Graphical Models in Computer Vision

Gerard Pons-Moll

Max Planck Institute for Intelligent Systems
Perceiving Systems

June 27, 2016

MAX-PLANCK-GESELLSCHAFT

Syllabus

11.04.2016	Introduction
18.04.2016	Graphical Models 1
25.04.2016	Graphical Models 2 (Sand 6/7)
02.05.2016	Graphical Models 3
09.05.2016	Graphical Models 4
23.05.2016	Body Models 1
30.05.2016	Body Models 2
06.06.2016	Body Models 3
13.06.2016	Body Models 4
20.06.2016	Object Detection 1
27.06.2016	Object Detection 2
04.07.2016	Stereo
11.07.2016	Optical Flow
18.07.2016	Segmentation

Todays topic

Object Detection

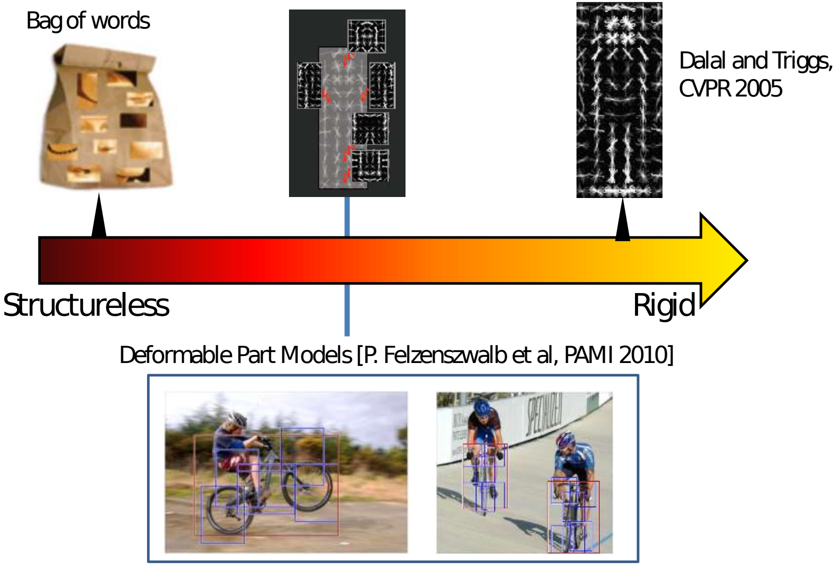
- Recap
- Part-based Models (DPM)

Object Tracking

- Introduction
- Bayes Filter
- Assignment Problem
- Graph-based Tracking

Part-based Models

Structureless vs. Rigid Models



Why Parts?

[Fergus, 2005]

Why do want to model parts?

- Useful to handle intra-class geometry variation

Why Parts?

[Fergus, 2005]

Why do want to model parts?

- Useful to handle intra-class geometry variation
- Objects may be globally different but they have parts in common

Why Parts?

[Fergus, 2005]

Why do want to model parts?

- Useful to handle intra-class geometry variation
- Objects may be globally different but they have parts in common
- Model prior knowledge of relative location and size

Why Parts?

Deformable parts can handle slight variations in pose:

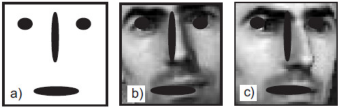


Figure 1. Matching with a single template. The schematic template of a frontal face is shown in a). Slight rotations of the face in the image plane b) and in depth c) lead to considerable discrepancies between template and face.

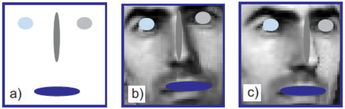
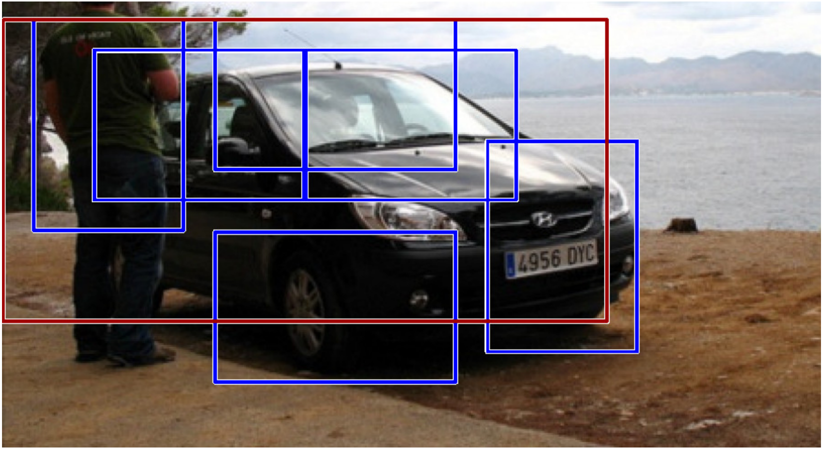


Figure 2. Matching with a set of component templates. The schematic component templates for a frontal face are shown in a). Shifting the component templates can compensate for slight rotations of the face in the image plane b) and in depth c).

[Heisele et al, 2001]

Why Parts?

Easier to handle occlusions:



[Felzenszwalb et al, 2010]

Connectivity Structures

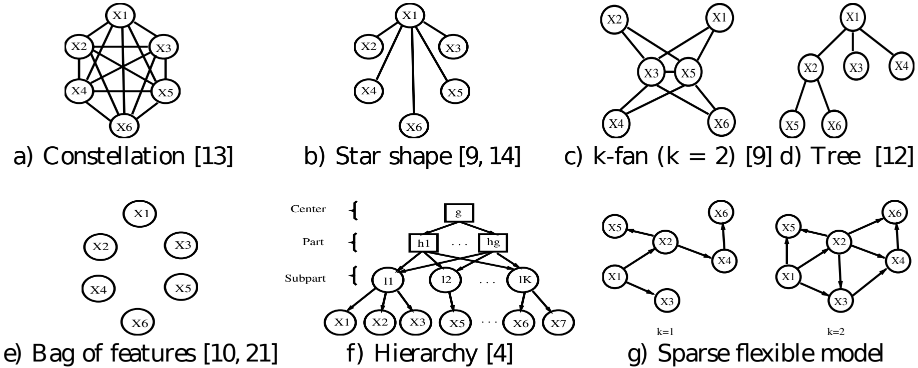
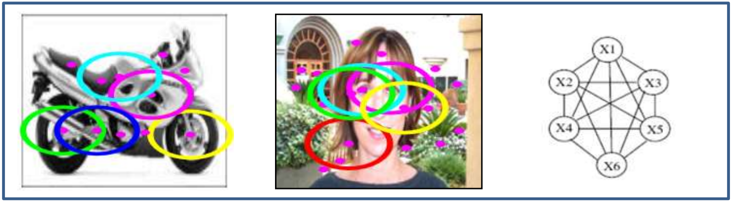


Fig. 1. Graphical geometric models of priors. Note that Xi represents a model part.

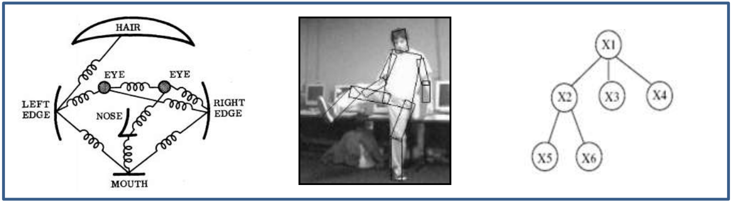
[Carneiro & Lowe, 2006]

Connectivity Structures

Constellation Model [Fergus et al, 2003]

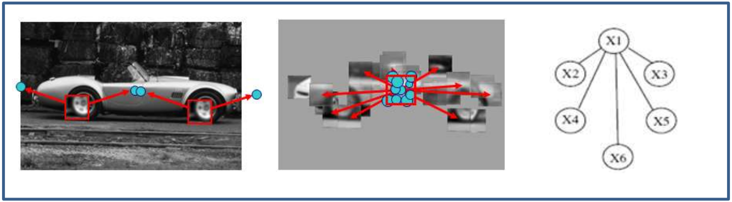


Efficient Pictorial Structures [Felzenszwalb & Huttenlocher, 2000]

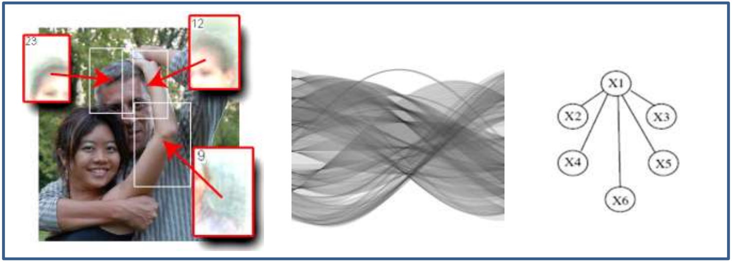


Connectivity Structures

Implicit Shape Model [Leibe et al, 2004]

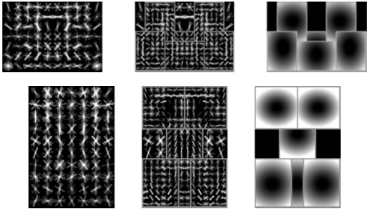


Poselets [Bourdev et al, 2009]



Deformable Part-based Model (DPM)

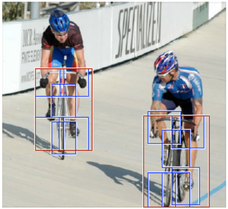
- 2-scale model
  - Whole object (root)
  - Deformable parts



root filters
coarse resolution

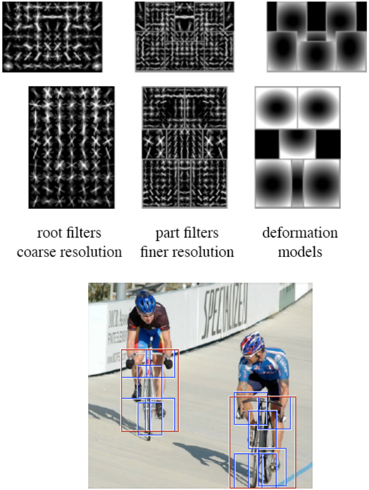
part filters
finer resolution

deformation
models



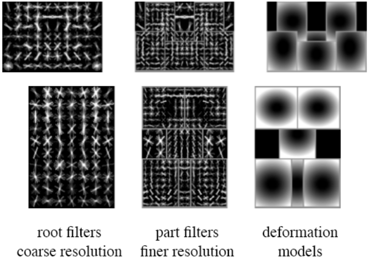
Deformable Part-based Model (DPM)

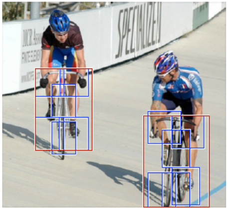
- 2-scale model
  - Whole object (root)
  - Deformable parts
- HoG representation + SVM training to obtain robust root and part detectors



Deformable Part-based Model (DPM)

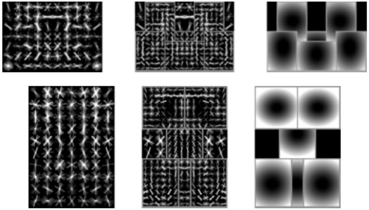
- 2-scale model
  - Whole object (root)
  - Deformable parts
- HoG representation + SVM training to obtain robust root and part detectors
- Efficient algorithm for detection



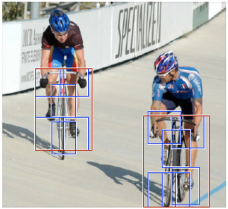


Deformable Part-based Model (DPM)

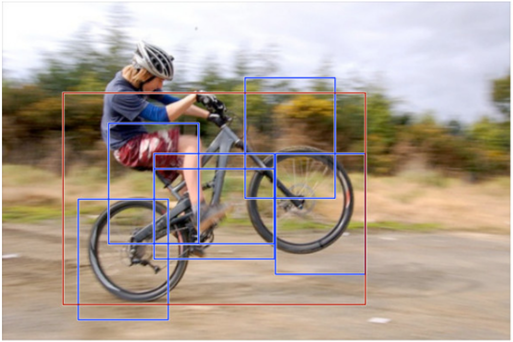
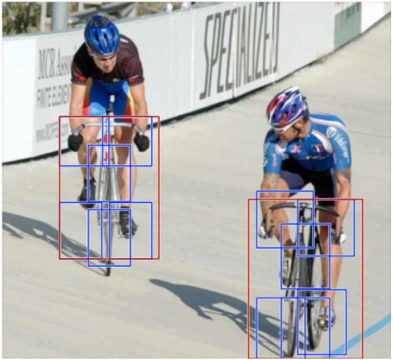
- 2-scale model
  - Whole object (root)
  - Deformable parts
- HoG representation + SVM training to obtain robust root and part detectors
- Efficient algorithm for detection
- [Felzenszwalb et al., 2010]



root filters coarse resolution | part filters finer resolution | deformation models

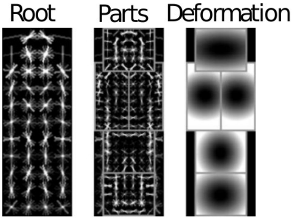


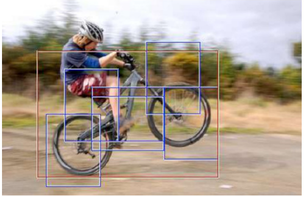
Deformable Part-based Model (DPM)

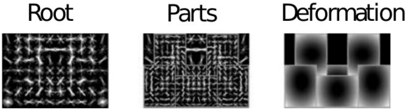
 

Models are fully trained from bounding boxes alone (weak labels).
The part locations are unknown (*i.e.*, latent variables).

DPM Pedestrian and Bicycle Model

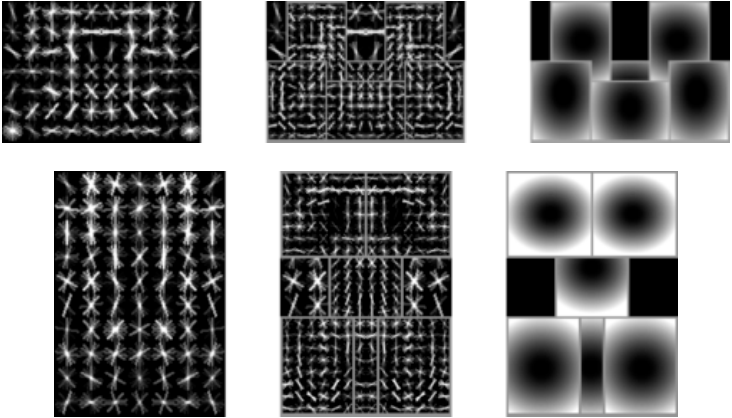






Different viewpoints are modeled using different models (=components).
Each component has a global template (root) + part templates.

DPM Bicycle Model with 2 Components



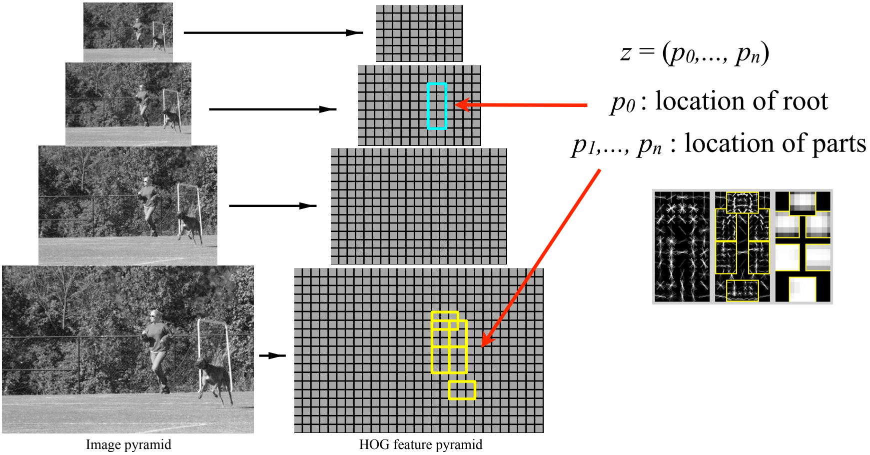
root filter
coarse resolution

part filter
finer resolution

deformation
models

Each component has a root filter F_0 and n part filters (F_i, v_i, d_i).

Multiscale Model captures Features at two Resolutions



The score is a sum of filter scores minus part deformation costs.

Score of a Hypothesis

The score is a sum of filter scores minus part deformation costs:

$$\text{score}(\mathbf{p}_0, \ldots, \mathbf{p}_n) = \sum_{i=0}^{n} \mathbf{f}_i^T \cdot \phi(\mathbf{p}_i) - \sum_{i=1}^{n} \mathbf{d}_i^T \cdot (dx_i^2, dy_i^2)$$

Score of a Hypothesis

The score is a sum of filter scores minus part deformation costs:

$$\text{score}(\mathbf{p}_0, \ldots, \mathbf{p}_n) = \sum_{i=0}^{n} \mathbf{f}_i^T \cdot \phi(\mathbf{p}_i) - \sum_{i=1}^{n} \mathbf{d}_i^T \cdot (dx_i^2, dy_i^2)$$

where:

- $\mathbf{p}_0, \ldots, \mathbf{p}_n$ denotes an object hypothesis, specified by root $(i = 0)$ and part $(i \geq 1)$ locations, with $\mathbf{p}_i = (x_i, y_i, l_i)^T$

Score of a Hypothesis

The score is a sum of filter scores minus part deformation costs:

$$\text{score}(\mathbf{p}_0, \dots, \mathbf{p}_n) = \sum_{i=0}^{n} \mathbf{f}_i^T \cdot \phi(\mathbf{p}_i) - \sum_{i=1}^{n} \mathbf{d}_i^T \cdot (dx_i^2, dy_i^2)$$

where:

- $\mathbf{p}_0, \dots, \mathbf{p}_n$ denotes an object hypothesis, specified by root ($i = 0$) and part ($i \geq 1$) locations, with $\mathbf{p}_i = (x_i, y_i, l_i)^T$
  - x_i, y_i denote pixel location
  - l_i specifies the level in the pyramid

Score of a Hypothesis

The score is a sum of filter scores minus part deformation costs:

$$\text{score}(\mathbf{p}_0, \ldots, \mathbf{p}_n) = \sum_{i=0}^{n} \mathbf{f}_i^T \cdot \phi(\mathbf{p}_i) - \sum_{i=1}^{n} \mathbf{d}_i^T \cdot (dx_i^2, dy_i^2)$$

where:

- $\mathbf{p}_0, \ldots, \mathbf{p}_n$ denotes an object hypothesis, specified by root $(i = 0)$ and part $(i \geq 1)$ locations, with $\mathbf{p}_i = (x_i, y_i, l_i)^T$
  - x_i, y_i denote pixel location
  - l_i specifies the level in the pyramid
- $\mathbf{f}_0, \ldots, \mathbf{f}_n$ are learned filter weights of the model

Score of a Hypothesis

The score is a sum of filter scores minus part deformation costs:

$$\text{score}(\mathbf{p}_0, \ldots, \mathbf{p}_n) = \sum_{i=0}^{n} \mathbf{f}_i^T \cdot \phi(\mathbf{p}_i) - \sum_{i=1}^{n} \mathbf{d}_i^T \cdot (dx_i^2, dy_i^2)$$

where:

- $\mathbf{p}_0, \ldots, \mathbf{p}_n$ denotes an object hypothesis, specified by root ($i = 0$) and part ($i \geq 1$) locations, with $\mathbf{p}_i = (x_i, y_i, l_i)^T$
  - x_i, y_i denote pixel location
  - l_i specifies the level in the pyramid
- $\mathbf{f}_0, \ldots, \mathbf{f}_n$ are learned filter weights of the model
- $\phi(\mathbf{p}_i)$ are the HoG features for the region specified by $\mathbf{p}_i$

Score of a Hypothesis

The score is a sum of filter scores minus part deformation costs:

$$\text{score}(\mathbf{p}_0, \ldots, \mathbf{p}_n) = \sum_{i=0}^{n} \mathbf{f}_i^T \cdot \phi(\mathbf{p}_i) - \sum_{i=1}^{n} \mathbf{d}_i^T \cdot (dx_i^2, dy_i^2)$$

where:

- $\mathbf{p}_0, \ldots, \mathbf{p}_n$ denotes an object hypothesis, specified by root $(i = 0)$ and part $(i \geq 1)$ locations, with $\mathbf{p}_i = (x_i, y_i, l_i)^T$
  - x_i, y_i denote pixel location
  - l_i specifies the level in the pyramid
- $\mathbf{f}_0, \ldots, \mathbf{f}_n$ are learned filter weights of the model
- $\phi(\mathbf{p}_i)$ are the HoG features for the region specified by $\mathbf{p}_i$
- $\mathbf{d}_i \in \mathbb{R}^2$ are deformation parameters

Score of a Hypothesis

The score is a sum of filter scores minus part deformation costs:

$$\text{score}(\mathbf{p}_0, \ldots, \mathbf{p}_n) = \sum_{i=0}^{n} \mathbf{f}_i^T \cdot \phi(\mathbf{p}_i) - \sum_{i=1}^{n} \mathbf{d}_i^T \cdot (dx_i^2, dy_i^2)$$

where:

- $\mathbf{p}_0, \ldots, \mathbf{p}_n$ denotes an object hypothesis, specified by root ($i = 0$) and part ($i \geq 1$) locations, with $\mathbf{p}_i = (x_i, y_i, l_i)^T$
  - x_i, y_i denote pixel location
  - l_i specifies the level in the pyramid
- $\mathbf{f}_0, \ldots, \mathbf{f}_n$ are learned filter weights of the model
- $\phi(\mathbf{p}_i)$ are the HoG features for the region specified by $\mathbf{p}_i$
- $\mathbf{d}_i \in \mathbb{R}^2$ are deformation parameters
- dx_i, dy_i denote the rel. displacement of $\mathbf{p}_i$ from its anchor $\mathbf{v}_i$:

$$(dx_i, dy_i) = (x_i, y_i) - (2(x_0, y_0) + \mathbf{v}_i)$$

Score of a Hypothesis

DPM score from previous slide:

$$\text{score}(\mathbf{p}_0, \ldots, \mathbf{p}_n) = \sum_{i=0}^{n} \mathbf{f}_i^T \cdot \phi(\mathbf{p}_i) - \sum_{i=1}^{n} \mathbf{d}_i^T \cdot (dx_i^2, dy_i^2)$$

Score of a Hypothesis

DPM score from previous slide:

$$\text{score}(\mathbf{p}_0, \ldots, \mathbf{p}_n) = \sum_{i=0}^{n} \mathbf{f}_i^T \cdot \phi(\mathbf{p}_i) - \sum_{i=1}^{n} \mathbf{d}_i^T \cdot (dx_i^2, dy_i^2)$$

This can be also written as a linear combination

$$\text{score}(\mathbf{p}_0, \ldots, \mathbf{p}_n) = \boldsymbol{\beta}^T \cdot \psi(\mathbf{p}_0, \ldots, \mathbf{p}_n)$$

Score of a Hypothesis

DPM score from previous slide:

$$\text{score}(\mathbf{p}_0, \ldots, \mathbf{p}_n) = \sum_{i=0}^{n} \mathbf{f}_i^T \cdot \phi(\mathbf{p}_i) - \sum_{i=1}^{n} \mathbf{d}_i^T \cdot (dx_i^2, dy_i^2)$$

This can be also written as a linear combination

$$\text{score}(\mathbf{p}_0, \ldots, \mathbf{p}_n) = \boldsymbol{\beta}^T \cdot \psi(\mathbf{p}_0, \ldots, \mathbf{p}_n)$$

where:

- $\boldsymbol{\beta} = (\mathbf{f}_0, \ldots, \mathbf{f}_n, \mathbf{d}_1, \ldots, \mathbf{d}_n)$

Score of a Hypothesis

DPM score from previous slide:

$$\text{score}(\mathbf{p}_0, \ldots, \mathbf{p}_n) = \sum_{i=0}^{n} \mathbf{f}_i^T \cdot \phi(\mathbf{p}_i) - \sum_{i=1}^{n} \mathbf{d}_i^T \cdot (dx_i^2, dy_i^2)$$

This can be also written as a linear combination

$$\text{score}(\mathbf{p}_0, \ldots, \mathbf{p}_n) = \boldsymbol{\beta}^T \cdot \psi(\mathbf{p}_0, \ldots, \mathbf{p}_n)$$

where:

- $\boldsymbol{\beta} = (\mathbf{f}_0, \ldots, \mathbf{f}_n, \mathbf{d}_1, \ldots, \mathbf{d}_n)$
- $\psi(\mathbf{p}_0, \ldots, \mathbf{p}_n) = (\phi(\mathbf{p}_0), \ldots, \phi(\mathbf{p}_n), -(dx_1^2, dy_1^2), \ldots, -(dx_n^2, dy_n^2))$

Score of a Hypothesis

DPM score from previous slide:

$$\text{score}(\mathbf{p}_0, \dots, \mathbf{p}_n) = \sum_{i=0}^{n} \mathbf{f}_i^T \cdot \phi(\mathbf{p}_i) - \sum_{i=1}^{n} \mathbf{d}_i^T \cdot (dx_i^2, dy_i^2)$$

This can be also written as a linear combination

$$\text{score}(\mathbf{p}_0, \dots, \mathbf{p}_n) = \boldsymbol{\beta}^T \cdot \psi(\mathbf{p}_0, \dots, \mathbf{p}_n)$$

where:

- $\boldsymbol{\beta} = (\mathbf{f}_0, \dots, \mathbf{f}_n, \mathbf{d}_1, \dots, \mathbf{d}_n)$
- $\psi(\mathbf{p}_0, \dots, \mathbf{p}_n) = (\phi(\mathbf{p}_0), \dots, \phi(\mathbf{p}_n), -(dx_1^2, dy_1^2), \dots, -(dx_n^2, dy_n^2))$

This illustrates the connection to linear classifiers:
The DPM learns the model parameters using the latent SVM framework.

Object Detection with DPM

Inference: Given a root location $\mathbf{p}_0$, calculate the detection score as:

$$
\begin{aligned}
\text{score}(\mathbf{p}_0) &= \max_{\mathbf{p}_1,\ldots,\mathbf{p}_n} \text{score}(\mathbf{p}_0,\ldots,\mathbf{p}_n) \\
&= \max_{\mathbf{p}_1,\ldots,\mathbf{p}_n} \boldsymbol{\beta}^T \cdot \psi(\mathbf{p}_0,\ldots,\mathbf{p}_n) \\
&= \max_{\mathbf{z}} \boldsymbol{\beta}^T \cdot \psi(\mathbf{p}_0,\mathbf{z})
\end{aligned}
$$

Object Detection with DPM

Inference: Given a root location $\mathbf{p}_0$, calculate the detection score as:

$$
\begin{aligned}
\text{score}(\mathbf{p}_0) &= \max_{\mathbf{p}_1,\dots,\mathbf{p}_n} \text{score}(\mathbf{p}_0,\dots,\mathbf{p}_n) \\
&= \max_{\mathbf{p}_1,\dots,\mathbf{p}_n} \boldsymbol{\beta}^T \cdot \psi(\mathbf{p}_0,\dots,\mathbf{p}_n) \\
&= \max_{\mathbf{z}} \boldsymbol{\beta}^T \cdot \psi(\mathbf{p}_0,\mathbf{z})
\end{aligned}
$$

- This maximizes the score by varying the parts $\mathbf{z} = (\mathbf{p}_1,\dots,\mathbf{p}_n)^T$ given the root location $\mathbf{p}_0$

Object Detection with DPM

Inference: Given a root location $\mathbf{p}_0$, calculate the detection score as:

$$\begin{aligned}
\text{score}(\mathbf{p}_0) &= \max_{\mathbf{p}_1,\dots,\mathbf{p}_n} \text{score}(\mathbf{p}_0,\dots,\mathbf{p}_n) \\
&= \max_{\mathbf{p}_1,\dots,\mathbf{p}_n} \boldsymbol{\beta}^T \cdot \psi(\mathbf{p}_0,\dots,\mathbf{p}_n) \\
&= \max_{\mathbf{z}} \boldsymbol{\beta}^T \cdot \psi(\mathbf{p}_0,\mathbf{z})
\end{aligned}$$

- This maximizes the score by varying the parts $\mathbf{z} = (\mathbf{p}_1,\dots,\mathbf{p}_n)^T$ given the root location $\mathbf{p}_0$
- High scoring root locations define detections

Object Detection with DPM

Inference: Given a root location $\mathbf{p}_0$, calculate the detection score as:

$$\begin{aligned}
\text{score}(\mathbf{p}_0) &= \max_{\mathbf{p}_1,\ldots,\mathbf{p}_n} \text{score}(\mathbf{p}_0,\ldots,\mathbf{p}_n) \\
&= \max_{\mathbf{p}_1,\ldots,\mathbf{p}_n} \boldsymbol{\beta}^T \cdot \psi(\mathbf{p}_0,\ldots,\mathbf{p}_n) \\
&= \max_{\mathbf{z}} \boldsymbol{\beta}^T \cdot \psi(\mathbf{p}_0,\mathbf{z})
\end{aligned}$$

- This maximizes the score by varying the parts $\mathbf{z} = (\mathbf{p}_1,\ldots,\mathbf{p}_n)^T$ given the root location $\mathbf{p}_0$
- High scoring root locations define detections
- This maximization (which is exponential in the number of parts n) can be efficiently computed using dynamic programming and generalized distance transforms

Object Detection with DPM

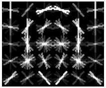
Inference: Given a root location $\mathbf{p}_0$, calculate the detection score as:

$$\begin{aligned}\text{score}(\mathbf{p}_0) &= \max_{\mathbf{p}_1,\dots,\mathbf{p}_n} \text{score}(\mathbf{p}_0,\dots,\mathbf{p}_n) \\ &= \max_{\mathbf{p}_1,\dots,\mathbf{p}_n} \boldsymbol{\beta}^T \cdot \psi(\mathbf{p}_0,\dots,\mathbf{p}_n) \\ &= \max_{\mathbf{z}} \boldsymbol{\beta}^T \cdot \psi(\mathbf{p}_0,\mathbf{z})\end{aligned}$$

- This maximizes the score by varying the parts $\mathbf{z} = (\mathbf{p}_1,\dots,\mathbf{p}_n)^T$ given the root location $\mathbf{p}_0$
- High scoring root locations define detections
- This maximization (which is exponential in the number of parts n) can be efficiently computed using dynamic programming and generalized distance transforms
- Which graphical model/inference problem do we have here?

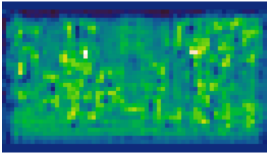
Fast Evaluation of Filter Responses

Head filter

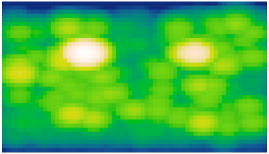


Input image

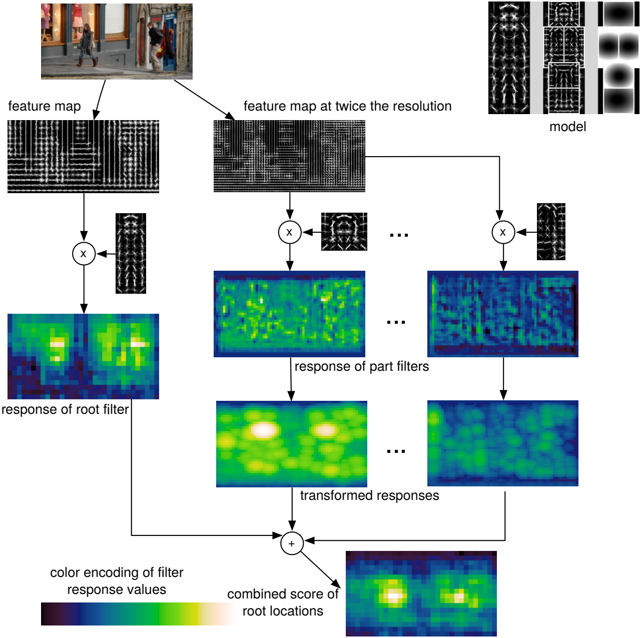
Filter response at level l:
$R_l(x, y) = \mathbf{f}^T \cdot \phi(x, y, l)$



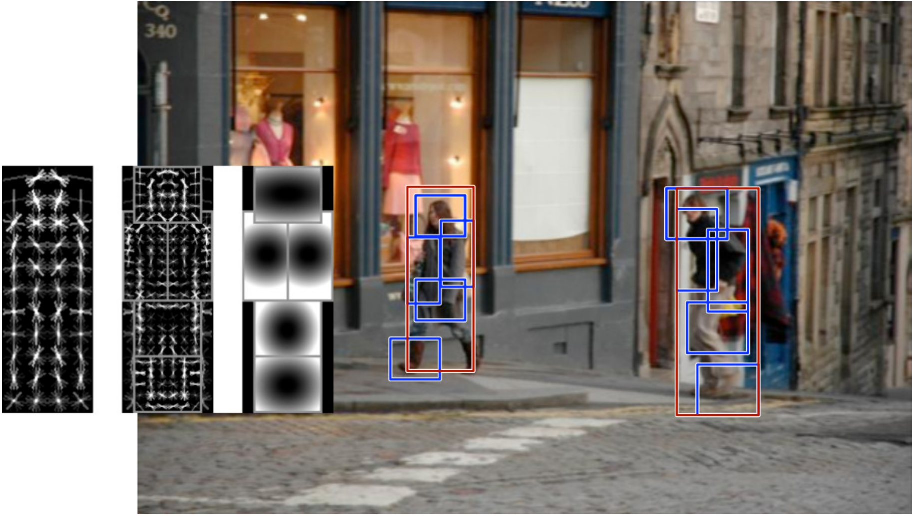
Transformed response:
$D_l(x, y) =$
$\max_{dx,dy}(R_l(x + dx, y + dy) - \mathbf{d}^T \cdot (dx^2, dy^2))$



Pipeline

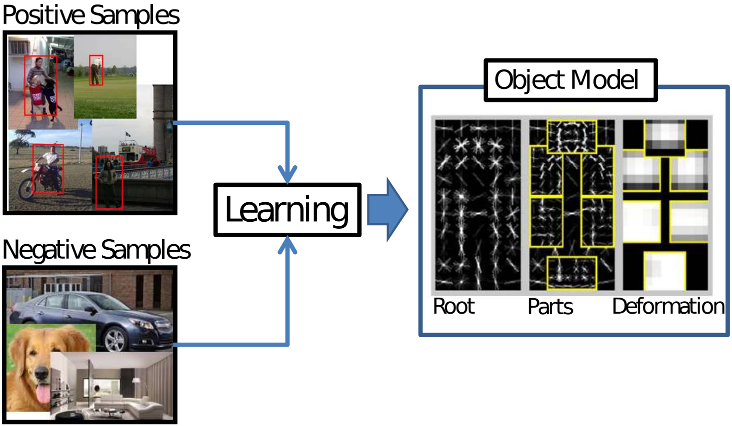

feature map
feature map at twice the resolution
model
x
x
x
...
...
response of part filters
response of root filter
...
transformed responses
+
color encoding of filter response values
combined score of root locations

Detection Results



Detection results after non-maxima-suppression (mode finding)

Training a DPM Detector (Parameter Estimation)



Given annotated images and background images we need to find:

- Root and part filter weights
- Deformation weights

Latent SVM Training

Learn a classifier that scores an example $\mathbf{p}_0$ as

$$\text{score}_{\boldsymbol{\beta}}(\mathbf{p}_0) = \max_{\mathbf{z}} \boldsymbol{\beta}^T \cdot \psi(\mathbf{p}_0, \mathbf{z})$$

Latent SVM Training

Learn a classifier that scores an example $\mathbf{p}_0$ as

$$\text{score}_{\boldsymbol{\beta}}(\mathbf{p}_0) = \max_{\mathbf{z}} \boldsymbol{\beta}^T \cdot \psi(\mathbf{p}_0, \mathbf{z})$$

where

- $\boldsymbol{\beta}$ are the model parameters from before (filter and deformation weights)

Latent SVM Training

Learn a classifier that scores an example $\mathbf{p}_0$ as

$$\text{score}_{\boldsymbol{\beta}}(\mathbf{p}_0) = \max_{\mathbf{z}} \boldsymbol{\beta}^T \cdot \psi(\mathbf{p}_0, \mathbf{z})$$

where

- $\boldsymbol{\beta}$ are the model parameters from before (filter and deformation weights)
- $\mathbf{z} = (\mathbf{p}_1, \ldots, \mathbf{p}_n)$ are latent values

Latent SVM Training

Learn a classifier that scores an example $\mathbf{p}_0$ as

$$\text{score}_{\boldsymbol{\beta}}(\mathbf{p}_0) = \max_{\mathbf{z}} \boldsymbol{\beta}^T \cdot \psi(\mathbf{p}_0, \mathbf{z})$$

where

- $\boldsymbol{\beta}$ are the model parameters from before (filter and deformation weights)
- $\mathbf{z} = (\mathbf{p}_1, \ldots, \mathbf{p}_n)$ are latent values
- Training data: $\{(\mathbf{p}_0^{(1)}, y^{(1)}), \ldots, (\mathbf{p}_0^{(n)}, y^{(n)})\}$ with $y^{(i)} \in \{-1, +1\}$

Latent SVM Training

Learn a classifier that scores an example $\mathbf{p}_0$ as

$$\text{score}_{\boldsymbol{\beta}}(\mathbf{p}_0) = \max_{\mathbf{z}} \boldsymbol{\beta}^T \cdot \psi(\mathbf{p}_0, \mathbf{z})$$

where

- $\boldsymbol{\beta}$ are the model parameters from before (filter and deformation weights)
- $\mathbf{z} = (\mathbf{p}_1, \ldots, \mathbf{p}_n)$ are latent values
- Training data: $\{(\mathbf{p}_0^{(1)}, y^{(1)}), \ldots, (\mathbf{p}_0^{(n)}, y^{(n)})\}$ with $y^{(i)} \in \{-1, +1\}$
- We want to find $\boldsymbol{\beta}$ such that: $y^{(i)} \cdot \text{score}_{\boldsymbol{\beta}}(\mathbf{p}_0^{(i)}) > 0$

Latent SVM Training

Learn a classifier that scores an example $\mathbf{p}_0$ as

$$\text{score}_{\boldsymbol{\beta}}(\mathbf{p}_0) = \max_{\mathbf{z}} \boldsymbol{\beta}^T \cdot \psi(\mathbf{p}_0, \mathbf{z})$$

where

- $\boldsymbol{\beta}$ are the model parameters from before (filter and deformation weights)
- $\mathbf{z} = (\mathbf{p}_1, \ldots, \mathbf{p}_n)$ are latent values
- Training data: $\{(\mathbf{p}_0^{(1)}, y^{(1)}), \ldots, (\mathbf{p}_0^{(n)}, y^{(n)})\}$ with $y^{(i)} \in \{-1, +1\}$
- We want to find $\boldsymbol{\beta}$ such that: $y^{(i)} \cdot \text{score}_{\boldsymbol{\beta}}(\mathbf{p}_0^{(i)}) > 0$
- Positive examples: pos. score, negative examples: neg. score

Latent SVM Training

Learn a classifier that scores an example $\mathbf{p}_0$ as

$$\text{score}_{\boldsymbol{\beta}}(\mathbf{p}_0) = \max_{\mathbf{z}} \boldsymbol{\beta}^T \cdot \psi(\mathbf{p}_0, \mathbf{z})$$

where

- $\boldsymbol{\beta}$ are the model parameters from before (filter and deformation weights)
- $\mathbf{z} = (\mathbf{p}_1, \ldots, \mathbf{p}_n)$ are latent values
- Training data: $\{(\mathbf{p}_0^{(1)}, y^{(1)}), \ldots, (\mathbf{p}_0^{(n)}, y^{(n)})\}$ with $y^{(i)} \in \{-1, +1\}$
- We want to find $\boldsymbol{\beta}$ such that: $y^{(i)} \cdot \text{score}_{\boldsymbol{\beta}}(\mathbf{p}_0^{(i)}) > 0$
- Positive examples: pos. score, negative examples: neg. score

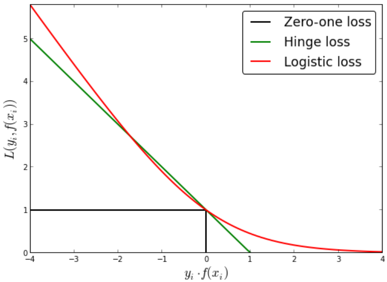
We minimize the following regularized latent SVM objective:

$$L_D(\boldsymbol{\beta}) = \underbrace{\frac{1}{2}\|\boldsymbol{\beta}\|^2}_{\text{regularizer}} + C \sum_{i=1}^{n} \underbrace{\max\left(0, 1 - y^{(i)} \cdot \text{score}_{\boldsymbol{\beta}}(\mathbf{p}_0^{(i)})\right)}_{\text{loss function}}$$

Loss Functions

$$L_D(\beta) = \underbrace{\frac{1}{2}\|\beta\|^2}_{\text{regularizer}} + C\sum_{i=1}^{n} \underbrace{\max\left(0, 1 - y^{(i)} \cdot \text{score}_\beta(\mathbf{p}_0^{(i)})\right)}_{\text{loss function}}$$

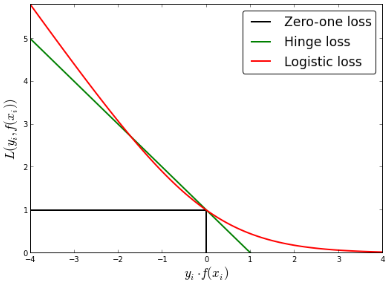
- Hinge and logistic loss approximate the missclassification error! (they are upper bounds)



Loss Functions

$$L_D(\beta) = \underbrace{\frac{1}{2}\|\beta\|^2}_{\text{regularizer}} + C\sum_{i=1}^{n} \underbrace{\max\left(0, 1 - y^{(i)} \cdot \text{score}_\beta(\mathbf{p}_0^{(i)})\right)}_{\text{loss function}}$$

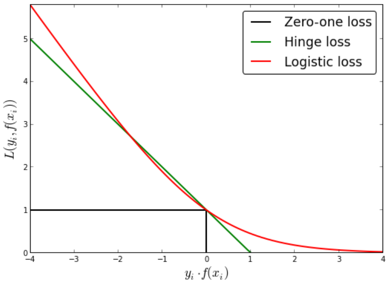
- Hinge and logistic loss approximate the missclassification error! (they are upper bounds)
- Important properties:
  - Robustness



Loss Functions

$$L_D(\beta) = \underbrace{\frac{1}{2}\|\beta\|^2}_{\text{regularizer}} + C\sum_{i=1}^{n} \underbrace{\max\left(0, 1 - y^{(i)} \cdot \text{score}_\beta(\mathbf{p}_0^{(i)})\right)}_{\text{loss function}}$$

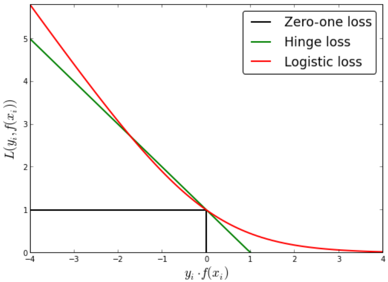
- Hinge and logistic loss approximate the missclassification error! (they are upper bounds)
- Important properties:
  - Robustness
  - Convexity



Loss Functions

$$L_D(\beta) = \underbrace{\frac{1}{2}\|\beta\|^2}_{\text{regularizer}} + C \sum_{i=1}^{n} \underbrace{\max\left(0, 1 - y^{(i)} \cdot \text{score}_\beta(\mathbf{p}_0^{(i)})\right)}_{\text{loss function}}$$

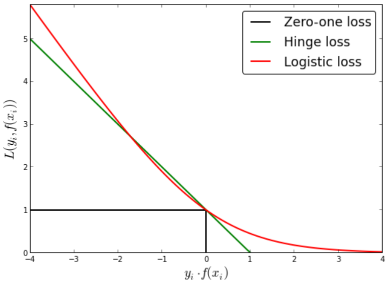
- Hinge and logistic loss approximate the missclassification error! (they are upper bounds)
- Important properties:
  - Robustness
  - Convexity
  - Smoothness



Loss Functions

$$L_D(\beta) = \underbrace{\frac{1}{2}\|\beta\|^2}_{\text{regularizer}} + C \sum_{i=1}^{n} \underbrace{\max\left(0, 1 - y^{(i)} \cdot \text{score}_\beta(\mathbf{p}_0^{(i)})\right)}_{\text{loss function}}$$

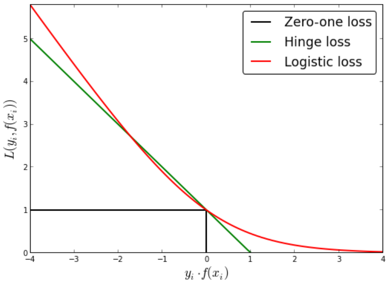
- Hinge and logistic loss approximate the missclassification error! (they are upper bounds)
- Important properties:
  - Robustness
  - Convexity
  - Smoothness
- 0-1 loss NP hard



Loss Functions

$$L_D(\beta) = \underbrace{\frac{1}{2}\|\beta\|^2}_{\text{regularizer}} + C\sum_{i=1}^{n} \underbrace{\max\left(0, 1 - y^{(i)} \cdot \text{score}_\beta(\mathbf{p}_0^{(i)})\right)}_{\text{loss function}}$$

- Hinge and logistic loss approximate the missclassification error! (they are upper bounds)
- Important properties:
  - Robustness
  - Convexity
  - Smoothness
- 0-1 loss NP hard
- SVM uses Hinge loss



Semi Convexity

We guaranteed to the find minimizer $\boldsymbol{\beta}^* = \operatorname{argmin}_{\boldsymbol{\beta}} L_D(\boldsymbol{\beta})$ using gradient decent if and exactly if $L_D(\boldsymbol{\beta})$ is convex! Is $L_D(\boldsymbol{\beta})$ convex?

Semi Convexity

We guaranteed to the find minimizer $\boldsymbol{\beta}^* = \operatorname{argmin}_{\boldsymbol{\beta}} L_D(\boldsymbol{\beta})$ using gradient decent if and exactly if $L_D(\boldsymbol{\beta})$ is convex! Is $L_D(\boldsymbol{\beta})$ convex?

- The sum and maximum of convex functions is convex

Semi Convexity

We guaranteed to the find minimizer $\boldsymbol{\beta}^* = \text{argmin}_{\boldsymbol{\beta}} L_D(\boldsymbol{\beta})$ using gradient decent if and exactly if $L_D(\boldsymbol{\beta})$ is convex! Is $L_D(\boldsymbol{\beta})$ convex?

- The sum and maximum of convex functions is convex
- $\text{score}_{\boldsymbol{\beta}}(\mathbf{p}_0) = \max_{\mathbf{z}} \boldsymbol{\beta}^T \cdot \psi(\mathbf{p}_0, \mathbf{z})$ is convex in $\boldsymbol{\beta}$! Why?

Semi Convexity

We guaranteed to the find minimizer $\boldsymbol{\beta}^* = \mathrm{argmin}_{\boldsymbol{\beta}}\, L_D(\boldsymbol{\beta})$ using gradient decent if and exactly if $L_D(\boldsymbol{\beta})$ is convex! Is $L_D(\boldsymbol{\beta})$ convex?

- The sum and maximum of convex functions is convex
- $\mathrm{score}_{\boldsymbol{\beta}}(\mathbf{p}_0) = \max_{\mathbf{z}}\ \boldsymbol{\beta}^T \cdot \psi(\mathbf{p}_0, \mathbf{z})$ is convex in $\boldsymbol{\beta}$! Why?
- $\max\left(0, 1 - y^{(i)} \cdot \mathrm{score}_{\boldsymbol{\beta}}(\mathbf{p}_0^{(i)})\right)$ is convex?

Semi Convexity

We guaranteed to the find minimizer $\boldsymbol{\beta}^* = \text{argmin}_{\boldsymbol{\beta}} L_D(\boldsymbol{\beta})$ using gradient decent if and exactly if $L_D(\boldsymbol{\beta})$ is convex! Is $L_D(\boldsymbol{\beta})$ convex?

- The sum and maximum of convex functions is convex
- $\text{score}_{\boldsymbol{\beta}}(\mathbf{p}_0) = \max_{\mathbf{z}} \boldsymbol{\beta}^T \cdot \psi(\mathbf{p}_0, \mathbf{z})$ is convex in $\boldsymbol{\beta}$! Why?
- $\max\left(0, 1 - y^{(i)} \cdot \text{score}_{\boldsymbol{\beta}}(\mathbf{p}_0^{(i)})\right)$ is convex? Yes, iff $y^{(i)} < 0$

Semi Convexity

We guaranteed to the find minimizer $\boldsymbol{\beta}^* = \text{argmin}_{\boldsymbol{\beta}}\, L_D(\boldsymbol{\beta})$ using gradient decent if and exactly if $L_D(\boldsymbol{\beta})$ is convex! Is $L_D(\boldsymbol{\beta})$ convex?

- The sum and maximum of convex functions is convex
- $\text{score}_{\boldsymbol{\beta}}(\mathbf{p}_0) = \max_{\mathbf{z}}\ \boldsymbol{\beta}^T \cdot \psi(\mathbf{p}_0, \mathbf{z})$ is convex in $\boldsymbol{\beta}$! Why?
- $\max\left(0, 1 - y^{(i)} \cdot \text{score}_{\boldsymbol{\beta}}(\mathbf{p}_0^{(i)})\right)$ is convex? Yes, iff $y^{(i)} < 0$
- Thus

$$L_D(\boldsymbol{\beta}) = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{n} \max\left(0, 1 - y^{(i)} \cdot \text{score}_{\boldsymbol{\beta}}(\mathbf{p}_0^{(i)})\right)$$

is convex if all latent values for the positive examples are fixed!

Semi Convexity

We guaranteed to the find minimizer $\boldsymbol{\beta}^* = \operatorname{argmin}_{\boldsymbol{\beta}} L_D(\boldsymbol{\beta})$ using gradient decent if and exactly if $L_D(\boldsymbol{\beta})$ is convex! Is $L_D(\boldsymbol{\beta})$ convex?

- The sum and maximum of convex functions is convex
- $\text{score}_{\boldsymbol{\beta}}(\mathbf{p}_0) = \max_{\mathbf{z}} \boldsymbol{\beta}^T \cdot \psi(\mathbf{p}_0, \mathbf{z})$ is convex in $\boldsymbol{\beta}$! Why?
- $\max\left(0, 1 - y^{(i)} \cdot \text{score}_{\boldsymbol{\beta}}(\mathbf{p}_0^{(i)})\right)$ is convex? Yes, iff $y^{(i)} < 0$
- Thus

$$L_D(\boldsymbol{\beta}) = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{n} \max\left(0, 1 - y^{(i)} \cdot \text{score}_{\boldsymbol{\beta}}(\mathbf{p}_0^{(i)})\right)$$

  is convex if all latent values for the positive examples are fixed!
- This is called “Semi Convexity” in [Felzenszwalb et al., 2010]

Optimization

$$L_D(\boldsymbol{\beta}) = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{n} \max\left(0, 1 - y^{(i)} \cdot \mathrm{score}_{\boldsymbol{\beta}}(\mathbf{p}_0^{(i)})\right)$$

$L_D(\boldsymbol{\beta})$ is convex if we fix $\mathbf{z}$ for all positive examples ($y^{(i)} > 0$)

Optimization

$$L_D(\boldsymbol{\beta}) = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{n}\max\left(0, 1 - y^{(i)}\cdot \text{score}_{\boldsymbol{\beta}}(\mathbf{p}_0^{(i)})\right)$$

$L_D(\boldsymbol{\beta})$ is convex if we fix $\mathbf{z}$ for all positive examples ($y^{(i)} > 0$)

- Alternating Optimization

Optimization

$$L_D(\boldsymbol{\beta}) = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{n}\max\left(0, 1 - y^{(i)} \cdot \text{score}_{\boldsymbol{\beta}}(\mathbf{p}_0^{(i)})\right)$$

$L_D(\boldsymbol{\beta})$ is convex if we fix $\mathbf{z}$ for all positive examples ($y^{(i)} > 0$)

- Alternating Optimization
  - Initialize $\boldsymbol{\beta}$ and iterate:
    1. Pick best $\mathbf{z}$ for each positive example

Optimization

$$L_D(\boldsymbol{\beta}) = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{n} \max\left(0, 1 - y^{(i)} \cdot \mathrm{score}_{\boldsymbol{\beta}}(\mathbf{p}_0^{(i)})\right)$$

$L_D(\boldsymbol{\beta})$ is convex if we fix $\mathbf{z}$ for all positive examples ($y^{(i)} > 0$)

- Alternating Optimization
  - Initialize $\boldsymbol{\beta}$ and iterate:
    1. Pick best $\mathbf{z}$ for each positive example
    2. Add new hard negative examples by running the detector on background images and collecting false detection with high scores

Optimization

$$L_D(\boldsymbol{\beta}) = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{n}\max\left(0, 1 - y^{(i)} \cdot \text{score}_{\boldsymbol{\beta}}(\mathbf{p}_0^{(i)})\right)$$

$L_D(\boldsymbol{\beta})$ is convex if we fix $\mathbf{z}$ for all positive examples ($y^{(i)} > 0$)

- Alternating Optimization
  - Initialize $\boldsymbol{\beta}$ and iterate:
    1. Pick best $\mathbf{z}$ for each positive example
    2. Add new hard negative examples by running the detector on background images and collecting false detection with high scores
    3. Throw away negative examples with low score

Optimization

$$L_D(\boldsymbol{\beta}) = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{n} \max\left(0, 1 - y^{(i)} \cdot \text{score}_{\boldsymbol{\beta}}(\mathbf{p}_0^{(i)})\right)$$

$L_D(\boldsymbol{\beta})$ is convex if we fix $\mathbf{z}$ for all positive examples ($y^{(i)} > 0$)

- Alternating Optimization
  - Initialize $\boldsymbol{\beta}$ and iterate:
    1. Pick best $\mathbf{z}$ for each positive example
    2. Add new hard negative examples by running the detector on background images and collecting false detection with high scores
    3. Throw away negative examples with low score
    4. Optimize $\boldsymbol{\beta}$ via gradient descent

Optimization

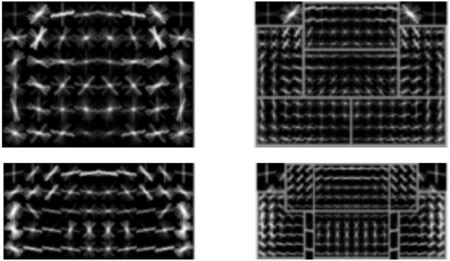
$$L_D(\boldsymbol{\beta}) = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C \sum_{i=1}^{n} \max\left(0, 1 - y^{(i)} \cdot \text{score}_{\boldsymbol{\beta}}(\mathbf{p}_0^{(i)})\right)$$

$L_D(\boldsymbol{\beta})$ is convex if we fix $\mathbf{z}$ for all positive examples ($y^{(i)} > 0$)

- Alternating Optimization
  - Initialize $\boldsymbol{\beta}$ and iterate:
    1. Pick best $\mathbf{z}$ for each positive example
    2. Add new hard negative examples by running the detector on background images and collecting false detection with high scores
    3. Throw away negative examples with low score
    4. Optimize $\boldsymbol{\beta}$ via gradient descent
- The data mining / harvesting step is required as there exists an extremely large number of negatives which can't all be included

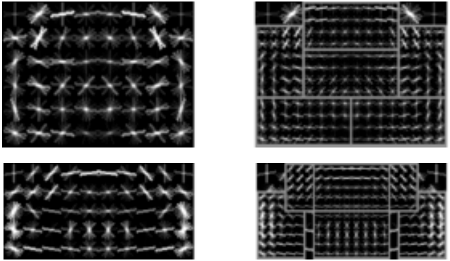
Training Procedure

- For one object category, several models (=components) are trained to deal with significant appearance variations which can't be handled by the deformable part filters (*e.g.*, front vs. side view)



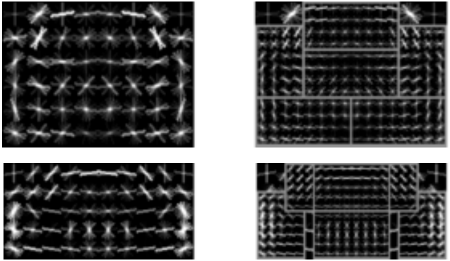
Training Procedure

- For one object category, several models (=components) are trained to deal with significant appearance variations which can't be handled by the deformable part filters (*e.g.*, front vs. side view)
- Coarse-to-fine training:
  1. Train root filters



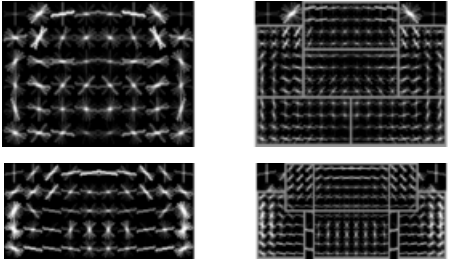
Training Procedure

- For one object category, several models (=components) are trained to deal with significant appearance variations which can't be handled by the deformable part filters (*e.g.*, front vs. side view)
- Coarse-to-fine training:
  1. Train root filters
  2. Initialize parts from root (greedy selection of strong coefficients)



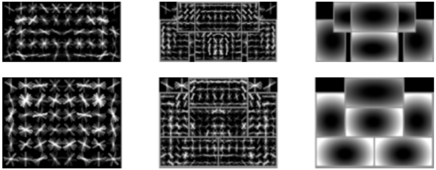
Training Procedure

- For one object category, several models (=components) are trained to deal with significant appearance variations which can't be handled by the deformable part filters (*e.g.*, front vs. side view)
- Coarse-to-fine training:
  1. Train root filters
  2. Initialize parts from root (greedy selection of strong coefficients)
  3. Train final model



Trained DPM Models

Car Model



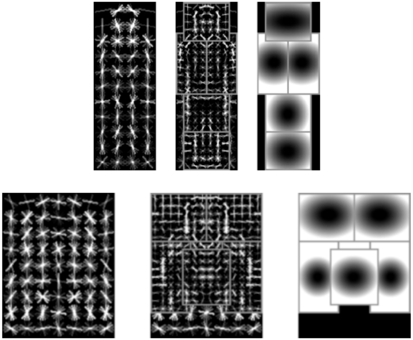
root filters
coarse resolution

part filters
finer resolution

deformation
models

Trained DPM Models

Person Model



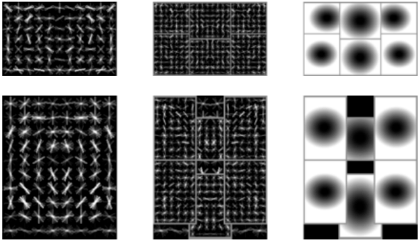
root filters
coarse resolution

part filters
finer resolution

deformation
models

Trained DPM Models

Cat Model



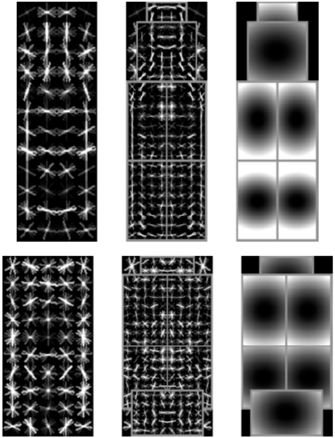
root filters
coarse resolution

part filters
finer resolution

deformation
models

Trained DPM Models

Bottle Model

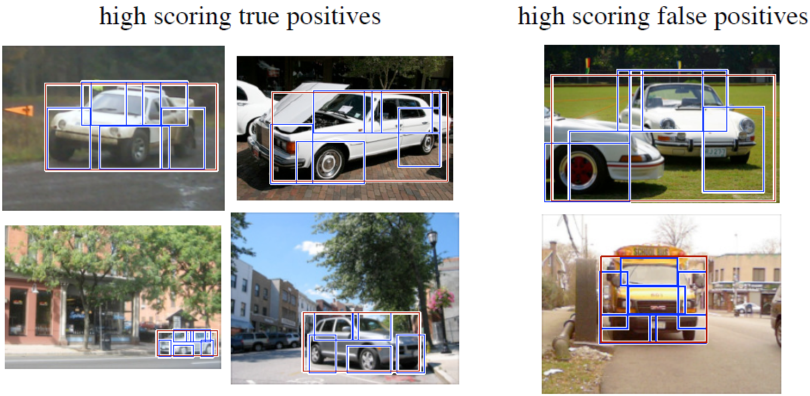


root filters
coarse resolution

part filters
finer resolution

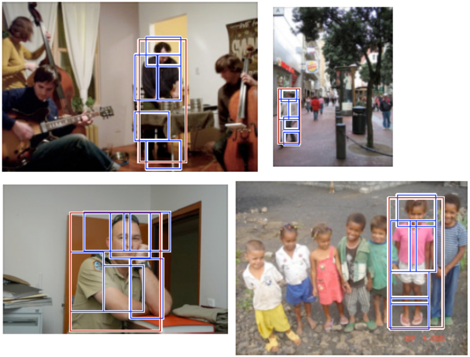
deformation
models

Results on PASCAL VOC



Results on PASCAL VOC

high scoring true positives

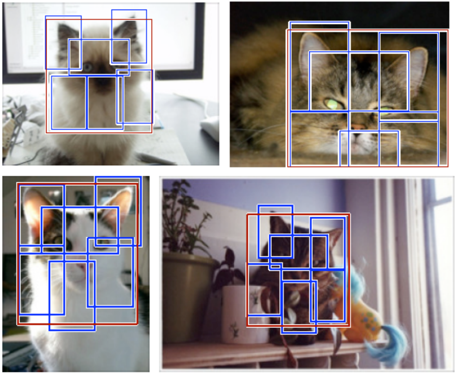


high scoring false positives
(not enough overlap)

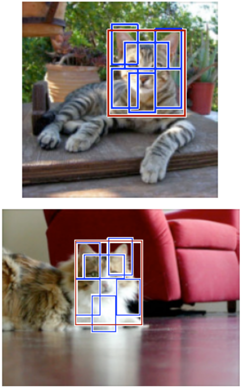
Results on PASCAL VOC

Results on PASCAL VOC

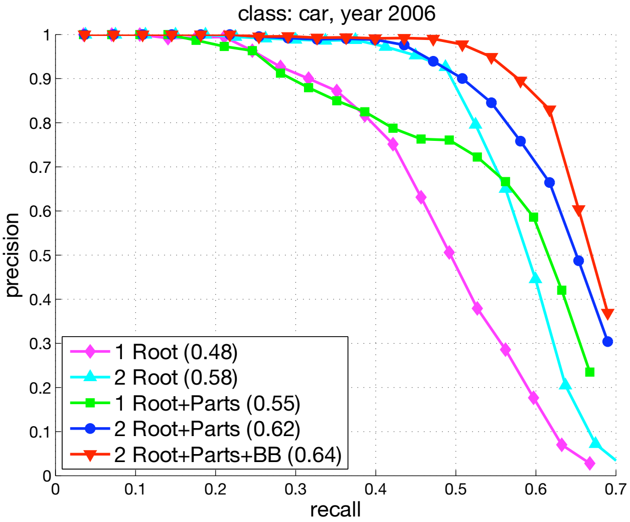
high scoring true positives



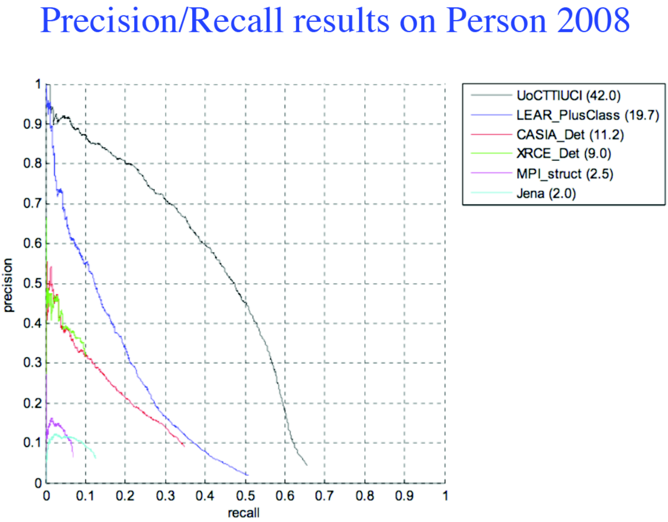
high scoring false positives
(not enough overlap)



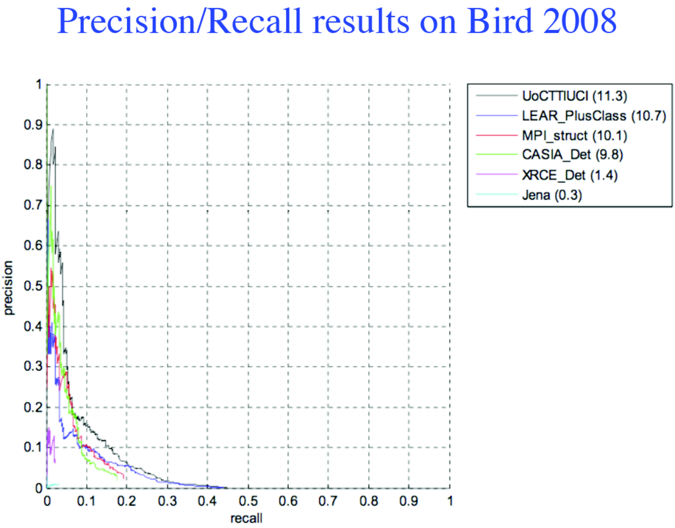
Results on PASCAL VOC



Results on PASCAL VOC



Results on PASCAL VOC



Code and Datasets

Try it yourself!

- MATLAB code available at:
  `http://www.cs.berkeley.edu/~rbg/latent/`
- Training requires about 4 hours for PASCAL
- Detection on one image runs in a few seconds
- Pre-trained models are available

Code and Datasets

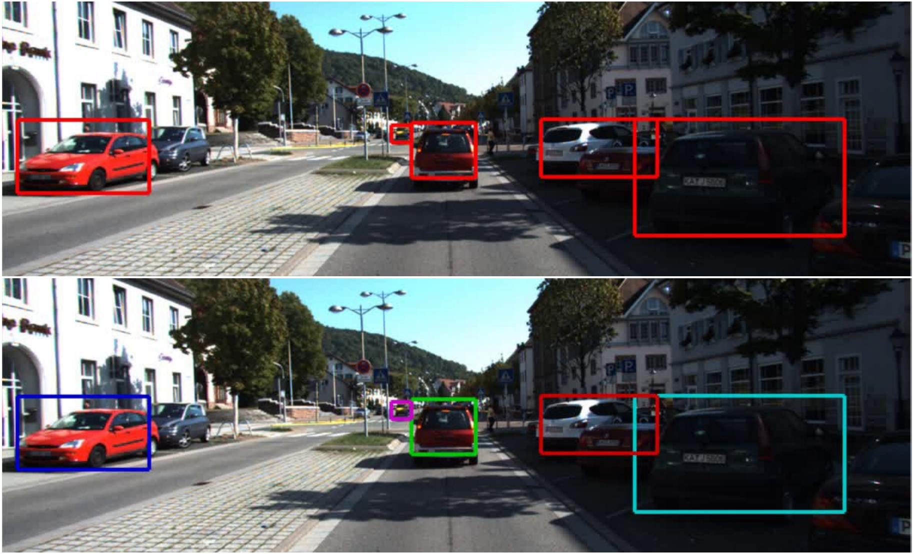
Try it yourself!

- MATLAB code available at:
  `http://www.cs.berkeley.edu/~rbg/latent/`
- Training requires about 4 hours for PASCAL
- Detection on one image runs in a few seconds
- Pre-trained models are available

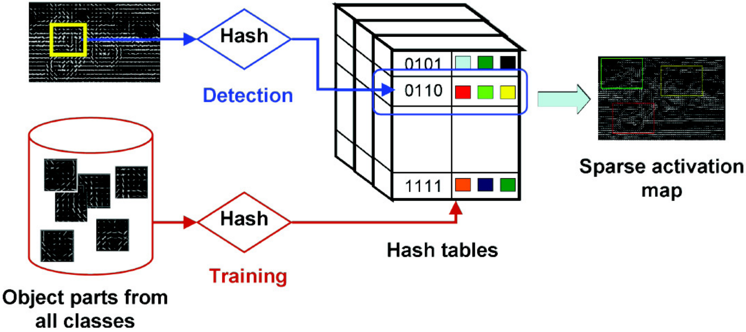
Useful datasets:

- PASCAL VOC:
  `http://pascallin.ecs.soton.ac.uk/challenges/VOC/`
- KITTI:
  `http://www.cvlibs.net/datasets/kitti/eval_object.php`

Results on KITTI



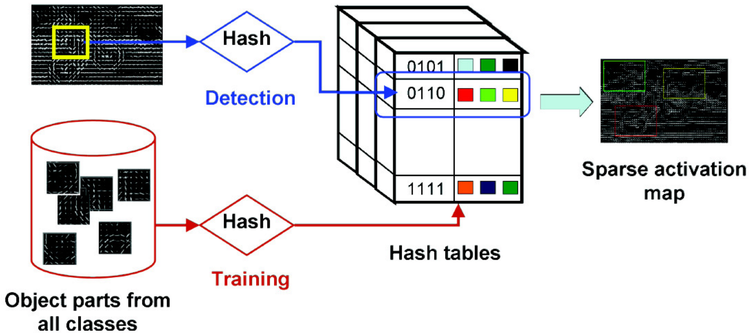
Detecting 100k classes via Hashing [Dean, 2013]



Training:

- Learn part filters using latent SVM
- Store index of each filter in hash table

Detecting 100k classes via Hashing [Dean, 2013]



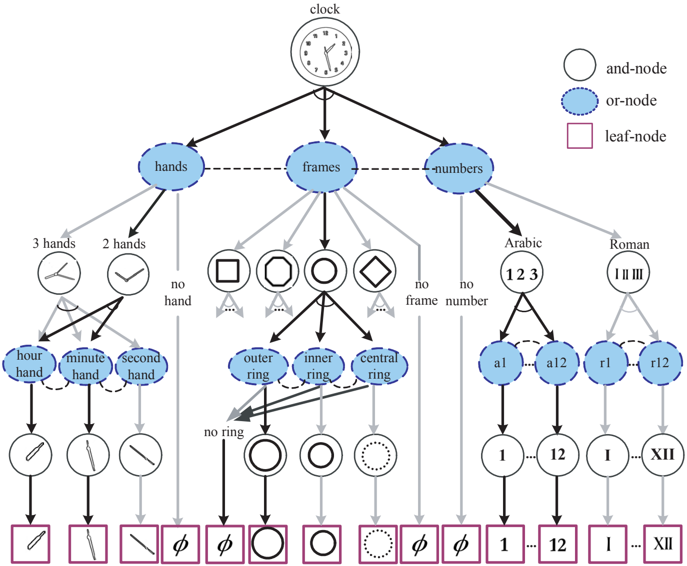
Training:

- Learn part filters using latent SVM
- Store index of each filter in hash table

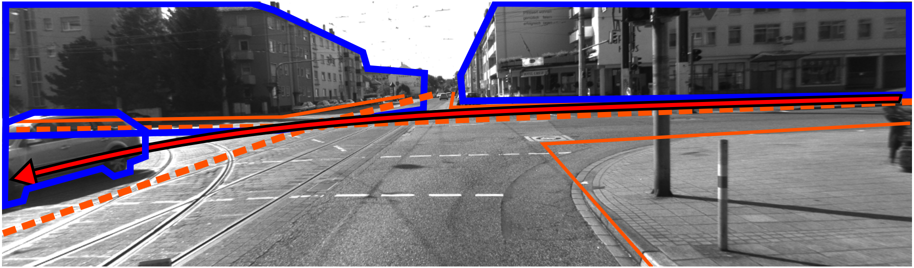
Detection:

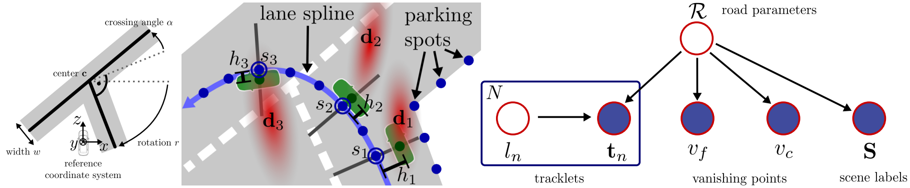
- Lookup hash table and retreive matching filters
- Detect objects using sparse filter scores

Richer Hierarchies: Stochastic Grammars [Zhu & Mumford, 2007]



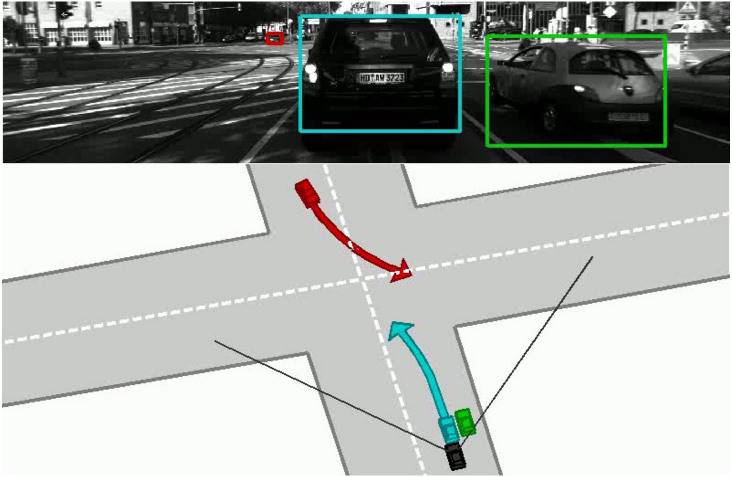
3D Urban Scene Understanding [Geiger at al., 2011-2013]





- Goal: Jointly infer from short videos (moving observer)
  - Topology and geometry of the scene
  - Semantic information (*e.g.*, traffic situation)

3D Urban Scene Understanding [Geiger at al., 2011-2013]



http://www.cvlibs.net/projects/intersection/

http://www.cvlibs.net/software/trackbydet/