Recap: 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

Recap: Fitting an Affine Transformation
- Assuming we know the correspondences, how do we get the transformation?

Recap: Fitting a Homography
- Estimating the transformation

Course Outline
- Image Processing Basics
- Segmentation & Grouping
- Object Recognition & Categorization
  - Sliding Window based Object Detection
- Local Features & Matching
  - Local Features – Detection and Description
  - Recognition with Local Features
- Deep Learning
- 3D Reconstruction
Recap: Fitting a Homography

- Estimating the transformation
- Solution:
  - Null-space vector of $A$
  - Corresponds to smallest eigenvector
- Singular vectors
- Equations of the form $Ax = 0$
- Minimizes least square error

Recap: A General Point

- Equations of the form $Ax = 0$
- How do we solve them? (always!)
  - Apply SVD
  - $SVD \quad A = UDV^T$
  - Singular values, singular vectors
  - Singular values of $A$ = square roots of the eigenvalues of $A^TA$.
  - The solution of $Ax = 0$ is the nullspace vector of $A$.
  - This corresponds to the smallest singular vector of $A$.

Recap: Object Recognition by Alignment

- Assumption
  - Known object, rigid transformation compared to model image
  - If we can find evidence for such a transformation, we have recognized the object.
- You learned methods for
  - Fitting an affine transformation from ≥ 3 correspondences
  - Fitting a homography from ≥ 4 correspondences
- Affine: solve a system $At = b$
- Homography: solve a system $Ah = 0$
- Correspondences may be noisy and may contain outliers
  - Need to use robust methods that can filter out outliers

Topics of This Lecture

- Recap: Recognition with Local Features
- Dealing with Outliers
  - RANSAC
  - Generalized Hough Transform
- Deep Learning
  - Motivation
  - Neural Networks
- Convolutional Neural Networks
  - Convolutional Layers
  - Pooling Layers
  - Nonlinearities

Problem: Outliers

- Outliers can hurt the quality of our parameter estimates,
  - e.g.,
  - An erroneous pair of matching points from two images
  - 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

- Outliers affect the least-squares fit, resulting in a poor estimate.

Strategy 1: RANSAC [Fischler81]

- RANdom SAmple Consensus
- Approach: we want to avoid the impact of outliers, so let's look for "inliers", and use only those.
- 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
  - How many points do we need to estimate the 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.

“7 inlier points”

Repeat, until we get a good result.

“11 inlier points”

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 \( k \) samples fail is: \( (1 - w^n)^k \)

\[ \Rightarrow \text{Choose } k \text{ high enough to keep this below the desired failure rate.} \]

RANSAC: Computed \( k \) (p=0.99)

<table>
<thead>
<tr>
<th>Sample size ( n )</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

Example: Finding Feature Matches

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

Problem with RANSAC

- In many practical situations, the percentage of outliers (incorrect putative matches) is often very high (90% or above).
- Alternative strategy: Generalized Hough Transform
Strategy 2: Generalized 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 from a single match is unreliable.
Solution: let each match vote for its hypothesis in a Hough space with very coarse bins.

Topics of This Lecture

- Recap: Recognition with Local Features
- Dealing with Outliers
  - RANSAC
  - Generalized Hough Transform
- Deep Learning
  - Motivation
  - Neural Networks
- Convolutional Neural Networks
  - Convolutional Layers
  - Pooling Layers
  - Nonlinearities

We’ve finally got there!

Traditional Recognition Approach

- Characteristics
  - Features are not learned, but engineered
  - Trainable classifier is often generic (e.g., SVM)
  - Many successes in 2000-2010.
What About Learning the Features?

- **Learn a feature hierarchy** all the way from pixels to classifier
  - Each layer extracts features from the output of previous layer
  - Train all layers jointly

“Shallow” vs. “Deep” Architectures

**Traditional recognition: “Shallow” architecture**

- Image/Video Pixels ➔ Hand-designed feature extraction ➔ Trainable classifier ➔ Object Class

**Deep learning: “Deep” architecture**

- Image/Video Pixels ➔ Layer 1 ➔ ... ➔ Layer N ➔ Simple classifier ➔ Object Class

Background: Perceptrons

- **Input**
  - $x_1, x_2, ..., x_d$

- **Weights**
  - $w_1, w_2, ..., w_d$

- **Output:** $\sigma(w \cdot x + b)$
  - Sigmoid function

  $\sigma(t) = \frac{1}{1 + e^{-t}}$

Inspiration: Neuron Cells

**Hubel/Wiesel Architecture**

- Visual cortex consists of a hierarchy of simple, complex, and hyper-complex cells

- **Hubel & Weisel topographical mapping**
  - Hyper-complex cells
  - Complex cells
  - Simple cells

- **featural hierarchy**
  - High level
  - Mid level
  - Low level

Background: Multi-Layer Neural Networks

- **Nonlinear classifier**
  - **Training:** find network weights $w$ to minimize the error between true training labels $t_n$ and estimated labels $f_w(x_n)$:

  $E(W) = \sum L(t_n, f(x_n; W))$

  - Minimization can be done by gradient descent, provided $f$ is differentiable
  - Training method: Error backpropagation.
Convolutional Neural Networks (CNN, ConvNet)

- Neural network with specialized connectivity structure
  - Stack multiple stages of feature extractors
  - Higher stages compute more global, more invariant features
  - Classification layer at the end


Topics of This Lecture

- Recap: Recognition with Local Features
- Dealing with Outliers
  - RANSAC
  - Generalized Hough Transform
- Deep Learning
  - Motivation
  - Neural Networks
- Convolutional Neural Networks
  - Convolutional Layers
  - Pooling Layers
  - Nonlinearities

Convolutional Networks: Structure

- Feed-forward feature extraction
  1. Convolve input with learned filters
  2. Non-linearity
  3. Spatial pooling
  4. (Normalization)
- Supervised training of convolutional filters by back-propagating classification error

Convolutional Networks: Intuition

- Locally connected net
  - E.g. 1000×1000 image
  - 1M hidden units
  - 10×10 receptive fields
  - 100M parameters!

- Ideas to improve this
  - Spatial correlation is local
  - Want translation invariance

- Convolutional net
  - Share the same parameters across different locations
  - Convolutions with learned kernels
Convolutional Networks: Intuition

- **Convolutional net**
  - Share the same parameters across different locations
  - Convolutions with learned kernels

**Learn multiple filters**
- E.g. 1000×1000 image
  - 100 filters
  - 10×10 filter size
  - \(10k\) parameters

- **Result: Response map**
  - size: 1000×1000×100
  - Only memory, not params!

**Convolution Layers**

- Example image: 32×32×3 volume

**Before:** Full connectivity
  - 32×32×3 weights

**Now:** Local connectivity
  - One neuron connects to, e.g., 5×5×3 region.
  - \(5\times5\times3\) shared weights.

**Convolution Layers**

- All Neural Net activations arranged in 3 dimensions
  - Multiple neurons all looking at the same input region, stacked in depth

**Naming convention:**
- \(H × W \times D\)
  - \(H\): Height (rows)
  - \(W\): Width (columns)
  - \(D\): Depth (channels)

**Convolution Layers**

- All Neural Net activations arranged in 3 dimensions
  - Convolution layers can be stacked
  - The filters of the next layer then operate on the full activation volume.
  - Filters are local in \((x,y)\), but densely connected in depth.
Activation Maps of Convolutional Filters

Each activation map is a depth slice through the output volume.

Convolution Layers

- Replicate this column of hidden neurons across space, with some \textit{stride}.

Example:

- 7 \times 7 input
- assume 3 \times 3 connectivity
- stride 1

\Rightarrow 5 \times 5 output
Convolution Layers

• Replicate this column of hidden neurons across space, with some stride.

Example: 7x7 input assume 3x3 connectivity stride 1 ⇒ 5x5 output
What about stride 2?

B. Leibe
Slide credit: FeiFei Li, Andrej Karpathy

Commonly Used Nonlinearities

• Sigmoid
g(α) = \frac{1}{1+\exp(-α)}

• Hyperbolic tangent
g(α) = \tanh(α) = 2α/(2α + 1)

• Rectified linear unit (ReLU)
g(α) = \max\{0, α\}

Preferred option for deep networks

B. Leibe
Slide credit: Yann LeCun
Convolutional Networks: Intuition

- Let’s assume the filter is an eye detector
  - How can we make the detection robust to the exact location of the eye?

Solution:
- By pooling (e.g., max or avg) filter responses at different spatial locations, we gain robustness to the exact spatial location of features.

Max Pooling

- Effect:
  - Make the representation smaller without losing too much information
  - Achieve robustness to translations

Note
- Pooling happens independently across each slice, preserving the number of slices.

Compare: SIFT Descriptor

- Lowe [IJCV 2004]
- Apply oriented filters
- Spatial pool (Sum)
- Normalize to unit length
- Feature Vector

Compare: Spatial Pyramid Matching

- Lazebrk, Schmidt, Ponce [CVPR 2006]
- Filter with Visual Words
- Take max V/W response (L-inf normalization)
- Multi-scale spatial pool (Sum)
- Global image descriptor
References and Further Reading

• More information on Deep Learning and CNNs can be found in Chapters 6 and 9 of the Goodfellow & Bengio book.

I. Goodfellow, Y. Bengio, A. Courville
Deep Learning
MIT Press, 2016
http://www.deeplearningbook.org/