Recap: Local Feature Matching Outline

1. Find a set of distinctive keypoints
2. Define a region around each keypoint
3. Extract and normalize the region content
4. Compute a local descriptor from the normalized region
5. Match local descriptors

Recap: Requirements for Local Features

- **Problem 1:**
  - Detect the same point independently in both images
- **Problem 2:**
  - For each point correctly recognize the corresponding one

Recap: Harris Detector [Harris88]

- **Compute second moment matrix (autocorrelation matrix)**
  \[
  M(\sigma_x, \sigma_y) = g(\sigma) \begin{bmatrix}
  I_x^2(\sigma_x) & I_x I_y(\sigma_x)
  
  I_x I_y(\sigma_x) & I_y^2(\sigma_x)
  
  \end{bmatrix}
  \]

1. Compute derivatives
2. Square of derivatives
3. Gaussian filter \( g(\sigma) \)
4. Cornerness function - two strong eigenvalues
   \[
   R = \text{det}(M(\sigma_x, \sigma_y)) - \text{trace}^2(M(\sigma_x, \sigma_y))
   = g(I_x^2) - 4g(I_x I_y)^2 - 4g(I_y^2)
   \]
5. Perform non-maximum suppression

Recap: Harris Detector Responses [Harris88]

- Effect: A very precise corner detector.
Recap: Hessian Detector [Beaudet78]

- Hessian determinant

\[
\text{Hessian}(I) = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix}
\]

\[
\det(\text{Hessian}(I)) = I_{xx}I_{yy} - I_{xy}^2
\]

In Matlab:

\[
\begin{bmatrix}
I_{xx} & I_{xy} \\
I_{xy} & I_{yy}
\end{bmatrix}
\]

Recap: Hessian Detector Responses [Beaudet78]

Effect: Responses mainly on corners and strongly textured areas.

Recap: Automatic Scale Selection

- Function responses for increasing scale (scale signature)

Recap: Laplacian-of-Gaussian (LoG)

- Interest points:
  - Local maxima in scale space of Laplacian-of-Gaussian

Recap: LoG Detector Responses

- Efficient implementation
  - Approximate LoG with a difference of Gaussians

Recap: Key point localization with DoG

- Approach DoG Detector
  - Detect maxima of difference-of-Gaussian (DoG) in scale space
  - Reject points with low contrast (threshold)
  - Eliminate edge responses
Recap: Harris-Laplace [Mikolajczyk ’01]

1. Initialization: Multiscale Harris corner detection
2. Scale selection based on Laplacian
   (same procedure with Hessian ⇒ Hessian-Laplace)

Recap: SIFT Feature Descriptor

- Scale Invariant Feature Transform
- Descriptor computation:
  - Divide patch into $4 \times 4$ sub-patches: 16 cells
  - Compute histogram of gradient orientations (8 reference angles)
    for all pixels inside each sub-patch
  - Resulting descriptor: $4 \times 4 \times 8 = 128$ dimensions

Topics of This Lecture

- Local Feature Extraction (cont’d)
  - Affine Invariant Feature Extraction
  - Uses of Local Features
- Recognition with Local Features
  - Matching local features
  - Finding consistent configurations
  - Alignment: linear transformations
  - Affine estimation
  - Homography estimation
- Dealing with Outliers
  - RANSAC
  - Incremental Alignment
  - Generalized Hough Transform

The Need for Invariance

- Up to now, we had invariance to
  - Translation
  - Scale
  - Rotation
- Not sufficient to match regions under viewpoint changes
  - For this, we need also affine adaptation

Affine Adaptation

- Problem:
  - Determine the characteristic shape of the region.
  - Assumption: shape can be described by “local affine frame”.
- Solution: iterative approach
  - Use a circular window to compute second moment matrix
  - Perform affine adaptation to find an ellipse-shaped window
  - Recompute second moment matrix using new window and iterate

Iterative Affine Adaptation

1. Detect keypoints, e.g. multi-scale Harris
2. Automatically select the scales
3. Adapt affine shape based on second order moment matrix
4. Refine point location

Affine Normalization/Deskewing

- Steps
  - Rotate the ellipse’s main axis to horizontal
  - Scale the x axis, such that it forms a circle

Slide credit: Tinne Tuytelaars

Affine Adaptation Example

Scale-invariant regions (blobs)

Slide credit: Svetlana Lazebnik

Affine Adaptation Example

Affine-adapted blobs

Slide credit: Svetlana Lazebnik

Summary: Affine-Inv. Feature Extraction

- Extract affine regions
- Normalize regions
- Eliminate rotational ambiguity

Slide credit: Svetlana Lazebnik, David Lowe

Invariance vs. Covariance

- Invariance:
  \[ \text{features(transform(image))} = \text{features(image)} \]
- Covariance:
  \[ \text{features(transform(image))} = \text{transform(features(image))} \]

Slide credit: Svetlana Lazebnik, Kristen Grauman

Applications of Local Invariant Features

- Wide baseline stereo
- Motion tracking
- Panoramas
- Mobile robot navigation
- 3D reconstruction
- Recognition
  - Specific objects
  - Textures
  - Categories
  - ...
### Wide-Baseline Stereo

- Image from T. Tuytelaars ECCV 2006 tutorial

### Automatic Mosaicing


### Panorama Stitching

- [Brown, Szeliski, and Winder, 2005]

### Recognition of Specific Objects, Scenes

- Schmid and Mohr 1997
- Sivic and Zisserman, 2003
- Rothganger et al. 2003

### Recognition of Categories

- Constellation model
- Bags of words

- Weber et al. (2000)
- Perona et al. (2001)

- Csurka et al. (2004)
- Csurka et al. (2005)
- Sivic et al. (2003)

- Lazebnik et al. (2004)

### Value of Local Features

- Critical to find distinctive and repeatable local regions for multi-view matching.
- Complexity reduction via selection of distinctive points.
- Describe images, objects, parts without requiring segmentation; robustness to clutter & occlusion.
- Robustness: similar descriptors in spite of moderate view changes, noise, blur, etc.
Topics of This Lecture

- Local Feature Extraction (cont’d)
  - Affine Invariant Feature Extraction
  - Uses of Local Features
- Recognition with Local Features
  - Matching local features
  - Finding consistent configurations
  - Alignment: linear transformations
  - Affine estimation
  - Homography estimation
- Dealing with Outliers
  - Manual
  - Incremental Alignment
  - Generalized Hough Transform

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.

- Options
  - 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 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

Recognition with Local Features

- Image content is transformed into local features that are invariant to translation, rotation, and scale
- Goal: Verify if they belong to a consistent configuration

Parametric (Global) Warping

- Transformation \( T \) is a coordinate-changing machine:
  \[ p^* = T(p) \]
- What does it mean that \( T \) is global?
  - It’s the same for any point \( p \)
  - It can be described by just a few numbers (parameters)
- Let’s represent \( T \) as a matrix:
  \[ \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \]
Perceptual and Sensory Augmented Computing

2D Linear Transforms

\[
\begin{pmatrix}
  x' \\
  y'
\end{pmatrix} =
\begin{pmatrix}
  a & b \\
  c & d
\end{pmatrix}
\begin{pmatrix}
  x \\
  y
\end{pmatrix}
\]

- Only linear 2D transformations can be represented with a 2x2 matrix.
- Linear transformations are combinations of ...
  - Scale, Rotation, Shear, and Mirror

Homogeneous Coordinates

- Q: How can we represent translation as a 3x3 matrix using homogeneous coordinates?
  \[
  \begin{pmatrix}
  x' = x + t_x \\
  y' = y + t_y
  \end{pmatrix}
  \]
- A: Using the rightmost column:
  \[
  \text{Translation} =
  \begin{pmatrix}
    1 & 0 & t_x \\
    0 & 1 & t_y
  \end{pmatrix}
  \]

Basic 2D Transformations

- Basic 2D transformations as 3x3 matrices

\[
\begin{pmatrix}
  x' \\
  y' \\
  1
\end{pmatrix} =
\begin{pmatrix}
  1 & 0 & t_x \\
  0 & 1 & t_y \\
  0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
  x \\
  y \\
  1
\end{pmatrix}
\]
- Affine transformations are combinations of ...
  - Linear transformations, and Translations
- Parallel lines remain parallel

2D Affine Transformations

\[
\begin{pmatrix}
  x' \\
  y' \\
  w'
\end{pmatrix} =
\begin{pmatrix}
  a & b & c \\
  d & e & f \\
  0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
  x \\
  y \\
  w
\end{pmatrix}
\]

**Projective Transformations**

\[
\begin{bmatrix}
  x' \\
  y' \\
  w'
\end{bmatrix} =
\begin{bmatrix}
  a & b & c \\
  d & e & f \\
  g & h & i
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  w
\end{bmatrix}
\]

- Projective transformations:
  - Affine transformations, and
  - Projective warps
- Parallel lines do not necessarily remain parallel

**Alignment Problem**

- We have previously considered how to fit a model to image evidence
  - e.g., a line to edge points, or a snake to a deforming contour
- In alignment, we will fit the parameters of some transformation according to a set of matching feature pairs ("correspondences").

**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**

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

**Recall: Least Squares Estimation**

- Set of data points: \((X_1, X_2, X_3)\)
- Goal: a linear function to predict \(X'\)’s from \(X\)’s:
  \[ Xa + b = X \]
- We want to find \(a\) and \(b\).
- How many \((X, X')\) pairs do we need?
  \[
  X_1a + b = X_1' \\
  X_2a + b = X_2'
  \]
- What if the data is noisy?

\[
\begin{bmatrix}
  X_1 & 1 & | & X_1' \\
  X_2 & 1 & | & X_2'
\end{bmatrix}
\]

Matlab:

\[
\begin{bmatrix}
  X_1 \\
  X_2
\end{bmatrix}
\begin{bmatrix}
  a \\
  b
\end{bmatrix} = \begin{bmatrix}
  X_1' \\
  X_2'
\end{bmatrix} \
\Rightarrow \text{Least-squares minimization}
\]

\[
Ax = B
\]

\[
\begin{bmatrix}
  X_1 & 1 \\
  X_2 & 1 \\
  X_3 & 1 \\
  \vdots & \vdots
\end{bmatrix}
\begin{bmatrix}
  a \\
  b \\
  c
\end{bmatrix} = \begin{bmatrix}
  X_1' \\
  X_2' \\
  X_3'
\end{bmatrix} \
\Rightarrow \text{Least-squares minimization}
\]

\[
\begin{bmatrix}
  X_1 \\
  X_2 \\
  X_3 \\
  \vdots
\end{bmatrix}
\begin{bmatrix}
  a \\
  b \\
  c
\end{bmatrix} = \begin{bmatrix}
  X_1' \\
  X_2' \\
  X_3'
\end{bmatrix}
\]
Fitting an Affine Transformation

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

\[
\begin{bmatrix}
  x' & y' \\
\end{bmatrix} =
\begin{bmatrix}
  m_1 & m_2 & m_3 \\
  m_4 & m_5 & m_6 \\
  m_7 & m_8 & m_9 \\
\end{bmatrix}
\begin{bmatrix}
  x & y & 1 \\
\end{bmatrix}
\]

Fitting a Homography

- Estimating the transformation

\[
\begin{bmatrix}
  x' & y' \\
\end{bmatrix} =
\begin{bmatrix}
  h_1 & h_2 & h_3 & 0 \\
  h_4 & h_5 & h_6 & 0 \\
  h_7 & h_8 & h_9 & 0 \\
  0 & 0 & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
  x & y & 1 \\
\end{bmatrix}
\]

Properties

- Rectangle should map to arbitrary quadrilateral
- Parallel lines aren't but must preserve straight lines
- This is called a homography

Set scale factor to 1 ⇒ 8 parameters left.

Homography

- A projective transform is a mapping between any two perspective projections with the same center of projection.
- i.e. two planes in 3D along the same sight ray
- Properties
  - Parallel lines aren’t but must preserve straight lines
  - This is called a homography

Fitting a Homography

- Estimating the transformation
Fitting a Homography

- Estimating the transformation

\[
x' = Hx
\]

Matrix notation

\[
\begin{bmatrix}
    x' \\
    y' \\
    1
\end{bmatrix} =
\begin{bmatrix}
    a_{11} & a_{12} & a_{13} \\
    a_{21} & a_{22} & a_{23} \\
    0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
    x \\
    y \\
    1
\end{bmatrix}
\]

Image coordinates

Homogenous coordinates

[Slide credit: Krystian Mikolajczyk]
Image Warping with Homographies

- Image plane in front
- Image plane below
- Black area where no pixel maps to
- $p'$

Slide credit: Steve Seitz

Uses: Analyzing Patterns and Shapes

- What is the shape of the b/w floor pattern?

Slide credit: Antonio Criminisi

Analyzing Patterns and Shapes

- From Martin Kemp. The Science of Art (manual reconstruction)

Slide credit: Antonio Criminisi

Summary: Recognition by Alignment

- Basic matching algorithm
  1. Detect interest points in two images.
  2. Extract patches and compute a descriptor for each one.
  3. Compare one feature from image 1 to every feature in image 2 and select the nearest-neighbor pair.
  4. Repeat the above for each feature from image 1.
  5. Use the list of best pairs to estimate the transformation between images.

- Transformation estimation
  - Affine
  - Homography

Topics of This Lecture

- Local Feature Extraction (cont’d)
  - Affine invariant feature extraction
  - Uses of Local Features
- Recognition with Local Features
  - Matching local features
  - Finding consistent configurations
  - Alignment using transformation
  - Affine estimation
  - Homography estimation
- Dealing with Outliers
  - RANSAC
  - Incremental Alignment
  - Generalized Hough Transform

Problem: Outliers

- Outliers can hurt the quality of our parameter estimates, e.g.,
  1. An erroneous pair of matching points from two images
  2. A feature point that is noise or doesn’t belong to the transformation we are fitting.
**Example: Least-Squares Line Fitting**

- Assuming all the points that belong to a particular line are known

**Outliers Affect Least-Squares Fit**

**Strategy 1: RANSAC** [Fischler81]

- **Random Sample Consensus**
  - Approach: we want to avoid the impact of outliers, so let’s look for “inliers”, and use those only.
  - Intuition: if an outlier is chosen to compute the current fit, then the resulting line won’t have much support from rest of the points.

**RANSAC**

**RANSAC loop:**

1. Randomly select a seed group of points on which to base transformation estimate (e.g., a 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 Line Fitting Example**

- Task: Estimate the best line
RANSAC Line Fitting Example

- Task: Estimate the best line

Sample two points

Fit a line to them

Total number of points within a threshold of line.

Repeat, until we get a good result.

RANSAC: How many samples?

- How many samples are needed?
  - Suppose $w$ is fraction of inliers (points from line).
  - $n$ points needed to define hypothesis (2 for lines)
  - $k$ samples chosen.
  - Prob. that a single sample of $n$ points is correct: $w^n$
  - Prob. that all samples fail: $(1 - w)^k$

⇒ Choose $k$ high enough to keep this below desired failure rate.

RANSAC: Computed $k$ (p=0.99)

<table>
<thead>
<tr>
<th>Sample size</th>
<th>5%</th>
<th>10%</th>
<th>20%</th>
<th>25%</th>
<th>30%</th>
<th>40%</th>
<th>50%</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>11</td>
<td>17</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>7</td>
<td>9</td>
<td>11</td>
<td>19</td>
<td>35</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>5</td>
<td>9</td>
<td>13</td>
<td>17</td>
<td>34</td>
<td>72</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>6</td>
<td>12</td>
<td>17</td>
<td>26</td>
<td>57</td>
<td>146</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>7</td>
<td>16</td>
<td>24</td>
<td>37</td>
<td>97</td>
<td>293</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>8</td>
<td>20</td>
<td>33</td>
<td>54</td>
<td>163</td>
<td>588</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>9</td>
<td>26</td>
<td>44</td>
<td>78</td>
<td>272</td>
<td>1177</td>
</tr>
</tbody>
</table>
**After RANSAC**

- RANSAC divides data into inliers and outliers and yields estimate computed from minimal set of inliers.
- Improve this initial estimate with estimation over all inliers (e.g. with standard least-squares minimization).
- But this may change inliers, so alternate fitting with re-classification as inlier/outlier.

---

**Example: Finding Feature Matches**

- Find best stereo match within a square search window (here 300 pixels$^2$)
- Global transformation model: epipolar geometry

Before RANSAC | After RANSAC
--- | ---

---

**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.

---

**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.
**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).

**Pose Clustering and Verification with SIFT**

- To detect instances of objects from a model base:
  1. Index descriptors
     - Distinctive features narrow down possible matches

**Indexing Local Features**

- Model base
- New Image
- Model base
Pose Clustering and Verification with SIFT

- To detect instances of objects from a model base:
  1. Index descriptors
     - Distinctive features narrow down possible matches
  2. Generalized Hough transform to vote for poses
     - Keypoints have record of parameters relative to model coordinate system
  3. Affine fit to check for agreement between model and image features
     - Fit and verify using features from Hough bins with 3+ votes

Object Recognition Results

- Background subtract for model boundaries
- Objects recognized
- Recognition in spite of occlusion

Recall: Difficulties of Voting

- Noise/clutter can lead to as many votes as true target.
- Bin size for the accumulator array must be chosen carefully.
- (Recall Hough Transform)
- In practice, good idea to make broad bins and spread votes to nearby bins, since verification stage can prune bad vote peaks.

Applications: Specific Object Recognition

- Sony Aibo (Evolution Robotics)
  - SIFT usage
    - Recognize docking station
    - Communicate with visual cards
- Commercial services coming out...
  - 30’000 movie posters indexed
  - Query-by-image from mobile phone available in Germany and Switzerland
Applications: Tourist Guide

Mobile tourist guide
- Self-localization
- Object/building recognition
- Photo/video augmentation

Application: Large-Scale Retrieval

Query: Results from 5k Flickr images (demo available for 10k set)
http://www.robots.ox.ac.uk/~vgg/research/oxbuildings/index.html

Application: Image Auto-Annotation

Left: Wikipedia image
Right: closest match from Flickr

Summary
- Recognition by alignment: looking for object and pose that fits well with image
  - Use good correspondences to designate hypotheses.
  - Invariant local features offer more reliable matches.
  - Find consistent “inlier” configurations in clutter
    - Generalized Hough Transform
    - RANSAC
- Alignment approach to recognition can be effective if we find reliable features within clutter.
  - Application: large-scale image retrieval
  - Application: recognition of specific (mostly planar) objects
    - Movie posters
    - Books
    - CD covers

References and Further Reading

- More details on RANSAC and homography estimation can be found in Chapter 4.7 of
  - R. Hartley, A. Zisserman
    Multiple View Geometry in Computer Vision
    2nd Ed., Cambridge Univ. Press, 2004
- Details about the Hough transform for object recognition can be found in
  - D. Lowe, Distinctive image features from scale-invariant keypoints.
    IJCV 60(2), pp. 91-110, 2004
- Try the Oxford image retrieval demo
  - http://www.robots.ox.ac.uk/~vgg/research/oxbuildings/index.html
- Try the available local feature detectors and descriptors
  - http://www.robots.ox.ac.uk/~vgg/research/affine/detectors.html#binaries