Computer Vision II - Lecture 7

Tracking by Detection

15.05.2014

Bastian Leibe
RWTH Aachen
http://www.vision.rwth-aachen.de
leibe@vision.rwth-aachen.de

Course Outline

- Single-Object Tracking
  - Background modeling
  - Template based tracking
  - Color based tracking
  - Contour based tracking
  - Tracking by online classification
  - Tracking-by-detection

- Bayesian Filtering
- Multi-Object Tracking
- Articulated Tracking

Today: Tracking by Detection

Recap: Tracking as Online Classification

- Tracking as binary classification problem

Recap: AdaBoost - “Adaptive Boosting”

- Main idea [Freund & Schapire, 1996]
  - 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:
    
    \[
    H(x) = \text{sign} \left( \sum_{m=1}^{M} \alpha_m h_m(x) \right)
    \]
Recap: AdaBoost - Algorithm

1. Initialization: Set \( w_i^{(1)} = \frac{1}{N} \) for \( i = 1, \ldots, N \).

2. For \( m = 1, \ldots, M \) iterations
   a) Train a new weak classifier \( h_m(x) \) using the current weighting coefficients \( W(n) \) by minimizing the weighted error function
   \[
   J_m = \sum_{n=1}^{N} w_i^{(m)} \cdot \delta(h_m(x) \neq t_n) \quad \text{if} \quad k \text{ is true}
   \]
   \[
   \text{else}
   \]
   b) Estimate the weighted error of this classifier on \( X \):
   \[
   \varepsilon_m = \frac{1}{N} \sum_{n=1}^{N} w_i^{(m)} \cdot \delta(h_m(x) \neq t_n)
   \]
   c) Calculate a weighting coefficient for \( h_m(x) \):
   \[
   \alpha_m = \ln \left( \frac{1 - \varepsilon_m}{\varepsilon_m} \right)
   \]
   d) Update the weighting coefficients:
   \[
   w_i^{(m+1)} = w_i^{(m)} \exp\left( \alpha_m \cdot \delta(h_m(x) \neq t_n) \right)
   \]

Recap: 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.


Recap: Online Boosting for Feature Selection

- Introducing “Selector”
  - Selects one feature from its local feature pool
    \( \mathcal{F} = \{ f_1, \ldots, f_M \} \)
    \[ h^{(m)}(x) = h^{(m)}_M(x) \]
    \[ m = \text{arg min}_m \varepsilon_m \]

Recap: Direct Feature Selection

- **Shared feature pool for all selectors to save computation**

Recap: Tracking by Online Classification

- **From time \( t \) to \( t+1 \)**
  - Update classifier (tracker)
  - Analyze map and set new object position
  - Create confidence map

- Image source: Disney / Pixar
Recap: Self-Learning and Drift

- Drift
  - Major problem in all adaptive or self-learning trackers.
  - Difficulty: distinguish "allowed" appearance changes due to lighting or viewpoint variation from "unwanted" appearance change due to drifting.
  - Cannot be decided based on the tracker confidence!
- Several approaches to address this
  - Comparison with initialization
  - Semi-supervised learning (additional data)
  - Additional information sources

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
- Random Forest based Detectors
  - Recap: ISM
  - Hough Forests

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

- Main Issue: Data Association
  - (We’ll come to that...)

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?
Recap: Sliding-Window Object Detection

- Basic component: a binary classifier

Slide credit: Kristen Grauman

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.

Slide credit: Kristen Grauman

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.”

Slide credit: Kristen Grauman

Recap: 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 Random Forests
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
- Random Forest based Detectors
  - Recap: ISM
  - Hough Forests

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
    \[ w^T x + b = 0 \]
- Formulation as a convex optimization problem
  - Find the hyperplane satisfying
    \[ \arg \min_{w, b} \frac{1}{2} \|w\|^2 \]
    under the constraints
    \[ t_n (w^T x_n + b) \geq 1 \quad \forall n \]
    based on training data points \( x_n \) and target values \( t_n \in \{-1, 1\} \).

Recap: Histograms of Oriented Gradients (HOG)

- Holistic object representation
  - Localized gradient orientations
    \[ [..., ..., ..., ...] \]
- Recap: Histograms of Oriented Gradients for Human Detection, CVPR 2005

Recap: Pedestrian Detection with HOG

- Train a pedestrian template using a linear SVM
- At test time, convolve feature map with template
  \[ y(x) = w^T x + b \]
- N. Dalal and B. Triggs, Histograms of Oriented Gradients for Human Detection, CVPR 2005
- Slide credit: Svetlana Lazebnik

Pedestrian detection with HoGs & SVMs

- N. Dalal, B. Triggs, Histograms of Oriented Gradients for Human Detection, CVPR'05
- Slide credit: Kristen Grauman

Extensions and Improvements(?)

(a) INRIA per-window results.
(b) INRIA per-image results.
- Choice of evaluation criterion is critical!
  - Traditional evaluations on per-window classification.
  - [Dollar et al.,'09]: None of the methods proposed from 2004-2009 brought an improvement for the actual detection task!
Some Extensions that Did Survive…

  - Compute LBP histograms over cells, as in HOG
  - Features seem to be complementary to some degree

- **HOG + Depth + Flow** [Wojek et al. 2010, Gavrila 2012]
  - For applications in intelligent vehicles where those are available
  - Factor 40 reduction in false positives possible

- **HIK-SVM** [Maji et al. 2008]
  - Apply non-linear SVM kernels at reduced cost

- **Explicit Feature Maps** [Vedaldi & Zisserman 2010, ’12]
  - Same as above, but on steroids

Incorporating Ground Plane Constraints

- Efficient integration into detector design (groundHOG)
  - Idea: only evaluate geometrically valid detection windows
  - Derivation: Region of interest lies between two parabolas…
  - …that can in most cases be approximated by straight lines.
  - Only touch pixels inside the ROI for all computations.
  - Factor 2-4 speed improvement on top of all other optimizations

Real-Time Pedestrian Detection

- Efficient CUDA HOG implementation (equivalent to original HOG code)
- Code made publicly available as open source under GPL
- Run-time comparison:

<table>
<thead>
<tr>
<th>run-time</th>
<th>1280 × 960</th>
<th>640 × 480</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>cuda</td>
<td>ground</td>
</tr>
<tr>
<td>Laptop GTX 285M</td>
<td>1.6 fps</td>
<td>9.6 fps</td>
</tr>
<tr>
<td>Desktop GTX 280</td>
<td>5.5 fps</td>
<td>17.2 fps</td>
</tr>
<tr>
<td>Desktop GTX 580</td>
<td>9.8 fps</td>
<td>27.8 fps</td>
</tr>
</tbody>
</table>

⇒ Detection at video frame rate possible even on laptops with mobile GPUs!

You Can Try It At Home...

- groundHOG GPU detector code publicly available
  - Highly optimized for speed
  - Can be used with or without ground plane constraints
  - Supports general ROI processing
  - Supports multi-class detection with feature sharing
  - Published under GPL license (other licensing negotiable)

  http://www.vision.rwth-aachen.de/projects/groundhog

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 = 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

Deformable Part-based Models

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

2-Component Bicycle Model

- Root filters: coarse resolution
- Part filters: finer resolution
- Deformation models

Object Hypothesis

- Multiscale model captures features at two resolutions

Score of a Hypothesis

\[ \text{score}(p_0, \ldots, p_n) = \sum_{i=1}^{n} d_i \cdot (d_{x_i}^2 + d_{y_i}^2) \]

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

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
  - Need to optimize over all possible part positions
Results: Persons

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

Results: Bicycles

• 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

You Can Try It At Home…

• Deformable part-based models have been very successful at several recent evaluations.
  ⇒ One of the state-of-the-art approaches in object detection

• Source code and models trained on PASCAL 2006, 2007, and 2008 data are available here:
  http://www.cs.uchicago.edu/~pff/latent

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/Roberei

• Random Forest based Detectors
  > Recap: ISM
  > Hough Forests
Recap: Viola-Jones Face Detector

Train cascade of classifiers with AdaBoost

- 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/]

Recap: Haar Wavelets

“Rectangular” filters

Feature output is difference between adjacent regions

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

Efficiently computable with integral image: any sum can be computed in constant time
Avoid scaling images \(\Rightarrow\) Scale features directly for same cost

AdaBoost for Efficient Feature Selection

- Image features = weak classifiers
- For each round of boosting:
  - Evaluate each rectangle filter on each example
  - Sort examples by filter values
  - Select best threshold for each filter (min error)
  - Sorted list can be quickly scanned for the optimal threshold
  - Select best filter/threshold combination
  - Weight on this feature is a simple function of error rate
  - Reweight examples

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

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/
- 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
- Random Forest based Detectors
  - Recap: ISM
  - Hough Forests

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
  - 1st order features:
    - Sum of pixels in rectangular region.
  - 2nd-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 ~10 Hz on GPU [Prisacariu 2009]
- DPM [Felzenszwalb 2008]
- ChnFtrs/FPDW ~5 Hz on CPU [Dollar 2009+2010]
**Issues for Efficient Detection**

- One template cannot detect at multiple scales...

**VeryFast 50 Hz**

**Issues for Efficient Detection**

- Typically, features are computed many times

**Practical Considerations**

- Training and running 1 model/scale is too expensive

VeryFast Detector

- Idea 2: Reduce training time by feature interpolation

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

- Shown to be possible for Integral Channel features


⇒ Result: 3x reduction in feature computation

VeryFast: Classifier Construction

- Ensemble of short trees, learned by AdaBoost

\[ \text{score} = w_1 \cdot h_1 + \cdots + w_N \cdot h_N \]

Learned Models

Integral Channel features

Models...
Results

- Detection without resizing provides quality

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
- Random Forest based Detectors
  - Recap: ISM
  - Hough Forests
Recap: Implicit Shape Model (ISM) Idea

- Visual vocabulary is used to index votes for object position [a visual word = “part”].

![Recap: Implicit Shape Model (ISM) Idea](image)

Recap: ISM - Representation

- Learn appearance codebook
  - Extract local features at interest points
  - Feature clustering on codebook
- Learn spatial distributions
  - Match codebook to training images
  - Record matching positions on object

![Recap: ISM - Representation](image)

Recap: ISM - Recognition

- Probabilistic vote weighting
- Learn appearance codebook
- Match codebook to training images

![Recap: ISM - Recognition](image)

Recap: ISM - Top-Down Segmentation

- Backproject hypotheses
- Backprojection of maxima

![Recap: ISM - Top-Down Segmentation](image)
Class-Specific Top-Down Segmentation

- During initial Hough Voting
  - When we first observe a feature, we do not know its context.
  - Different figure-ground labels may be consistent with the appearance.
  - Strategy: we cast votes for many locations...
- After voting
  - Voting groups features that are consistent with the same object.
  - We can now consider each feature conditioned on the selected object location hypothesis.
  - This allows us to backproject a local figure-ground label from selected votes.

Top-Down Segmentation

- Interpretation of $p(\text{figure})$ map
  - per-pixel confidence in object hypothesis
  - Useful for hypothesis verification

Recap: ISM - Example Results

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
- Random Forest based Detectors
  - Recap: ISM
  - Hough Forests

Hough Forest Object Detector

- Combine idea of ISM-style Hough voting with dense feature sampling and discriminative training.
  - Randomized forest classifier densely processes image patches
  - Leaf nodes correspond to visual words
  - Cast votes for possible object hypotheses
- Good empirical performance, fast to evaluate

Fast Dense Matching with Random Forests

- Ideas
  - Solve feature extraction and codebook matching at the same time
  - Discriminative training of codebook features
- Extremely simple features
  - 2-pixel comparisons in different feature channels
  - Evaluation sub-linear in patch size
- Tree construction
  - Each leaf node contains occurrence distribution for Hough Voting
  - Training goal: Minimize class entropy while keeping distributions compact
Multi-View Extension

- Random Forests are implicitly multi-class capable
  - Create multi-class tree with per-class occurrence distributions
  - Use one Hough space per class or viewpoint
  - Necessary: multi-class non-maximum suppression

Top-Down Segmentation with Hough Forests

- Extend HF's with top-down segmentation mechanism
- Better results than for ISM due to dense sampling

HF-ISM: Qualitative Results

- Observations
  - Improved detection performance compared to original HF
    (competitive with HOG + HIKSVM on pedestrians).
  - Better segmentations than original ISM due to dense sampling.

You Can Try All of This At Home...

- Detector code is publicly available
  - HOG: 
    - Dalal’s original implementation: http://www.navneetdalal.com/software/
    - Our CUDA-optimized groundHOG code (>80 fps on GTX 580) http://www.mmp.rwth-aachen.de/projects/groundhog
  - DPM: 
    - Felzenswalb’s original implementation: http://www.cs.uchicago.edu/~pff/latent
  - ISM: 
    - My original implementation: http://www.vision.rwth-aachen.de/software/ism
  - HF: 
    - Gall’s original implementation: http://www.vision.ee.ethz.ch/~gallju/index.html#software
  - VeryFast: 
    - Benenson’s original implementation: https://bitbucket.org/rodrigob/doppia/