Computer Vision – Lecture 10

Deep Learning

27.05.2019

Bastian Leibe
Visual Computing Institute
RWTH Aachen University
http://www.vision.rwth-aachen.de/

leibe@vision.rwth-aachen.de
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
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
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.

Local Features, e.g. SIFT

Slide credit: David Lowe
Recap: Fitting an Affine Transformation

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

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

- Estimating the transformation

\[
\begin{align*}
\mathbf{x}_{A1} & \leftrightarrow \mathbf{x}_{B1} \\
\mathbf{x}_{A2} & \leftrightarrow \mathbf{x}_{B2} \\
\mathbf{x}_{A3} & \leftrightarrow \mathbf{x}_{B3} \\
\vdots & \quad \vdots \\

x_{A1} &= \frac{h_{11} x_{B1} + h_{12} y_{B1} + h_{13}}{h_{31} x_{B1} + h_{32} y_{B1} + 1} \\
y_{A1} &= \frac{h_{21} x_{B1} + h_{22} y_{B1} + h_{23}}{h_{31} x_{B1} + h_{32} y_{B1} + 1}
\end{align*}
\]

Homogenous coordinates

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

Image coordinates

\[
\begin{bmatrix}
x'' \\
y'' \\
1
\end{bmatrix}
= \begin{bmatrix}
1 \\
y'
1
\end{bmatrix}
\]

Matrix notation

\[
x' = Hx
\]

\[
x'' = \frac{1}{z'} x'
\]

Slide credit: Krystian Mikolajczyk

B. Leibe
Recap: Fitting a Homography

- Estimating the transformation

\[
\begin{align*}
    h_{11} x_{B_1} + h_{12} y_{B_1} + h_{13} - x_{A_i} h_{31} x_{B_1} - x_{A_i} h_{32} y_{B_1} - x_{A_i} &= 0 \\
    h_{21} x_{B_1} + h_{22} y_{B_1} + h_{23} - y_{A_i} h_{31} x_{B_1} - y_{A_i} h_{32} y_{B_1} - y_{A_i} &= 0
\end{align*}
\]

\[
\begin{bmatrix}
    x_{B_1} & y_{B_1} & 1 & 0 & 0 & 0 & -x_{A_i} x_{B_1} & -x_{A_i} y_{B_1} & -x_{A_i} \\
    0 & 0 & 0 & x_{B_1} & y_{B_1} & 1 & -y_{A_i} x_{B_1} & -y_{A_i} y_{B_1} & -y_{A_i}
\end{bmatrix}
\begin{bmatrix}
    h_{11} \\
    h_{12} \\
    h_{13} \\
    h_{21} \\
    h_{22} \\
    h_{23} \\
    h_{31} \\
    h_{32} \\
    1
\end{bmatrix} = \begin{bmatrix}
    0 \\
    0 \\
    . \\
    . \\
    . \\
    . \\
    . \\
    . \\
    .
\end{bmatrix}
\]

\[Ah = 0\]
Recap: Fitting a Homography

- Estimating the transformation
- Solution:
  - Null-space vector of $A$
  - Corresponds to smallest eigenvector

Solution:

\[ Ah = 0 \]

\[ A = UDV^T = U \begin{bmatrix} d_{11} & \cdots & d_{19} \\ \vdots & \ddots & \vdots \\ d_{91} & \cdots & d_{99} \end{bmatrix} \begin{bmatrix} v_{11} & \cdots & v_{19} \\ \vdots & \ddots & \vdots \\ v_{91} & \cdots & v_{99} \end{bmatrix}^T \]

\[ h = [v_{19}, \cdots, v_{99}] \]

Minimizes least square error

Slide credit: Krystian Mikolajczyk
Recap: A General Point

• Equations of the form

\[ Ax = 0 \]

• How do we solve them? (always!)
  
  ➢ Apply SVD

  SVD

  \[ A = UDV^T = U \begin{bmatrix} d_{11} & \cdots & \cdot \\ \cdot & \ddots & \cdot \\ \cdot & \cdot & d_{NN} \end{bmatrix} \begin{bmatrix} v_{11} & \cdots & v_{1N} \\ \vdots & \ddots & \vdots \\ v_{N1} & \cdots & v_{NN} \end{bmatrix}^T \]

  Singular values  Singular vectors

  ➢ Singular values of \( A \) = square roots of the eigenvalues of \( A^TA \).
  ➢ The solution of \( Ax=0 \) is the \textit{nullspace} vector of \( A \).
  ➢ This corresponds to the \textit{smallest singular vector} of \( A \).

  Think of this as an eigenvector equation \( Ax = \lambda x \) for the special case of \( \lambda = 0 \).

  SVD is the generalization of the eigenvector decomposition for non-square matrices \( A \).
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 \( \geq 3 \) correspondences
  - Fitting a *homography* from \( \geq 4 \) correspondences

\[
\text{Affine: solve a system} \quad A t = b \\
\text{Homography: solve a system} \quad A h = 0
\]

• 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.

Slide credit: Kristen Grauman
Example: Least-Squares Line Fitting

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

Source: Forsyth & Ponce
Outliers Affect Least-Squares Fit
Outliers Affect Least-Squares Fit
Strategy 1: RANSAC [Fischler81]

- **RAN**dom **SA**mple **C**onsensus

- 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

Slide credit: Jinxiang Chai

B. Leibe
RANSAC Line Fitting Example

- Task: Estimate the best line

Fit a line to them
RANSAC Line Fitting Example

• Task: Estimate the best line

Total number of points within a threshold of line.

Slide credit: Jinxiang Chai

B. Leibe
RANSAC Line Fitting Example

• Task: Estimate the best line

Total number of points within a threshold of line.

“7 inlier points”
RANSAC Line Fitting Example

- Task: Estimate the best line

Repeat, until we get a good result.

Slide credit: Jinxiang Chai
RANSAC Line Fitting Example

• Task: Estimate the best line

Repeat, until we get a good result.

“11 inlier points”

Slide credit: Jinxiang Chai
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: \( (1 - w^n)^k \)

⇒ Choose \( k \) high enough to keep this below the desired failure rate.
<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.

Slide credit: David Lowe
Example: Finding Feature Matches

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

Images from Hartley & Zisserman

Slide credit: David Lowe
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

Images from Hartley & Zisserman

Slide credit: David Lowe
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).
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.

[Model of a toy train with detected features highlighted.]
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!

Deep Learning
Traditional Recognition Approach

- **Characteristics**
  - Features are not learned, but engineered
  - Trainable classifier is often generic (e.g., SVM)
  - Many successes in 2000-2010.
Traditional Recognition Approach

- Features are key to recent progress in recognition
  - Multitude of hand-designed features currently in use
  - SIFT, HOG, ...........

$\Rightarrow$ Where next? Better classifiers? Or keep building more features?

DPM
[Felzenszwalb et al., PAMI’07]

Dense SIFT+LBP+HOG $\rightarrow$ BOW $\rightarrow$ Classifier
[Yan & Huan ‘10] (Winner of PASCAL 2010 Challenge)
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

Slide credit: Svetlana Lazebnik
“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

Slide credit: Svetlana Lazebnik
Background: Perceptrons

Input

Weights

\[ x_1 \]
\[ x_2 \]
\[ x_3 \]
\[ \vdots \]
\[ x_d \]

\[ w_1 \]
\[ w_2 \]
\[ w_3 \]
\[ \vdots \]
\[ w_d \]

Output: \[ \sigma(w \cdot x + b) \]

Sigmoid function

\[ \sigma(t) = \frac{1}{1 + e^{-t}} \]
Inspiration: Neuron Cells

Slide credit: Svetlana Lazebnik, Rob Fergus
Background: Multi-Layer Neural Networks

- Nonlinear classifier
  - Training: find network weights $\mathbf{w}$ to minimize the error between true training labels $t_n$ and estimated labels $f_{\mathbf{w}}(x_n)$:
    $$E(\mathbf{w}) = \sum L(t_n, f(x_n; \mathbf{W}))$$
  - Minimization can be done by gradient descent, provided $f$ is differentiable
    - Training method: Error backpropagation.

Slide credit: Svetlana Lazebnik
Hubel/Wiesel Architecture

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


Slide credit: Svetlana Lazebnik
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

Slide credit: Svetlana Lazebnik
Convolutional Networks: Intuition

- Fully connected network
  - E.g. 1000×1000 image
  - 1M hidden units
  \[ \implies 1T \text{ parameters!} \]

- Ideas to improve this
  - Spatial correlation is local

Slide adapted from Marc'Aurelio Ranzato

B. Leibe

Image source: Yann LeCun
Convolutional Networks: Intuition

- Locally connected net
  - E.g. $1000 \times 1000$ image
  - 1M hidden units
  - $10 \times 10$ receptive fields
  - $\Rightarrow 100M$ parameters!

- Ideas to improve this
  - Spatial correlation is local
  - Want translation invariance
Convolutional Networks: Intuition

- **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!
Important Conceptual Shift

• Before

• Now:
Convolution Layers

Example
image: $32 \times 32 \times 3$ volume

**Before**: Full connectivity
$32 \times 32 \times 3$ weights

**Now**: Local connectivity
One neuron connects to, e.g., $5 \times 5 \times 3$ region.
⇒ Only $5 \times 5 \times 3$ shared weights.

- **Note**: Connectivity is
  - Local in space  ($5 \times 5$ inside $32 \times 32$)
  - But full in depth (all 3 depth channels)
Convolution Layers

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

- All Neural Net activations arranged in 3 dimensions
  - Multiple neurons all looking at the same input region, stacked in depth
  - Form a single $[1 \times 1 \times \text{depth}]$ depth column in output volume.

Naming convention:

Slide credit: FeiFei Li, Andrej Karpathy
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.

Slide adapted from: FeiFei Li, Andrej Karpathy
Activation Maps of Convolutional Filters

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

5×5 filters

Activation maps

Slide adapted from FeiFei Li, Andrej Karpathy
Convolution Layers

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

Example:
7×7 input
assume 3×3 connectivity
stride 1
Convolution Layers

Example:
7×7 input
assume 3×3 connectivity
stride 1

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

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

Example:
- 7×7 input
- assume 3×3 connectivity
- stride 1

Slide credit: FeiFei Li, Andrej Karpathy
Convolution Layers

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

Example:
7×7 input
assume 3×3 connectivity
stride 1
Convolution Layers

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

Example:
7×7 input
assume 3×3 connectivity
stride 1
⇒ 5×5 output
Convolution Layers

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

Example:
- 7×7 input
- assume 3×3 connectivity
- stride 1
  ⇒ 5×5 output

What about stride 2?
Convolution Layers

Example:
7×7 input
assume 3×3 connectivity
stride 1
⇒ 5×5 output

What about stride 2?

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

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

Example:
7×7 input
assume 3×3 connectivity
stride 1
⇒ 5×5 output

What about stride 2?
⇒ 3×3 output
Convolution Layers

- Replicate this column of hidden neurons across space, with some **stride**.
- In practice, common to zero-pad the border.
  - Preserves the size of the input spatially.

Example:
7×7 input
assume 3×3 connectivity
stride 1
⇒ 5×5 output

What about stride 2?
⇒ 3×3 output

- Example:
  - 7×7 input
  - Assume 3×3 connectivity
  - Stride 1
  - ⇒ 5×5 output

Slide credit: FeiFei Li, Andrej Karpathy
Effect of Multiple Convolution Layers

Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Slide credit: Yann LeCun
Commonly Used Nonlinearities

- **Sigmoid**
  \[ g(a) = \sigma(a) = \frac{1}{1 + \exp\{-a\}} \]

- **Hyperbolic tangent**
  \[ g(a) = \tanh(a) = 2\sigma(2a) - 1 \]

- **Rectified linear unit (ReLU)**
  \[ g(a) = \max\{0, a\} \]

Preferred option for deep networks
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?
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 the features.

Image source: Yann LeCun

Slide adapted from Marc'Aurelio Ranzato
Max Pooling

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

Slide adapted from FeiFei Li, Andrej Karpathy
Max Pooling

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

Slide adapted from FeiFei Li, Andrej Karpathy
Compare: SIFT Descriptor

Image Pixels → Apply oriented filters

Spatial pool (Sum) → Normalize to unit length

→ Feature Vector

Lowe [IJCV 2004]

Slide credit: Svetlana Lazebnik
Compare: Spatial Pyramid Matching

SIFT features → Filter with Visual Words

Take max VW response (L-inf normalization)

Multi-scale spatial pool (Sum)

Global image descriptor

Lazebnik, Schmid, Ponce
[CVPR 2006]
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/