_j - \bar{u})^2\right)\left(\sum_j (v_j - \bar{v})^2\right)}}
  \]
  - Invariant to affine intensity change
Disadvantage of intensity vectors as descriptors

- Small deformations can affect the matching score a lot
Feature descriptors: SIFT

- Descriptor computation:
  - Divide patch into 4x4 sub-patches
  - Compute histogram of gradient orientations (8 reference angles) inside each sub-patch
  - Resulting descriptor: $4 \times 4 \times 8 = 128$ dimensions

Feature descriptors: SIFT

- Descriptor computation:
  - Divide patch into 4x4 sub-patches
  - Compute histogram of gradient orientations (8 reference angles) inside each sub-patch
  - Resulting descriptor: 4x4x8 = 128 dimensions

- Advantage over raw vectors of pixel values
  - Gradients less sensitive to illumination change
  - Pooling of gradients over the sub-patches achieves robustness to small shifts, but still preserves some spatial information

Feature matching

- Generating *putative matches*: for each patch in one image, find a short list of patches in the other image that could match it based solely on appearance.
Problem: Ambiguous putative matches

Source: Y. Furukawa
Rejection of unreliable matches

- How can we tell which putative matches are more reliable?
- Heuristic: compare distance of **nearest** neighbor to that of **second** nearest neighbor
  - Ratio of closest distance to second-closest distance will be *high* for features that are *not* distinctive

RANSAC

- The set of putative matches contains a very high percentage of outliers

**RANSAC loop:**
1. Randomly select a *seed group* of matches
2. Compute transformation from seed group
3. Find *inliers* to this transformation
4. If the number of inliers is sufficiently large, re-compute least-squares estimate of transformation on all of the inliers

Keep the transformation with the largest number of inliers
RANSAC example: Translation

Putative matches
RANSAC example: Translation

Select one match, count inliers
RANSAC example: Translation

Select *one* match, count *inliers*
RANSAC example: Translation

Select translation with the most inliers
Scalability: Alignment to large databases

- What if we need to align a test image with thousands or millions of images in a model database?
  - Efficient putative match generation
    - Approximate descriptor similarity search, inverted indices
Large-scale visual search

Figure from: Kristen Grauman and Bastian Leibe, *Visual Object Recognition*, Synthesis Lectures on Artificial Intelligence and Machine Learning, April 2011, Vol. 5, No. 2, Pages 1-181
How to do the indexing?

- Idea: find a set of **visual codewords** to which descriptors can be **quantized**
Recall: Visual codebook for generalized Hough transform

Source: B. Leibe
K-means clustering

- Want to minimize sum of squared Euclidean distances between points $x_i$ and their nearest cluster centers $m_k$

$$D(X, M) = \sum_{\text{cluster } k} \sum_{\text{point } i \text{ in cluster } k} (x_i - m_k)^2$$

Algorithm:
- Randomly initialize K cluster centers
- Iterate until convergence:
  - Assign each data point to the nearest center
  - Recompute each cluster center as the mean of all points assigned to it
K-means demo

Source: http://shabal.in/visuals/kmeans/1.html
How to do the indexing?

- Cluster descriptors in the database to form codebook
- At query time, quantize descriptors in query image to nearest codevectors
- Problem solved?
Efficient indexing technique: Vocabulary trees

Hierarchical k-means clustering of descriptor space (vocabulary tree)
Vocabulary tree/inverted index
Populating the vocabulary tree/inverted index
Populating the vocabulary tree/inverted index

Model images
Populating the vocabulary tree/inverted index

Model images

Slide credit: D. Nister
Populating the vocabulary tree/inverted index

Model images

Slide credit: D. Nister
Looking up a test image

Slide credit: D. Nister
Cool application of large-scale alignment: searching the night sky

http://www.astrometry.net/