Image alignment
Image alignment: Motivation

Panorama stitching

Recognition of object instances
Image alignment: Challenges

Small degree of overlap

Occlusion, clutter
• **Two broad approaches:**
  • Direct (pixel-based) alignment
    – Search for alignment where most pixels agree
  • Feature-based alignment
    – Search for alignment where *extracted features* agree
    – Can be verified using pixel-based alignment
Alignment as fitting

- Last lecture: fitting a model to features in one image

Find model $M$ that minimizes

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

- Last lecture: fitting a model to features in one image

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

\[ \sum_i \text{residual}(x_i, M) \]

Find model \( M \) that minimizes the residual sum of squared differences between features and the model.

\[ \sum_i \text{residual}(T(x_i), x'_i) \]

Find transformation \( T \) that minimizes the residual sum of squared differences between transformed features and their matches.
Feature-based alignment outline
Feature-based alignment outline

- Extract features
Feature-based alignment outline

- Extract features
- Compute *putative matches*
Feature-based alignment outline

- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation $T$ (small group of putative matches that are related by $T$)
Feature-based alignment outline

- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation $T$ (small group of putative matches that are related by $T$)
  - *Verify* transformation (search for other matches consistent with $T$)
Feature-based alignment outline

• Extract features
• Compute *putative matches*
• Loop:
  • *Hypothesize* transformation $T$ (small group of putative matches that are related by $T$)
  • *Verify* transformation (search for other matches consistent with $T$)
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?

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

\[
\begin{bmatrix}
  x_i & y_i & 0 & 0 & 1 & 0 \\
  0 & 0 & x_i & y_i & 0 & 1 \\
  \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\end{bmatrix} \begin{bmatrix}
  m_1 \\
  m_2 \\
  m_3 \\
  m_4 \\
  t_1 \\
  t_2 \\
\end{bmatrix} = \begin{bmatrix}
  x'_i \\
  y'_i \\
  \cdots \\
\end{bmatrix}
\]
Fitting an affine transformation

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

- Linear system with six unknowns
- Each match gives us two linearly independent equations: need at least three to solve for the transformation parameters
What if we don’t know the correspondences?
What if we don’t know the correspondences?

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

• Assuming the patches are already normalized (i.e., the local effect of the geometric transformation is factored out), how do we compute their similarity?
• Want invariance to intensity changes, noise, perceptually insignificant changes of the pixel pattern
Feature descriptors

- Simplest descriptor: vector of raw intensity values
- How to compare two such vectors?
  - Sum of squared differences (SSD)
    \[
    \text{SSD}(u, v) = \sum_i (u_i - v_i)^2
    \]
    - Not invariant to intensity change
  - Normalized correlation
    \[
    \rho(u, v) = \frac{\sum_i (u_i - \overline{u})(v_i - \overline{v})}{\sqrt{\left(\sum_j (u_j - \overline{u})^2\right)\left(\sum_j (v_j - \overline{v})^2\right)}}
    \]
    - Invariant to affine intensity change
Feature descriptors

- Disadvantage of patches as descriptors:
  - Small shifts can affect matching score a lot

- Solution: histograms
Feature descriptors: SIFT

- Descriptor computation:
  - Divide patch into $4 \times 4$ 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
  • “Subdivide and disorder” strategy 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
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
  - Exhaustive search
    - For each feature in one image, compute the distance to *all* features in the other image and find the “closest” ones (threshold or fixed number of top matches)
  - Fast approximate nearest neighbor search
    - Hierarchical spatial data structures (kd-trees, vocabulary trees)
    - Hashing
Feature space outlier rejection

• How can we tell which putative matches are more reliable?

• Heuristic: compare distance of nearest neighbor to that of second nearest neighbor
  • Ratio will be high for features that are not distinctive
  • Threshold of 0.8 provides good separation

Dealing with outliers

• The set of putative matches still contains a very high percentage of outliers
• How do we fit a geometric transformation to a small subset of all possible matches?
• Possible strategies:
  • RANSAC
  • Incremental alignment
  • Hough transform
  • Hashing
Strategy 1: RANSAC

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

Find “average” translation vector
Problem with RANSAC

• In many practical situations, the percentage of outliers (incorrect putative matches) is often very high (90% or above)
• Alternative strategy: restrict search space by using strong locality constraints on seed groups and inliers
  • Incremental alignment
Strategy 2: Incremental alignment

- Take advantage of strong locality constraints: only pick close-by matches to start with, and gradually add more matches in the same neighborhood

Strategy 2: Incremental alignment

- Take advantage of strong locality constraints: only pick close-by matches to start with, and gradually add more matches in the same neighborhood
Strategy 2: Incremental alignment

• Take advantage of strong locality constraints: only pick close-by matches to start with, and gradually add more matches in the same neighborhood
Strategy 2: Incremental alignment

- Take advantage of strong locality constraints: only pick close-by matches to start with, and gradually add more matches in the same neighborhood
Strategy 2: Incremental alignment

• Take advantage of strong locality constraints: only pick close-by matches to start with, and gradually add more matches in the same neighborhood
Incremental alignment: Details

Generating seed groups:

- Identify triples of neighboring features \((i, j, k)\) in first image
- Find all triples \((i', j', k')\) in the second image such that \(i'\) (resp. \(j', k'\)) is a putative match of \(i\) (resp. \(j, k\)), and \(j', k'\) are neighbors of \(i'\)
Incremental alignment: Details

Beginning with each seed triple, repeat:

- Estimate the aligning transformation between corresponding features in current group of matches
- Grow the group by adding other consistent matches in the neighborhood

Until the transformation is no longer consistent or no more matches can be found
Beginning with each seed triple, repeat:

- Estimate the aligning transformation between corresponding features in current group of matches
- Grow the group by adding other consistent matches in the neighborhood

Until the transformation is no longer consistent or no more matches can be found
Beginning with each seed triple, repeat:

- Estimate the aligning transformation between corresponding features in current group of matches
- Grow the group by adding other consistent matches in the neighborhood

Until the transformation is no longer consistent or no more matches can be found
Incremental alignment: Details

Beginning with each seed triple, repeat:

• Estimate the aligning transformation between corresponding features in current group of matches
• Grow the group by adding other consistent matches in the neighborhood

Until the transformation is no longer consistent or no more matches can be found.
Strategy 3: Hough transform

- Suppose our features are scale- and rotation-invariant
  - Then a single feature match provides an alignment hypothesis (translation, scale, orientation)

Strategy 3: Hough transform

- Suppose our features are scale- and rotation-invariant
  - Then a single feature match provides an alignment hypothesis (translation, scale, orientation)
  - Of course, a hypothesis obtained from a single match is unreliable
  - Solution: let each match vote for its hypothesis in a Hough space with very coarse bins

Recall: Generalized Hough transform

B. Leibe, A. Leonardis, and B. Schiele, Combined Object Categorization and Segmentation with an Implicit Shape Model, ECCV Workshop on Statistical Learning in Computer Vision 2004
Hough transform details (D. Lowe’s system)

- **Training phase:** For each model feature, record 2D location, scale, and orientation of model (relative to normalized feature frame)

- **Test phase:** Let each match between a test and a model feature vote in a 4D Hough space
  - Use broad bin sizes of 30 degrees for orientation, a factor of 2 for scale, and 0.25 times image size for location
  - Vote for two closest bins in each dimension

- Find all bins with at least three votes and perform geometric verification
  - Estimate least squares *affine* transformation
  - Use stricter thresholds on transformation residual
  - Search for additional features that agree with the alignment

David G. Lowe. [*Distinctive image features from scale-invariant keypoints.*] (IJCV 60 (2), pp. 91-110, 2004.)
Image alignment: Review

• What is the bias/variance tradeoff?
• What are the advantages of a 2D affine alignment model?
• What are the main steps in feature-based alignment?
• How do we generate putative matches?
• How does the SIFT descriptor work?
• What are different computational strategies for estimating an alignment in the presence of outliers?
  • Strategy 1: RANSAC
  • Strategy 2: Incremental alignment
  • Strategy 3: Hough transform
  • Strategy 4: Hashing (next)
Strategy 4: Hashing

- Make each invariant image feature into a low-dimensional “key” that indexes into a table of hypotheses
Strategy 4: Hashing

- Make each invariant image feature into a low-dimensional “key” that indexes into a table of hypotheses.
- Given a new test image, compute the hash keys for all features found in that image, access the table, and look for consistent hypotheses.
Strategy 4: Hashing

- Make each invariant image feature into a low-dimensional “key” that indexes into a table of hypotheses.
- Given a new test image, compute the hash keys for all features found in that image, access the table, and look for consistent hypotheses.
- This can even work when we don’t have any feature descriptors: we can take n-tuples of neighboring features and compute invariant hash codes from their geometric configurations.
Application: Searching the sky

http://www.astrometry.net/
Beyond affine transformations

- What is the transformation between two views of a planar surface?

- What is the transformation between images from two cameras that share the same center?
Beyond affine transformations

• **Homography**: plane projective transformation (transformation taking a quad to another arbitrary quad)
Fitting a homography

- Recall: homogenous coordinates

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

Converting to homogenous image coordinates

\[\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow \left(\frac{x}{w}, \frac{y}{w}\right)\]

Converting from homogenous image coordinates
Fitting a homography

- Recall: homogenous coordinates

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

Converting to homogenous image coordinates

Converting from homogenous image coordinates

- Equation for homography:

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

- Equation for homography:

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

\[
\lambda x'_i = H x_i = \begin{bmatrix} h_1^T \\ h_2^T \\ h_3^T \end{bmatrix} x_i
\]

9 entries, 8 degrees of freedom (scale is arbitrary)

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

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

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

3 equations, only 2 linearly independent
Direct linear transform

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

- H has 8 degrees of freedom (9 parameters, but scale is arbitrary)
- One match gives us two linearly independent equations
- Four matches needed for a minimal solution (null space of 8x9 matrix)
- More than four: homogeneous least squares
Application: Panorama stitching
Recognizing panoramas

- Given contents of a camera memory card, automatically figure out which pictures go together and stitch them together into panoramas.

http://www.cs.ubc.ca/~mbrown/panorama/panorama.html
1. Estimate homography (RANSAC)
1. Estimate homography (RANSAC)
1. Estimate homography (RANSAC)
2. Find connected sets of images
2. Find connected sets of images
2. Find connected sets of images
3. Stitch and blend the panoramas
Results
Issues in alignment-based applications

• Choosing the geometric alignment model
  • Tradeoff between “correctness” and robustness (also, efficiency)

• Choosing the descriptor
  • “Rich” imagery (natural images): high-dimensional patch-based descriptors (e.g., SIFT)
  • “Impoverished” imagery (e.g., star fields): need to create invariant geometric descriptors from k-tuples of point-based features

• Strategy for finding putative matches
  • Small number of images, one-time computation (e.g., panorama stitching): brute force search
  • Large database of model images, frequent queries: indexing or hashing
  • Heuristics for feature-space pruning of putative matches
Issues in alignment-based applications

• Choosing the geometric alignment model
• Choosing the descriptor
• Strategy for finding putative matches
• Hypothesis generation strategy
  • Relatively large inlier ratio: RANSAC
  • Small inlier ratio: locality constraints, Hough transform
• Hypothesis verification strategy
  • Size of consensus set, residual tolerance depend on inlier ratio and expected accuracy of the model
  • Possible refinement of geometric model
  • Dense verification
Next time: Single-view geometry