Computer Vision 2
WS 2018/19

Part 6 – Tracking by Detection
31.10.2018

Prof. Dr. Bastian Leibe

RWTH Aachen University, Computer Vision Group
http://www.vision.rwth-aachen.de
Course Outline

• Single-Object Tracking
  – Background modeling
  – Template based tracking
  – Tracking by online classification
  – Tracking-by-detection

• Bayesian Filtering

• Multi-Object Tracking

• Visual Odometry

• Visual SLAM & 3D Reconstruction

• Deep Learning for Video Analysis
Recap: Tracking as Online Classification

• Tracking as binary classification problem

object vs. background
Recap: Tracking as Online Classification

- Tracking as binary classification problem

-- Handle object and background changes by online updating

• Main idea
  – Iteratively select an ensemble of classifiers
  – Reweight misclassified training examples after each iteration to focus training on difficult cases.

• Components
  – $h_m(x)$: “weak” or base classifier
    • Condition: <50% training error over any distribution
  – $H(x)$: “strong” or final classifier

• AdaBoost:
  – Construct a strong classifier as a thresholded linear combination of the weighted weak classifiers:

  $$H(x) = \text{sign} \left( \sum_{m=1}^{M} \alpha_m h_m(x) \right)$$

[Freund & Schapire, 1996]
Recap: AdaBoost – Algorithm

1. Initialization: Set \( w^{(1)}_n = \frac{1}{N} \) for \( n = 1, \ldots, N \).

2. For \( m = 1, \ldots, M \) iterations
   a) Train a new weak classifier \( h_m(x) \) using the current weighting coefficients \( W^{(m)} \) by minimizing the weighted error function
      \[
      J_m = \sum_{n=1}^{N} w^{(m)}_n I(h_m(x) \neq t_n)
      \]
   b) Estimate the weighted error of this classifier on \( X \):
      \[
      \epsilon_m = \frac{\sum_{n=1}^{N} w^{(m)}_n I(h_m(x) \neq t_n)}{\sum_{n=1}^{N} w^{(m)}_n}
      \]
   c) Calculate a weighting coefficient for \( h_m(x) \):
      \[
      \alpha_m = \ln \left\{ \frac{1 - \epsilon_m}{\epsilon_m} \right\}
      \]
   d) Update the weighting coefficients:
      \[
      w^{(m+1)}_n = w^{(m)}_n \exp \{ \alpha_m I(h_m(x_n) \neq t_n) \}
      \]
From Offline to Online Boosting

• Main issue
  – Computing the weight distribution for the samples.
  – We do not know a priori the difficulty of a sample!
    (Could already have seen the same sample before...)

• Idea of Online Boosting
  – Estimate the importance of a sample by propagating it through
    a set of weak classifiers.
  – This can be thought of as modeling the information gain w.r.t. the first $n$
    classifiers and code it by the importance weight $\lambda$ for the $n+1$ classifier.
  – Proven [Oza]: Given the same training set, Online Boosting converges
    to the same weak classifiers as Offline Boosting in the limit of $N \to \infty$
    iterations.

N. Oza and S. Russell. Online Bagging and Boosting.
Recap: From Offline to Online Boosting

**off-line**

Given:
- set of labeled training samples
  \( \mathcal{X} = \{ \langle x_1, y_1 \rangle, \ldots, \langle x_L, y_L \rangle \mid y_i \pm 1 \} \)
- weight distribution over them
  \( D_0 = 1/L \)

for \( n = 1 \) to \( N \)
- train a weak classifier using samples and weight dist.
  \( h_n^{weak}(x) = \mathcal{L}(\mathcal{X}, D_{n-1}) \)
- calculate error \( e_n \)
- calculate weight \( \alpha_n = f(e_n) \)
- update weight dist. \( D_n \)

next

\( h^{strong}(x) = \text{sign}( \sum_{n=1}^{N} \alpha_n \cdot h_n^{weak}(x) ) \)

**on-line**

Given:
- ONE labeled training sample
  \( \langle x, y \rangle \mid y \pm 1 \)
- strong classifier to update
- initial importance \( \lambda = 1 \)

for \( n = 1 \) to \( N \)
- update the weak classifier using samples and importance
  \( h_n^{weak}(x) = \mathcal{L}(h_n^{weak}, \langle x, y \rangle, \lambda) \)
- update error estimation \( \tilde{e}_n \)
- update weight \( \alpha_n = f(\tilde{e}_n) \)
- update importance weight \( \lambda \)

next

\( h^{strong}(x) = \text{sign}( \sum_{n=1}^{N} \alpha_n \cdot h_n^{weak}(x) ) \)
Recap: Online Boosting for Feature Selection

- Introducing “Selector”
  - Selects one feature from its local feature pool

\[ \mathcal{H}^{weak} = \{ h_1^{weak}, \ldots, h_M^{weak} \} \]
\[ \mathcal{F} = \{ f_1, \ldots, f_M \} \]

\[ h_{sel}(x) = h_m^{weak}(x) \]
\[ m = \arg\min_i e_i \]

On-line boosting is performed on the Selectors and not on the weak classifiers directly.

H. Grabner and H. Bischof. 
*On-line boosting and vision.*
Recap: Direct Feature Selection

1. One training sample
2. Estimate errors
3. Select best weak classifier
4. Estimate importance
5. Update weight
6. Repeat for each training sample

- Initial importance $\lambda = 1$
- Select best weak classifier
- Estimate importance
- Update weight
- Current strong classifier $h_{\text{Strong}}$
Recap: Tracking by Online Classification

- **Update classifier (tracker)**
- **Search region**
- **Create confidence map**
- **Evaluate classifier on sub-patches**
- **Analyze map and set new object position**

Visual object position from time $t$ to $t+1$
Recap: Drifting Due to Self-Learning Policy

Not only does it drift, it also remains confident about it!
Today: Tracking by Detection

Can we use generic object detection to track people?

Object detections

Spacetime trajectories

Image source: B. Leibe
Topics of This Lecture

- **Tracking by Detection**
  - Motivation
  - Recap: Object detection

- **SVM based Detectors**
  - Recap: HOG
  - DPM

- **AdaBoost based Detectors**
  - Recap: Viola-Jones
  - Integral Channel features
  - VeryFast/Roerei

- **CNN-based Detectors**
  - Recap: CNNs
  - R-CNN, Faster R-CNN
  - YOLO, SSD
Detection-Based Tracking

• Main ideas
  – Apply a generic object detector to find objects of a certain class
  – Based on the detections, extract object appearance models
    ▪ Even possible to derive figure-ground segmentations from detection results
  – Link detections into trajectories
Tracking-by-Detection in 3D

Object detections

Spacetime trajectories

3D Camera path estimation

Simple f/g model: E.g., elliptical region in detection box

Main Issue: Data Association (We’ll come to that later…)

[Leibe, Cornelis, Schindler, Van Gool, PAMI’08]
Spacetime Trajectory Analysis

![Spacetime Trajectory Analysis Diagram]

- Pedestrian detection
- Car detections
- Own vehicle

[Leibe, Cornelis, Schindler, Van Gool, CVPR'07]
Elements of Tracking

- Detection
  - *Where are candidate objects?*

- Data association
  - *Which detection corresponds to which object?*

- Prediction
  - *Where will the tracked object be in the next time step?*

Today’s topic
Recap: Sliding-Window Object Detection

• Basic component: a binary classifier
Recap: Sliding-Window Object Detection

• If object may be in a cluttered scene, slide a window around looking for it.

• Essentially, this is a brute-force approach with many local decisions.
What is a Sliding Window Approach?

- Search over space and scale

- Detection as subwindow classification problem

- “In the absence of a more intelligent strategy, any global image classification approach can be converted into a localization approach by using a sliding-window search.”
Recap: Non-Maximum Suppression

After multi-scale dense scan

Clip detection score

Map each detection to 3D \([x, y, scale]\) space

Apply robust mode detection, e.g. mean shift

Non-maximum suppression

Image source: Navneet Dalal, PhD Thesis
Recap: Sliding-Window Object Detection

- Fleshing out this pipeline a bit more, we need to:
  1. Obtain training data
  2. Define features
  3. Define classifier

Slide credit: Kristen Grauman
Object Detector Design

• In practice, the classifier often determines the design.
  – Types of features
  – Speedup strategies

• Today, we’ll look at 3 state-of-the-art detector designs
  – Based on SVMs
    – Based on Boosting
  – Based on CNNs
Topics of This Lecture

• Tracking by Detection
  – Motivation
  – Recap: Object detection

• SVM based Detectors
  – Recap: HOG
  – DPM

• AdaBoost based Detectors
  – Recap: Viola-Jones
  – Integral Channel features
  – VeryFast/Roerei

• CNN-based Detectors
  – Recap: CNNs
  – R-CNN
Recap: Histograms of Oriented Gradients (HOG)

- Holistic object representation
  - Localized gradient orientations

Object/Non-object

Linear SVM

Collect HOGs over detection window

Contrast normalize over overlapping spatial cells

Weighted vote in spatial & orientation cells

Compute gradients

Gamma compression

Image Window
Recap: Support Vector Machine (SVM)

• Basic idea
  – The SVM tries to find a classifier which maximizes the margin between pos. and neg. data points.
  – Up to now: consider linear classifiers
    \[ \mathbf{w}^T \mathbf{x} + b = 0 \]

• Formulation as a convex optimization problem
  – Find the hyperplane satisfying
    \[
    \arg \min_{\mathbf{w},b} \frac{1}{2} \|\mathbf{w}\|^2
    \]
    under the constraints
    \[
    t_n (\mathbf{w}^T \mathbf{x}_n + b) \geq 1 \quad \forall n
    \]
    based on training data points \( \mathbf{x}_n \) and target values \( t_n \in \{-1, 1\} \)
Recap: Pedestrian Detection with HOG

- Train a pedestrian template using a linear SVM
- At test time, convolve feature map with template

\[ y(x) = w^T x + b \]

Pedestrian detection with HoGs & SVMs

Topics of This Lecture

• Tracking by Detection
  – Motivation
  – Recap: Object detection

• SVM based Detectors
  – Recap: HOG
  – DPM

• AdaBoost based Detectors
  – Recap: Viola-Jones
  – Integral Channel features
  – VeryFast/Roerei

• CNN-based Detectors
  – Recap: CNNs
  – R-CNN
Recap: Part-Based Models

- Pictorial Structures model
  - [Fischler & Elschlager 1973]

- Model has two components
  - Parts (2D image fragments)
  - Structure (configuration of parts)

- Use in **Deformable Part-based Model (DPM)**
  - Parts $\equiv$ 5-7 semantically meaningful parts
  - Probabilistic model enabling efficient inference
Starting Point: HOG Sliding-Window Detector

- Array of weights for features in window of HOG pyramid
- Score is dot product of filter and vector

Filter $F$

Score of $F$ at position $p$ is

$$F \cdot \phi(p, H)$$

$$\phi(p, H) = \text{concatenation of HOG features from window specified by } p.$$
Deformable Part-based Models

- Mixture of deformable part models (Pictorial Structures)
- Each component has global template + deformable parts
- Fully trained from bounding boxes alone

Slide credit: Pedro Felzenszwalb

[Felzenszwalb, McAllister, Ramanan, CVPR'08]
2-Component Bicycle Model

Root filters
coarse resolution

Part filters
finer resolution

Deformation models

Slide credit: Pedro Felzenszwalb [Felzenszwalb, McAllister, Ramanan, CVPR'08]
Object Hypothesis

• Multiscale model captures features at two resolutions

Score of filter: dot product of filter with HOG features underneath it

Score of object hypothesis is sum of filter scores minus deformation costs

Slide credit: Pedro Felzenszwalb

Felzenszwalb, McAllister, Ramanan, CVPR'08
Score of a Hypothesis

\[ \text{score}(p_0, \ldots, p_n) = \sum_{i=0}^{n} F_i \cdot \phi(H, p_i) - \sum_{i=1}^{n} d_i \cdot (dx_i^2, dy_i^2) \]

Score of Hypothesis

\[ \text{score}(z) = \beta \cdot \Psi(H, z) \]

concatenation filters and deformation parameters

concatenation of HOG features and part displacement features
Recognition Model

\[
f_w(x) = w \cdot \Phi(x)
\]

\[
f_w(x) = \max_z w \cdot \Phi(x, z)
\]

- Difference to standard HOG model
  - Hidden variable \(z\): vector of part offsets
  - \(\Phi(x, z)\): vector of HOG features (from root filter & appropriate part sub-windows) and part offsets
  \[\Rightarrow\] Need to optimize over all possible part positions

Slide credit: Pedro Felzenszwalb
Results: Persons

• Results (after non-maximum suppression)
  – ~1s to search all scales
Results: Bicycles

Slide adapted from Trevor Darrell
• More efficient features
  – Very simplified version of HOG

• Latent part (re-)learning
  – Perform several rounds of training, adapting the annotation bboxes

• Multi-aspect detection
  – Mixture model of different aspects to capture different viewpoints of objects

• Bounding box prediction
  – Infer final detection bounding box from detected part locations

• Multi-resolution models

• Cascaded evaluation

[Felzenszwalb, McAllister, Ramanan, PAMI’10]
You Can Try It At Home…

• Deformable part-based models have been very successful in several evaluations.
  ⇒ Approach was *state-of-the-art* until few years ago

• Source code and models trained on PASCAL 2006, 2007, and 2008 data are available here:
Topics of This Lecture

• Tracking by Detection
  – Motivation
  – Recap: Object detection

• SVM based Detectors
  – Recap: HOG
  – DPM

• AdaBoost based Detectors
  – Recap: Viola-Jones
  – Integral Channel features
  – VeryFast/Roerei

• CNN-based Detectors
  – Recap: CNNs
  – R-CNN
Recap: Viola-Jones Face Detector

- Train with 5K positives, 350M negatives
- Real-time detector using 38 layer cascade (6061 features in final layer)
- [Implementation available in OpenCV: http://sourceforge.net/projects/opencvlibrary/]

Slide credit: Kristen Grauman
Recap: Haar Wavelets

“Rectangular” filters

Feature output is difference between adjacent regions

Efficiently computable with integral image: any sum can be computed in constant time

Avoid scaling images ⇒ Scale features directly for same cost

Value at \((x,y)\) is sum of pixels above and to the left of \((x,y)\)

\[
D = 1 + 4 - (2 + 3) \\
= A + (A + B + C + D) - (A + C + A + B) \\
= D
\]
Recap: Cascading Classifiers for Detection

- Even if the filters are fast to compute, each new image has a lot of possible windows to search...

- Idea: Classifier cascade
  - Observation: most image windows are negative and look very different from the searched object class.
  - Filter for promising regions with an initial inexpensive classifier
  - Build a chain of classifiers, choosing cheap ones with low false negative rates early in the chain

[Fleuret & Geman, IJCV’01; Rowley et al., PAMI’98; Viola & Jones, CVPR’01]
Viola-Jones Face Detector: Results
You Can Try It At Home…

- The Viola & Jones detector was a huge success
  - First real-time face detector available
  - Many derivative works and improvements

- C++ implementation available in OpenCV [Lienhart, 2002]
  - [http://sourceforge.net/projects/opencvlibrary/](http://sourceforge.net/projects/opencvlibrary/)
- Matlab wrappers for OpenCV code available, e.g. here

Topics of This Lecture

• Tracking by Detection
  – Motivation
  – Recap: Object detection

• SVM based Detectors
  – Recap: HOG
  – DPM

• AdaBoost based Detectors
  – Recap: Viola-Jones
  – Integral Channel features
  – VeryFast/Roerei

• CNN-based Detectors
  – Recap: CNNs
  – R-CNN
Integral Channel Features

- Generalization of Haar Wavelet idea from Viola-Jones
  - Instead of only considering intensities, also take into account other feature channels (gradient orientations, color, texture).
  - Still efficiently represented as integral images.

Integral Channel Features

- Generalize also block computation
  - 1\textsuperscript{st} order features:
    - Sum of pixels in rectangular region.
  - 2\textsuperscript{nd}-order features:
    - Haar-like difference of sum-over-blocks
  - Generalized Haar:
    - More complex combinations of weighted rectangles
  - Histograms
    - Computed by evaluating local sums on quantized images.
Results: Integral Channel Features

- fastHOG (Prisacariu, 2009)
- DPM (Felzenszwalb, 2008)
- ChnFtrs/FPDW (Dollar, 2009+2010)

Slide credit: Rodrigo Benenson
Topics of This Lecture

• Tracking by Detection
  – Motivation
  – Recap: Object detection

• SVM based Detectors
  – Recap: HOG
  – DPM

• AdaBoost based Detectors
  – Recap: Viola-Jones
  – Integral Channel features
  – VeryFast/Roerei

• CNN-based Detectors
  – Recap: CNNs
  – R-CNN
Performance Comparison of Detectors

INRIA dataset

- Shapelet–orig (90.5%)
- PoseInvSvm (68.6%)
- VJ–OpenCv (53.0%)
- PoseInv (51.4%)
- Shapelet (50.4%)
- VJ (47.5%)
- FtrMine (34.0%)
- Pls (23.4%)
- HOG (23.1%)
- HikSvm (21.9%)
- LatSvm–V1 (17.5%)
- MultiFtr (15.6%)
- MultiFtr+CSS (10.9%)
- LatSvm–V2 (9.9%)
- FPDW (9.9%)
- ChnFtrs (8.7%)

false positives per image

miss rate

Better
Performance Comparison of Detectors

INRIA dataset

VeryFast 50 Hz

Better

Better

false positives per image

miss rate

1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.05
0.02
0.01
0.001
0.0001

Shapelet–orig (90.5%)
PoseInvSvm (68.6%)
VJ–OpenCv (53.0%)
PoseInv (51.4%)
Shapelet (50.4%)
VJ (47.5%)
FtrMine (34.0%)
Pls (23.4%)
HOG (23.1%)
HikSvm (21.9%)
LatSvm–V1 (17.5%)
MultiFtr (15.6%)
MultiFtr+CSS (10.9%)
LatSvm–V2 (9.3%)
FPDW (9.3%)
ChnFtrs (8.7%)
Ours–VeryFast (6.8%)
Issues for Efficient Detection

• One template cannot detect at multiple scales...
Issues for Efficient Detection

• Typically, features are computed many times

~50 scales
Issues for Efficient Detection

- Typically, features are computed many times

~50 scales
VeryFast Detector

• **Idea 1:** Invert the relation

Practical Considerations

• Training and running 1 model/scale is too expensive
VeryFast Detector

- **Idea 2**: Reduce training time by feature interpolation

- Shown to be possible for Integral Channel features

5 models, 1 image scale
≈
50 models, 1 image scale
VeryFast Detector

- Effect: Transfer test time computation to training time

5 models, 1 image scale
≈
50 models, 1 image scale

⇒ Result: 3x reduction in feature computation

Slide credit: Rodrigo Benenson
VeryFast: Classifier Construction

- Ensemble of short trees, learned by AdaBoost

\[ score = w_1 \cdot h_1 + \]
VeryFast: Classifier Construction

- Ensemble of short trees, learned by AdaBoost

\[ \text{score} = w_1 \cdot h_1 + w_2 \cdot h_2 + \]

Slide credit: Rodrigo Benenson
VeryFast: Classifier Construction

• Ensemble of short trees, learned by AdaBoost

$$score = w_1 \cdot h_1 + w_2 \cdot h_2 + \ldots + w_N \cdot h_N$$
Learned Models

Integral Channel features

Models
Results

- Detection without resizing improves quality of results

Slide credit: Rodrigo Benenson
Multi-Scale Models > Single-Scale Model
Comparison to State-of-the-Art

- Extension: Roerei detector
  - Detailed evaluation of design space
  - Non-regular pooling regions found to work best.
Roerei Results

Applications: Mobile Robot Navigation

link to the video
Topics of This Lecture

• Tracking by Detection
  – Motivation
  – Recap: Object detection

• SVM based Detectors
  – Recap: HOG
  – DPM

• AdaBoost based Detectors
  – Recap: Viola-Jones
  – Integral Channel features
  – VeryFast/Roerei

• CNN-based Detectors
  – Recap: CNNs
  – R-CNN
Recap: Convolutional Neural Networks

- 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

Recap: Intuition of CNNs

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

- Learn *multiple* filters
  - E.g. $1000 \times 1000$ image
  - 100 filters
  - $10 \times 10$ filter size
  - $\Rightarrow$ only $10k$ parameters

- Result: Response map
  - size: $1000 \times 1000 \times 100$
  - Only memory, not params!
Recap: 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.
Recap: Activation Maps

5×5 filters

Activation maps

Slide adapted from FeiFei Li, Andrej Karpathy
Recap: Pooling Layers

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

Slide adapted from FeiFei Li, Andrej Karpathy
Recap: R-CNN for Object Detection

1. Input image
2. Warped image regions
3. Regions of Interest (RoI) from a proposal method (~2k)
4. Forward each region through ConvNet
5. Classify regions with SVMs
6. Bbox reg
7. SVMs
8. Bbox reg
9. ConvNet
10. ConvNet

Slide credit: Ross Girshick
Recap: Faster R-CNN

- One network, four losses
  - Remove dependence on external region proposal algorithm.
  - Instead, infer region proposals from same CNN.
  - Feature sharing
  - Joint training
    ⇒ Object detection in a single pass becomes possible.
Most Recent Version: Mask R-CNN

Mask R-CNN Results

- Detection + Instance segmentation

- Detection + Pose estimation

Figure credit: K. He, G. Gkioxari, P. Dollar, R. Girshick
YOLO / SSD

- Idea: Directly go from image to detection scores
- Within each grid cell
  - Start from a set of anchor boxes
  - Regress from each of the B anchor boxes to a final box
  - Predict scores for each of C classes (including background)
You Can Try All of This At Home…

- Detector code is publicly available
  - **HOG:**
    - Dalal’s original implementation: [http://www.navneetdalal.com/software/](http://www.navneetdalal.com/software/)
    - Our CUDA-optimized *groundHOG* code (>80 fps on GTX 580) [http://www.vision.rwth-aachen.de/software/groundhog](http://www.vision.rwth-aachen.de/software/groundhog)
  - **DPM:**
    - Felzenswalb’s original implementation: [http://www.cs.uchicago.edu/~pff/latent](http://www.cs.uchicago.edu/~pff/latent)
  - **VeryFast**
    - Benenson’s original implementation: [https://bitbucket.org/rodrigob/doppia/](https://bitbucket.org/rodrigob/doppia/)
  - **YOLO**