Computer Vision II - Lecture 7

Tracking by Detection

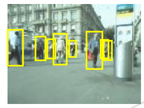
15.05.2014

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



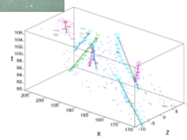
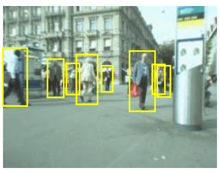
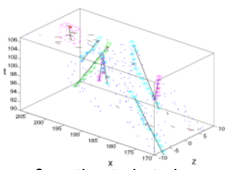


Image source: Helmut Grabner, Disney/Pixar

Today: Tracking by Detection



Object detections

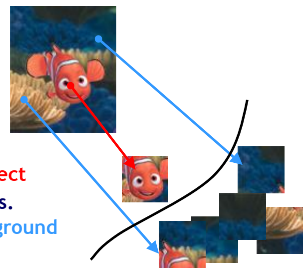


Spacetime trajectories

Image source: B. Leibe

Recap: Tracking as Online Classification

- Tracking as binary classification problem



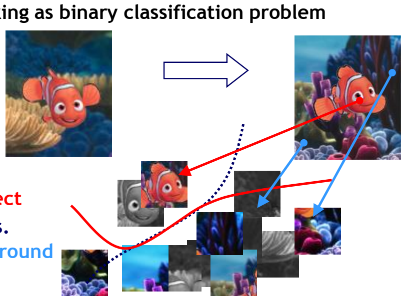
object
vs.
background

Slide credit: Helmut Grabner

Image source: Disney /Pixar

Recap: Tracking as Online Classification

- Tracking as binary classification problem



object
vs.
background

  - Handle object and background changes by online updating

Slide credit: Helmut Grabner

Image source: Disney /Pixar

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
  - $h_m(\mathbf{x})$: "weak" or base classifier
    - Condition: <50% training error over any distribution
  - $H(\mathbf{x})$: "strong" or final classifier
- AdaBoost:
  - Construct a strong classifier as a thresholded linear combination of the weighted weak classifiers:

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

Recap: AdaBoost - Algorithm

1. Initialization: Set $w_n^{(1)} = \frac{1}{N}$ for $n = 1,\dots,N$.
2. For $m = 1,\dots,M$ iterations
   a) Train a new weak classifier $h_m(\mathbf{x})$ using the current weighting coefficients $\mathbf{W}^{(m)}$ by minimizing the weighted error function
   $$J_m = \sum_{n=1}^{N} w_n^{(m)} I(h_m(\mathbf{x}) \neq t_n) \qquad I(A) = \begin{cases} 1, & \text{if } A \text{ is true} \\ 0, & \text{else} \end{cases}$$
   b) Estimate the weighted error of this classifier on $\mathbf{X}$:
   $$\epsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} I(h_m(\mathbf{x}) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}}$$
   c) Calculate a weighting coefficient for $h_m(\mathbf{x})$:
   $$\alpha_m = \ln\left\{\frac{1-\epsilon_m}{\epsilon_m}\right\}$$
   d) Update the weighting coefficients:
   $$w_n^{(m+1)} = w_n^{(m)} \exp\left\{\alpha_m I(h_m(\mathbf{x}_n) \neq t_n)\right\}$$

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 λ 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. Artificial Intelligence and Statistics, 2001.

Recap: From Offline to Online Boosting

off-line	on-line
Given:	**Given**:
- set of labeled training samples $\mathcal{X} = \{\langle \mathbf{x}_1, y_1\rangle, \dots, \langle \mathbf{x}_L, y_L\rangle \mid y_i \pm 1\}$	- ONE labeled training sample $\langle \mathbf{x}, y\rangle \mid y \pm 1$
- weight distribution over them $D_0 = 1/L$	- strong classifier to update
	- initial importance $\lambda = 1$
for n = 1 to N	for n = 1 to N
- train a weak classifier using samples and weight dist. $h_n^{weak}(\mathbf{x}) = \mathcal{L}(\mathcal{X}, D_{n-1})$	- update the weak classifier using samples and importance $h_n^{weak}(\mathbf{x}) = \mathcal{L}(h_n^{weak}, \langle x, y\rangle, \lambda)$
- calculate error e_n	- update error estimation $\tilde{e}_n$
- calculate weight $\alpha_n = f(e_n)$	- update weight $\alpha_n = f(\tilde{e}_n)$
- update weight dist. D_n	- update importance weight λ
next	next
$h^{strong}(\mathbf{x}) = \text{sign}(\sum_{n=1}^{N} \alpha_n \cdot h_n^{weak}(\mathbf{x}))$	$h^{strong}(\mathbf{x}) = \text{sign}(\sum_{n=1}^{N} \alpha_n \cdot h_n^{weak}(\mathbf{x}))$

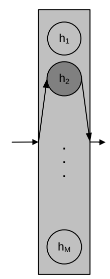
Slide credit: Helmut Grabner

Recap: Online Boosting for Feature Selection

- Introducing "Selector"
  - Selects one feature from its local feature pool
    $\mathcal{H}^{weak} = \{h_1^{weak}, \dots, h_M^{weak}\}$
    $\mathcal{F} = \{f_1, \dots, f_M\}$
    $h^{sel}(\mathbf{x}) = h_m^{weak}(\mathbf{x})$
    $m = \arg\min_i e_i$

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

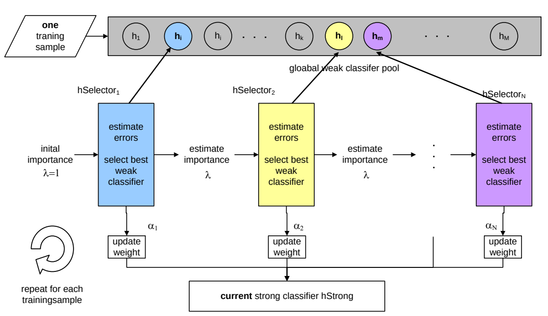
hSelector



H. Grabner and H. Bischof. On-line boosting and vision. CVPR, 2006.

Slide credit: Helmut Grabner

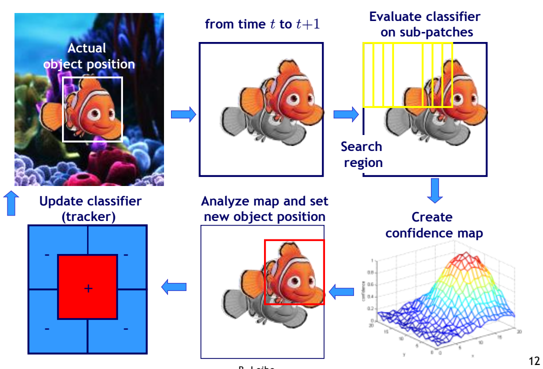
Recap: Direct Feature Selection



- Shared feature pool for all selectors to save computation

Slide credit: Helmut Grabner

Recap: Tracking by Online Classification

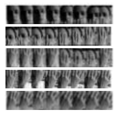


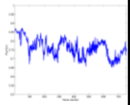
Slide credit: Helmut Grabner

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

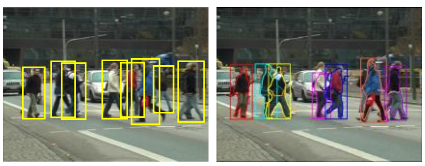

Tracked Patches


Confidence

Topics of This Lecture

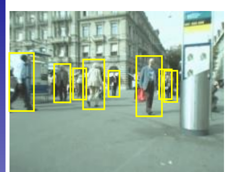
- Tracking by Detection
  - Motivation
  - Recap: Object detection
- SVM based Detectors
  - Recap: HOG
  - DPM
- AdaBoost based Detectors
  - Recap: Viola-Jones
  - Integral Channel features
  - VeryFast/Roerei
- Random Forest based Detectors
  - Recap: ISM
  - Hough Forests

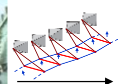
Detection-Based Tracking

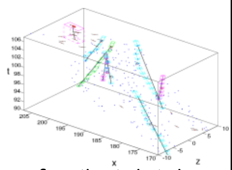


- Main ideas
  - Apply a generic object detector to find objects of a certain class
  - Based on the detections, extract object appearance models
    - Even possible to derive figure-ground segmentations from detection results
  - Link detections into trajectories

Tracking-by-Detection in 3D


Object detections


3D Camera path estimation


Spacetime trajectories

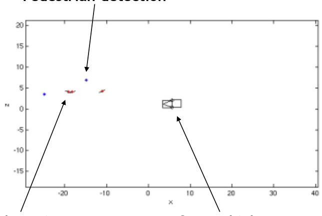

Simple f/g model:
E.g., elliptical region in detection box

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

[Leibe, Cornelis, Schindler, Van Gool, PAMI’08]

Spacetime Trajectory Analysis

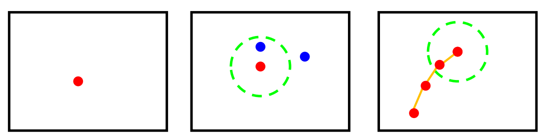
Pedestrian detection



Car detections   Own vehicle

[Leibe, Cornelis, Cornelis, Van Gool, CVPR’07]

Elements of Tracking


Detection   Data association   Prediction

- Detection
  - *Where are candidate objects?*   Today’s topic
- 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

Car/non-car Classifier

No, not a car. / Yes, car.

Slide credit: Kristen Grauman

Recap: Sliding-Window Object Detection

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

Car/non-car Classifier

- 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."*

Recap: Non-Maximum Suppression

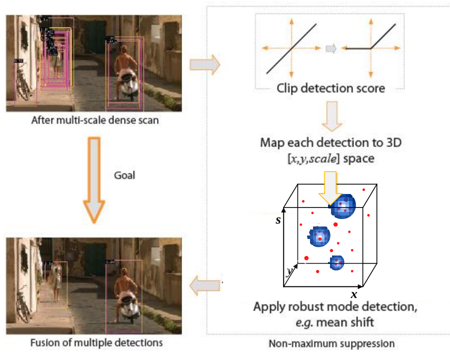


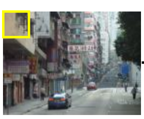
Image source: Navneet Dalal, PhD Thesis

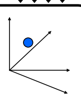
Recap: Sliding-Window Object Detection

Fleshing out this pipeline a bit more, we need to:

1. Obtain training data
2. Define features
3. Define classifier

Training examples





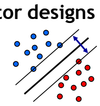
Feature extraction

Car/non-car Classifier

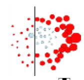
Slide credit: Kristen Grauman

Object Detector Design

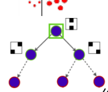
- In practice, the classifier often determines the design.
  - Types of features
  - Speedup strategies
- Today, we'll look at 3 state-of-the-art detector designs
  - Based on SVMs



  - Based on Boosting

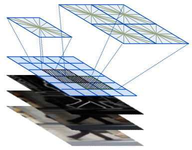


  - Based on Random Forests



Topics of This Lecture

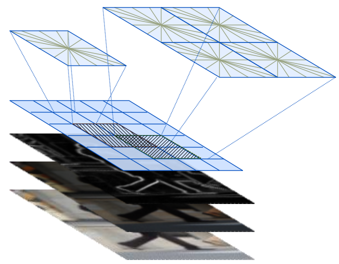
- Tracking by Detection
  - Motivation
  - Recap: Object detection
- **SVM based Detectors**
  - Recap: HOG
  - DPM
- AdaBoost based Detectors
  - Recap: Viola-Jones
  - Integral Channel features
  - VeryFast/Roerei
- Random Forest based Detectors
  - Recap: ISM
  - Hough Forests



Recap: Histograms of Oriented Gradients (HOG)

- Holistic object representation
  - Localized gradient orientations

[…, …, …, …]



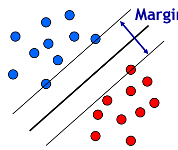
Object/Non-object
↑
Linear SVM
↑
Collect HOGs over detection window
↑
Contrast normalize over overlapping spatial cells
↑
Weighted vote in spatial & orientation cells
↑
Compute gradients
↑
Gamma compression
↑
Image Window

Slide adapted from Navneet Dalal

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

$$\mathbf{w}^{\mathrm{T}}\mathbf{x} + b = 0$$



- Formulation as a convex optimization problem
  - Find the hyperplane satisfying

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$$

under the constraints

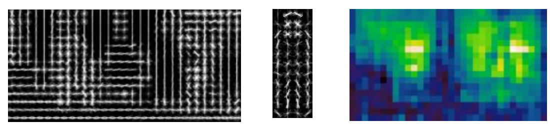
$$t_n(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n + b) \geq 1 \quad \forall n$$

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

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b$$



N. Dalal and B. Triggs, Histograms of Oriented Gradients for Human Detection, CVPR 2005

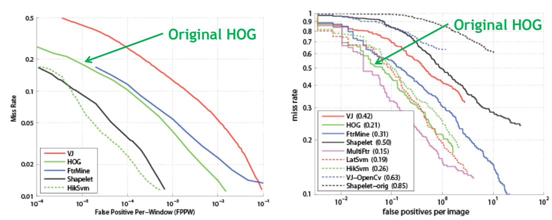
Slide credit: Svetlana Lazebnik

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(?)



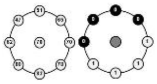
(a) INRIA per-window results. (b) INRIA per-image results.

- 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!

[Dollar et al., CVPR'09]

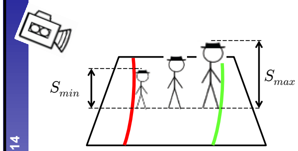
Some Extensions that Did Survive…

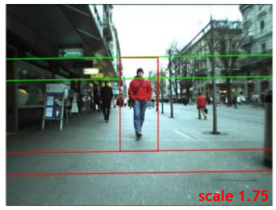
- HOG + LBP [Ojala & Pietikäinen 1999, Wang et al. '09]
  - Compute LBP histograms over cells, as in HOG
  - ⇒ Features seem to be complementary to some degree



- 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

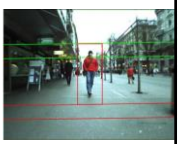




- Efficient integration into detector design (*groundHOG*)
  - Idea: only evaluate geometrically valid detection windows
  - Derivation: Region of interest lies between two parabolas…
  - …that can in most cases be approximated by straight lines.
  - ⇒ *Only touch pixels inside the ROI for all computations.*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

[P. Sudowe, B. Leibe, ICVS'11]

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:

run-time	1280 × 960		640 × 480	
	cuda	ground	cuda	ground
Laptop GTX 285M	1.6 fps	9.6 fps	7.2 fps	26 fps
Desktop GTX 280	5.5 fps	17.2 fps	22.7 fps	56 fps
Desktop GTX 580	9.8 fps	27.8 fps	41.6 fps	83 fps

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

[P. Sudowe, B. Leibe, ICVS'11]

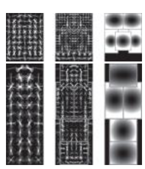
You Can Try It At Home…

- 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

P. Sudowe, B. Leibe, Efficient Use of Geometric Constraints for Sliding Window Object Detection in Video, ICVS 2011

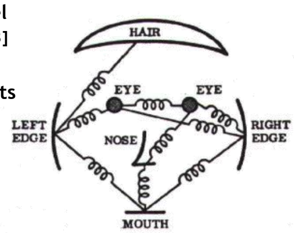
Topics of This Lecture

- 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



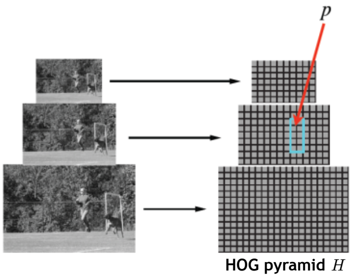
Recap: Part-Based Models

- 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



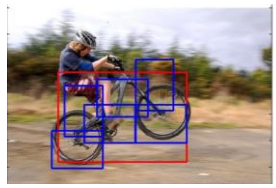
Filter F

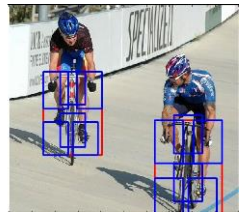
Score of F at position p is $F \cdot \phi(p, H)$

$\phi(p, H)$ = concatenation of HOG features from window specified by p.

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

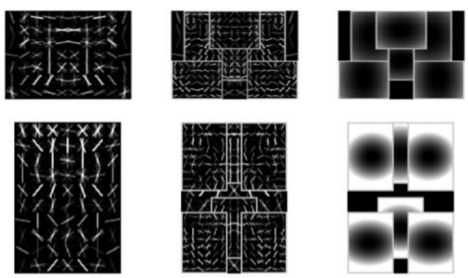
Deformable Part-based Models





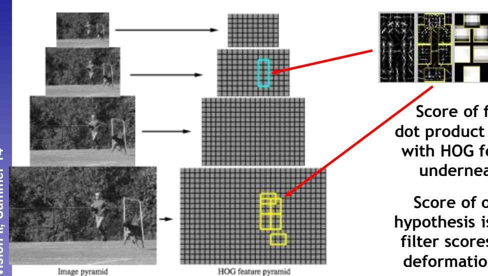
- Mixture of deformable part models (Pictorial Structures)
- Each component has global template + deformable parts
- Fully trained from bounding boxes alone

2-Component Bicycle Model



Root filters coarse resolution | Part filters finer resolution | Deformation models

Object Hypothesis



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

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

- Multiscale model captures features at two resolutions

Score of a Hypothesis

$$\text{score}(p_0, \dots, p_n) = \underbrace{\sum_{i=0}^{n} F_i \cdot \phi(H, p_i)}_{\text{“data term”}} - \underbrace{\sum_{i=1}^{n} d_i \cdot (dx_i^2, dy_i^2)}_{\text{“spatial prior”}}$$

filters — deformation parameters — displacements

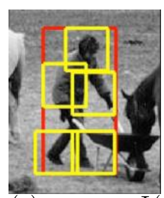
$$\text{score}(z) = \beta \cdot \Psi(H, z)$$

concatenation filters and deformation parameters

concatenation of HOG features and part displacement features

Recognition Model

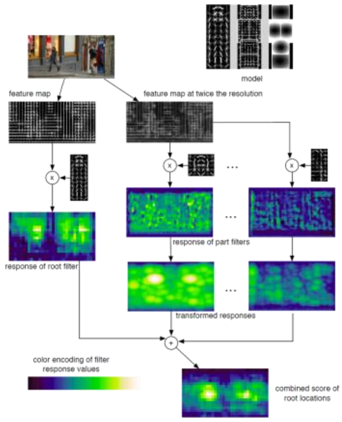
$f_w(x) = w \cdot \Phi(x)$



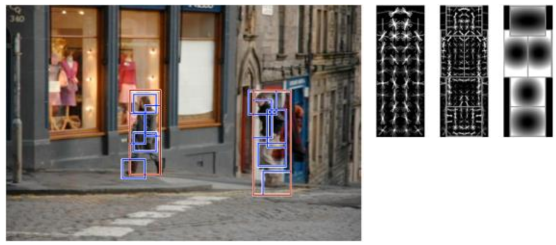
$f_w(x) = \max_z w \cdot \Phi(x, z)$

- 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

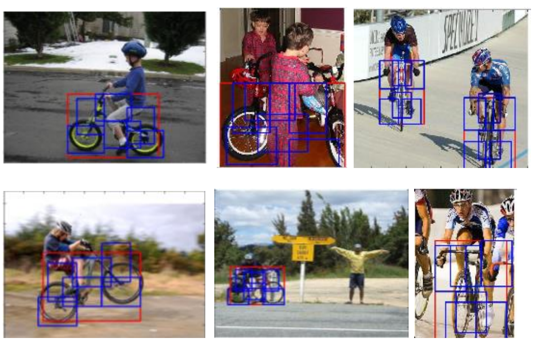


Results: Persons



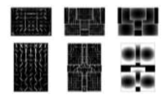
- Results (after non-maximum suppression)
  - ~1s to search all scales

Results: Bicycles



Extensions and Detailed Improvements

- 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

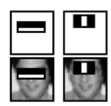


You Can Try It At Home…

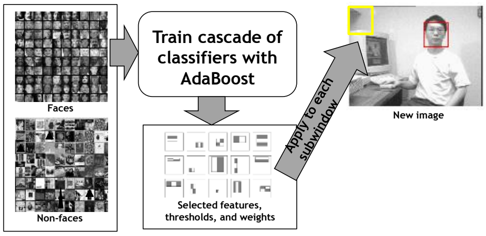
- Deformable part-based models have been very successful at several recent evaluations.
  ⇒ One of the state-of-the-art approaches in object detection
- Source code and models trained on PASCAL 2006, 2007, and 2008 data are available here:
  http://www.cs.uchicago.edu/~pff/latent

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



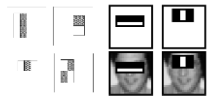
Recap: Viola-Jones Face Detector



- 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/]

Recap: Haar Wavelets

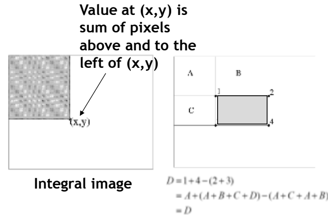
"Rectangular" filters



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



$D = 1 + 4 - (2 + 3)$
$= A + (A + B + C + D) - (A + C + A + B)$
$= D$

[Viola & Jones, CVPR 2001]

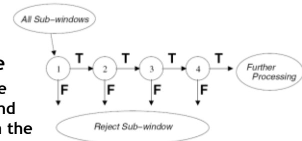
AdaBoost for Efficient Feature Selection

- 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

P. Viola, M. Jones, Robust Real-Time Face Detection, IJCV, Vol. 57(2), 2004. (first version appeared at CVPR 2001)

Recap: Cascading Classifiers for Detection

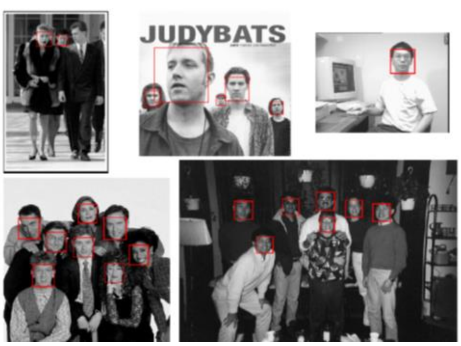
- Even if the filters are fast to compute, each new image has a lot of possible windows to search…
- Idea: Classifier cascade
  - Observation: most image windows are negative and look very different from the searched object class.
  - Filter for promising regions with an initial inexpensive classifier
  - 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]

Figure from Viola & Jones CVPR 2001

Viola-Jones Face Detector: Results



You Can Try It At Home…

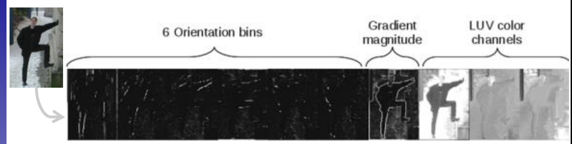
- The Viola & Jones detector was a huge success
  - First real-time face detector available
  - Many derivative works and improvements
- C++ implementation available in OpenCV [Lienhart, 2002]
  - http://sourceforge.net/projects/opencvlibrary/
- Matlab wrappers for OpenCV code available, e.g. here
  - http://www.mathworks.com/matlabcentral/fileexchange/19912

P. Viola, M. Jones, Robust Real-Time Face Detection, IJCV, Vol. 57(2), 2004

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

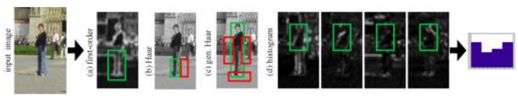
Integral Channel Features



- 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.

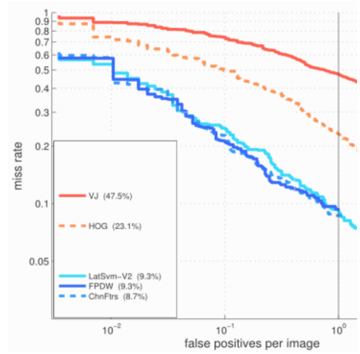
P. Dollar, Z. Tu, P. Perona, S. Belongie. Integral Channel Features, BMVC'09.

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



[Viola&Jones 2004]

fastHOG
~10 Hz on GPU
[Prisacariu 2009]

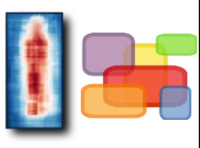
DPM
[Felzenszwalb 2008]

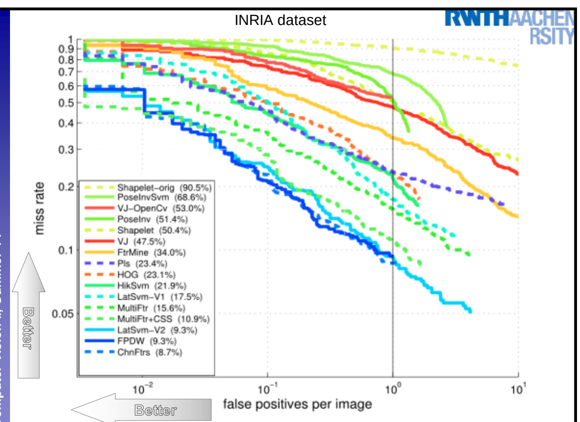
ChnFtrs/FPDW
~5 Hz on CPU
[Dollar 2009+2010]

Slide credit: Rodrigo Benenson

Topics of This Lecture

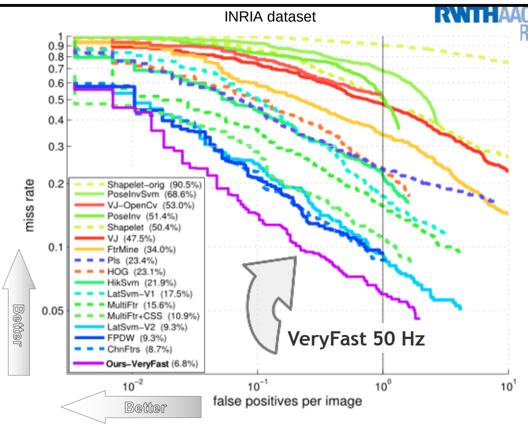
- Tracking by Detection
  - Motivation
  - Recap: Object detection
- SVM-based Detectors
  - Recap: HOG
  - DPM
- AdaBoost based Detectors
  - Recap: Viola-Jones
  - Integral Channel features
  - VeryFast/Roerei
- Random Forest based Detectors
  - Recap: ISM
  - Hough Forests





[Dollar et al. 2011]

Slide credit: Rodrigo Benenson



Issues for Efficient Detection

- One template cannot detect at multiple scales…

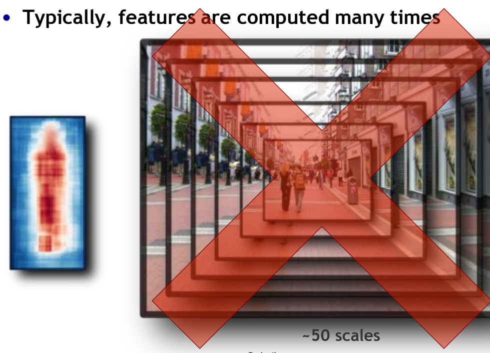
Issues for Efficient Detection

- Typically, features are computed many times

~50 scales

Issues for Efficient Detection

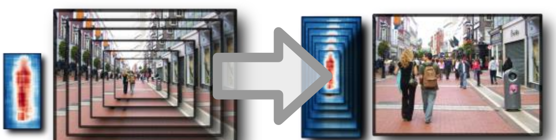
- Typically, features are computed many times



~50 scales

VeryFast Detector

- Idea 1: Invert the relation



1 model,
50 image scales

50 models,
1 image scale

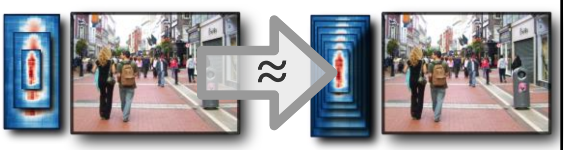
R. Benenson, M. Mathias, R. Timofte, L. Van Gool. [Pedestrian Detection at 100 Frames per Second](), CVPR'12.

Practical Considerations

- Training and running 1 model/scale is too expensive

VeryFast Detector

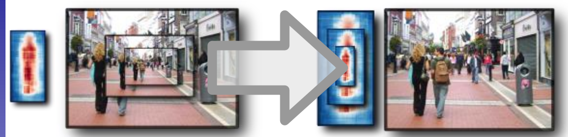
- Idea 2: Reduce training time by feature interpolation



- Shown to be possible for Integral Channel features
  - P. Dollár, S. Belongie, Perona. The Fastest Pedestrian Detector in the West, BMVC 2010.

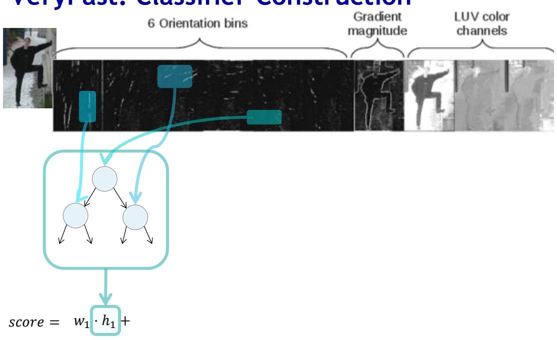
VeryFast Detector

- Effect: Transfer test time computation to training time



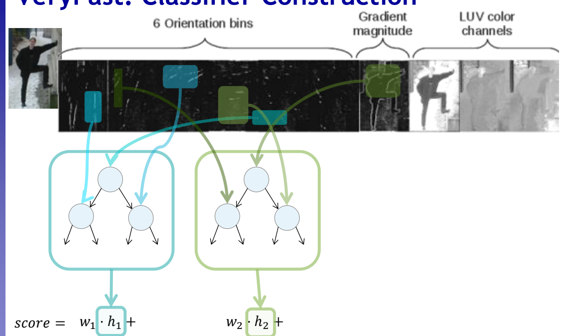
⇒ *Result: 3x reduction in feature computation*

VeryFast: Classifier Construction



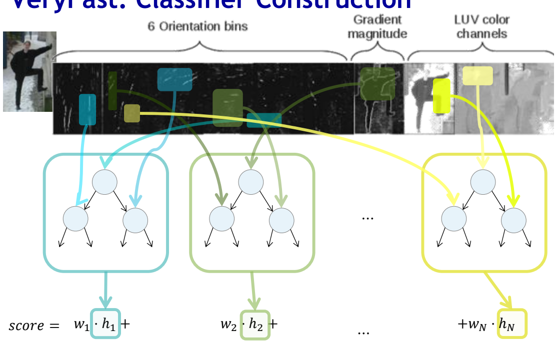
- Ensemble of short trees, learned by AdaBoost

VeryFast: Classifier Construction



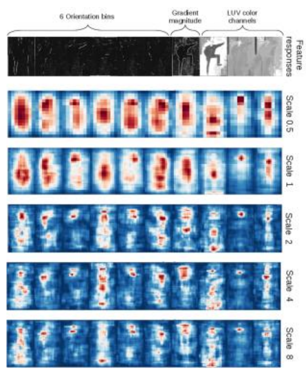
- Ensemble of short trees, learned by AdaBoost

VeryFast: Classifier Construction



- Ensemble of short trees, learned by AdaBoost

Learned Models

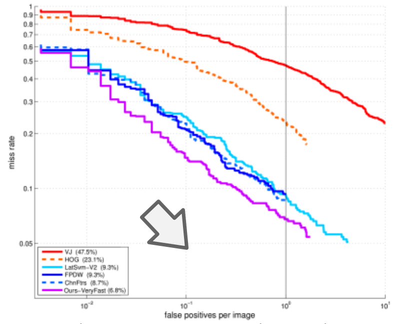


Integral Channel features

Models

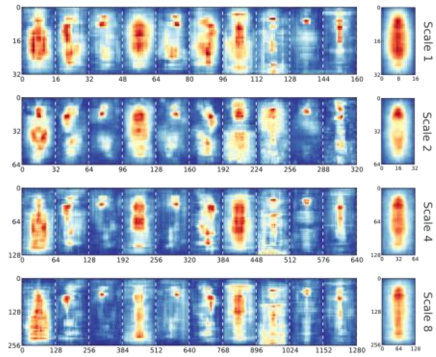
⋮

Results

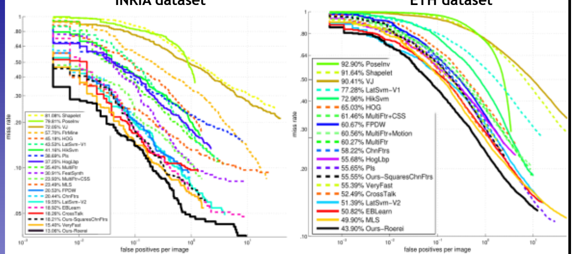


- Detection without resizing provides quality

Multi-Scale Models > Single-Scale Model

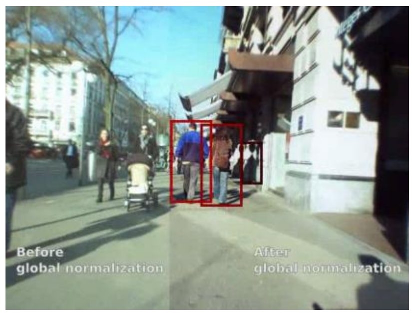


Comparison to State-of-the-Art



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

Roerei Results



R. Benenson, M. Mathias, R. Timofte, L. Van Gool. Seeking the Strongest Rigid Detector. CVPR'13.

Applications: Mobile Robot Navigation

link to the video

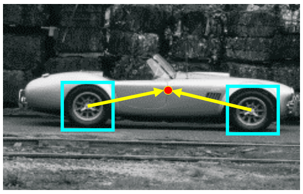
EUROPA

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"].



Training image

Visual codeword with displacement vectors

B. Leibe, A. Leonardis, and B. Schiele, Robust Object Detection with Interleaved Categorization and Segmentation, International Journal of Computer Vision, Vol. 77(1-3), 2008.

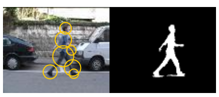
Recap: Implicit Shape Model (ISM) Idea

- Objects are detected as consistent configurations of the observed parts (visual words).

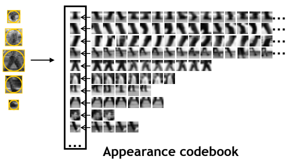
Test image

B. Leibe, A. Leonardis, and B. Schiele, Robust Object Detection with Interleaved Categorization and Segmentation, International Journal of Computer Vision, Vol. 77(1-3), 2008.

Recap: ISM - Representation

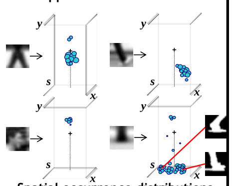


Training images
(+reference segmentation)



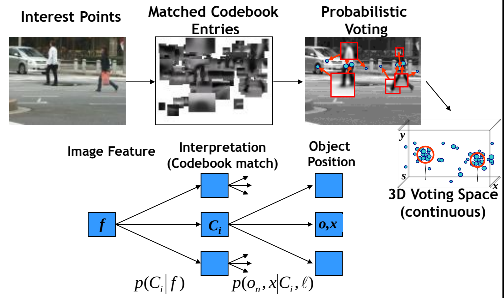
Appearance codebook

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



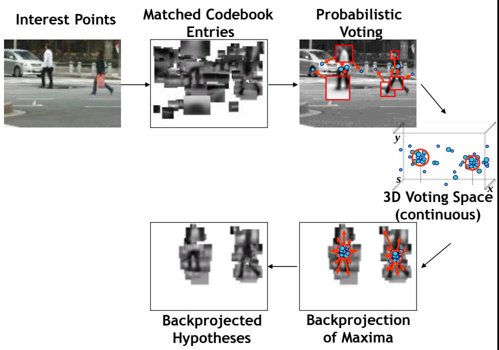
Spatial occurrence distributions
+ local figure-ground labels

Recap: ISM - Recognition

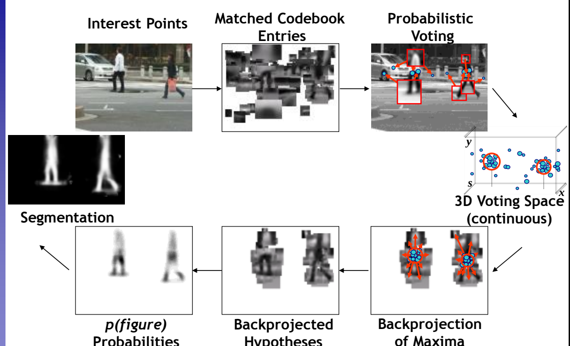


Probabilistic vote weighting

Recap: ISM - Recognition

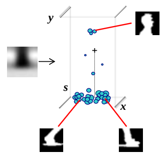


Recap: ISM - Top-Down Segmentation

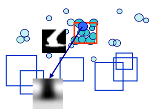


Class-Specific Top-Down Segmentation

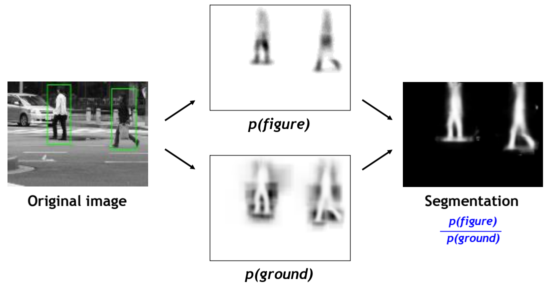
- During initial Hough Voting
  - When we first observe a feature, we do not know its context.
  - Different figure-ground labels may be consistent with the appearance.
  - ⇒ Strategy: we cast votes for many locations…



- 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.



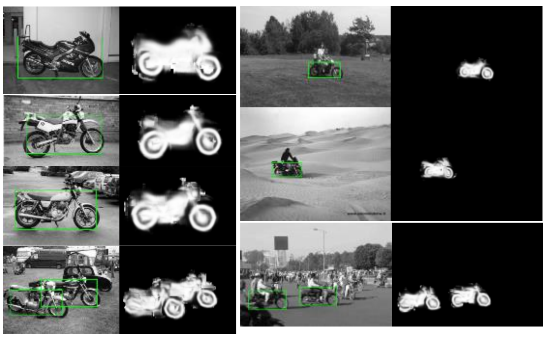
Top-Down Segmentation



- Interpretation of *p(figure)* map
  - per-pixel confidence in object hypothesis
  - Useful for hypothesis verification

[Leibe, Leonardis, Schiele, SLCV'04; IJCV'08]

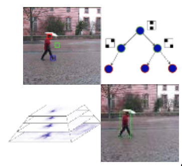
Recap: ISM - Example Results



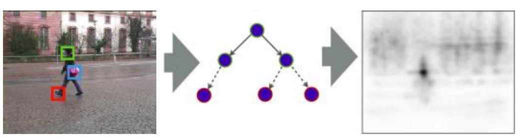
[Leibe, Leonardis, Schiele, SLCV'04; IJCV'08]

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 [Gall CVPR'09]

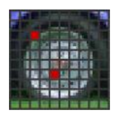


- Combine idea of ISM-style Hough voting with dense feature sampling and discriminative training.
  - Randomized forest classifier densely processes image patches
  - Leaf nodes correspond to visual words
  - Cast votes for possible object hypotheses
- Good empirical performance, fast to evaluate

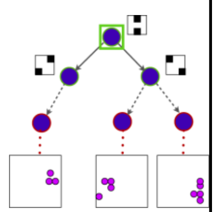
[Gall, Lempitsky, CVPR'09]

Fast Dense Matching with Random Forests

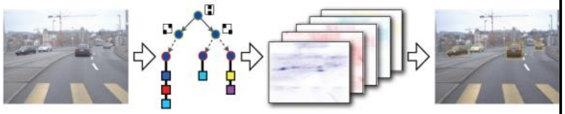
- Ideas
  - Solve feature extraction and codebook matching at the same time
  - Discriminative training of codebook features



- 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

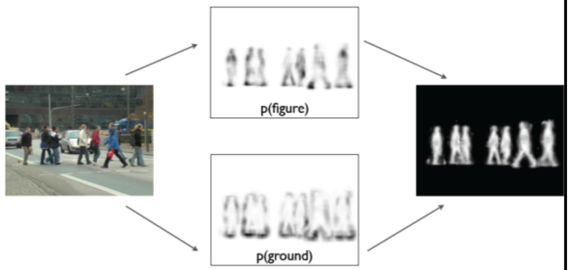


(a) Original image (b) Multi-view tree (c) Multi-view Hough spaces (d) Multi-view detection

- 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

[Razavi, Alvar, Gall, van Gool, CVPR'11; Rematas, Leibe, CORP'11]

Top-Down Segmentation with Hough Forests



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

[Rematas, Leibe, CORP'11]

HF-ISM: Qualitative Results

(no ground plane constraints used)

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

B. Leibe

[Rematas, Leibe, CORP'11]

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/