Object Detection with Discriminatively Trained Part Based Model

P.F. Felzenszwalb, R.B. Girshick, D. McAllester and D. Ramanan

PAMI 2010

Presented by : Philippe Weinzaepfel

1st February 2013

Object localization

Goal

- Detect and localize objects from generic categories in static images
- Training: bounding boxes around objects

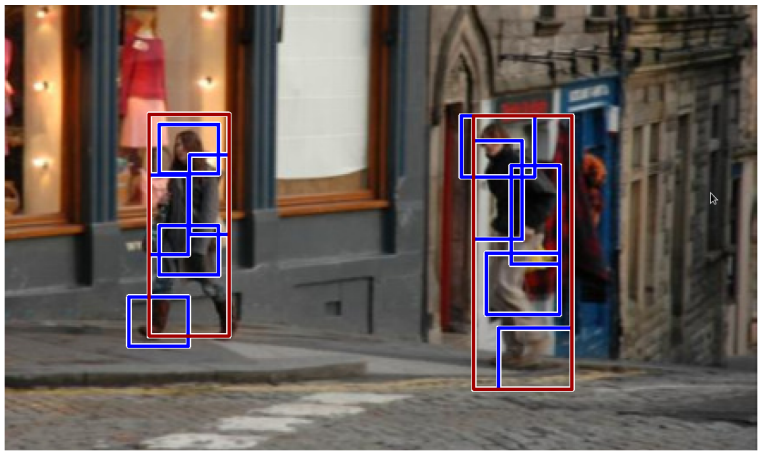
Challenges

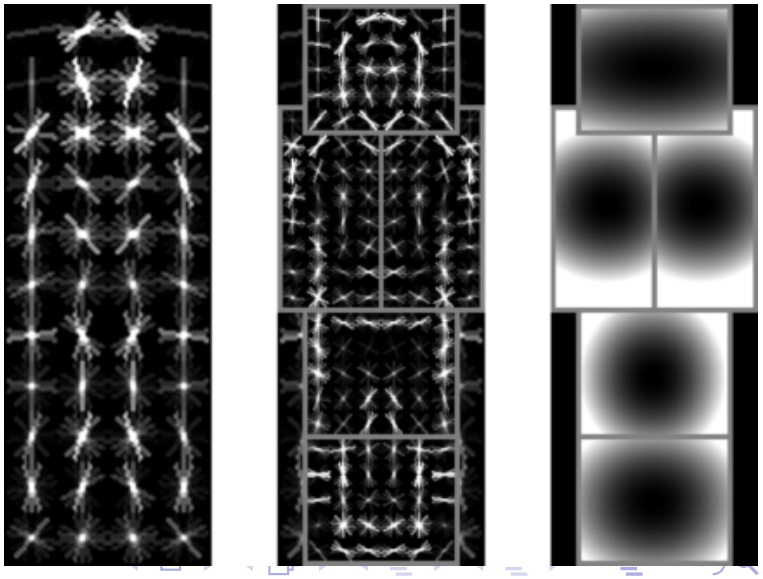
- Illumination changes
- Viewpoint
- Intraclass variability
- Non-rigid deformation

Part-based model

- A collection of parts arranged in a deformable configuration
- Part locations are not known: latent variables
- In this paper: star model (1 root + multiple parts)
- Parts filter at twice resolution of the root filter
- Score of the detection:

$$score(model, \mathbf{x}) = score(root, \mathbf{x}) + \sum_{p \in parts} \max_{\mathbf{y}} [score(p, \mathbf{y}) - cost(p, \mathbf{x}, \mathbf{y})]$$





Part-based model with latent variables

Simple model

Score:

$$f_\beta(x) = \beta.\Phi(x)$$

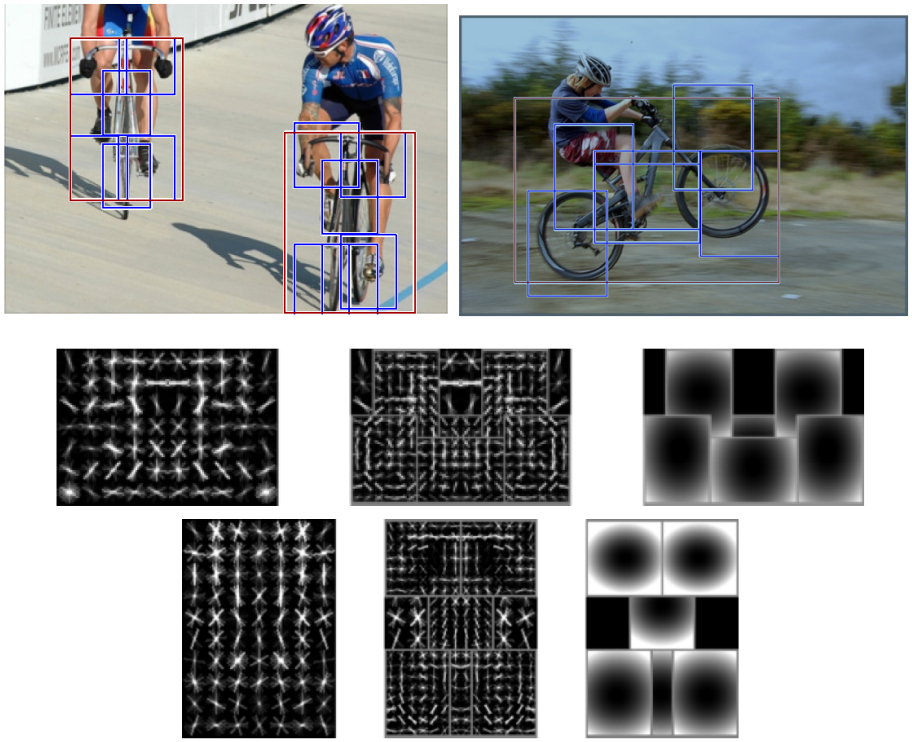
Part-based model

- Score:

$$f_\beta(x) = \max_{x \in Z(x)} \beta.\Phi(x, z)$$

- β: model parameters
- z: specification of object configuration
- Latent SVM to learn β and data-minng for hard negative examples

Mixture of part-based models



- Score: max over components
- Include the component in z

Outline

Outline

Linear filter of features

Each part: linear filter of feature map

- Build a feature pyramid H
- 5 levels per octave at training, 10 at testing
- Weight vector F
- Position $\mathbf{p} = (x, y, l)$
- Score of F at $\mathbf{p}$: $F'.\phi(H, \mathbf{p})$

Outline

Deformable part models

A model

- A root filter F_0
- n parts P_i
  - A filter F_i
  - An anchor v_i
  - A deformation cost d_i
- A bias term b (to compare models in mixtures)

An object hypothesis

- Position of root and parts $z = (\mathbf{p}_0, ..., \mathbf{p}_n)$
- $\mathbf{p}_i = (x_i, y_i, l_i)$
- Hypothesis: parts at twice the resolution of root

Deformable part models

Score of an hypothesis

- $score(\mathbf{p}_0, ..., \mathbf{p}_n) = \sum_{i=0}^{n} F_i' . \phi(H, \mathbf{p}_i) - \sum_{i=1}^{n} d_i . \phi_d(dx_i, dy_i) + b$
- $(dx_i, dy_i) = (x_i, y_i) - (2(x_0, y_0) + v_i)$
- $\phi_d(dx, dy) = (dx, dy, dx^2, dy^2)$

Deformable part models

Score of an hypothesis

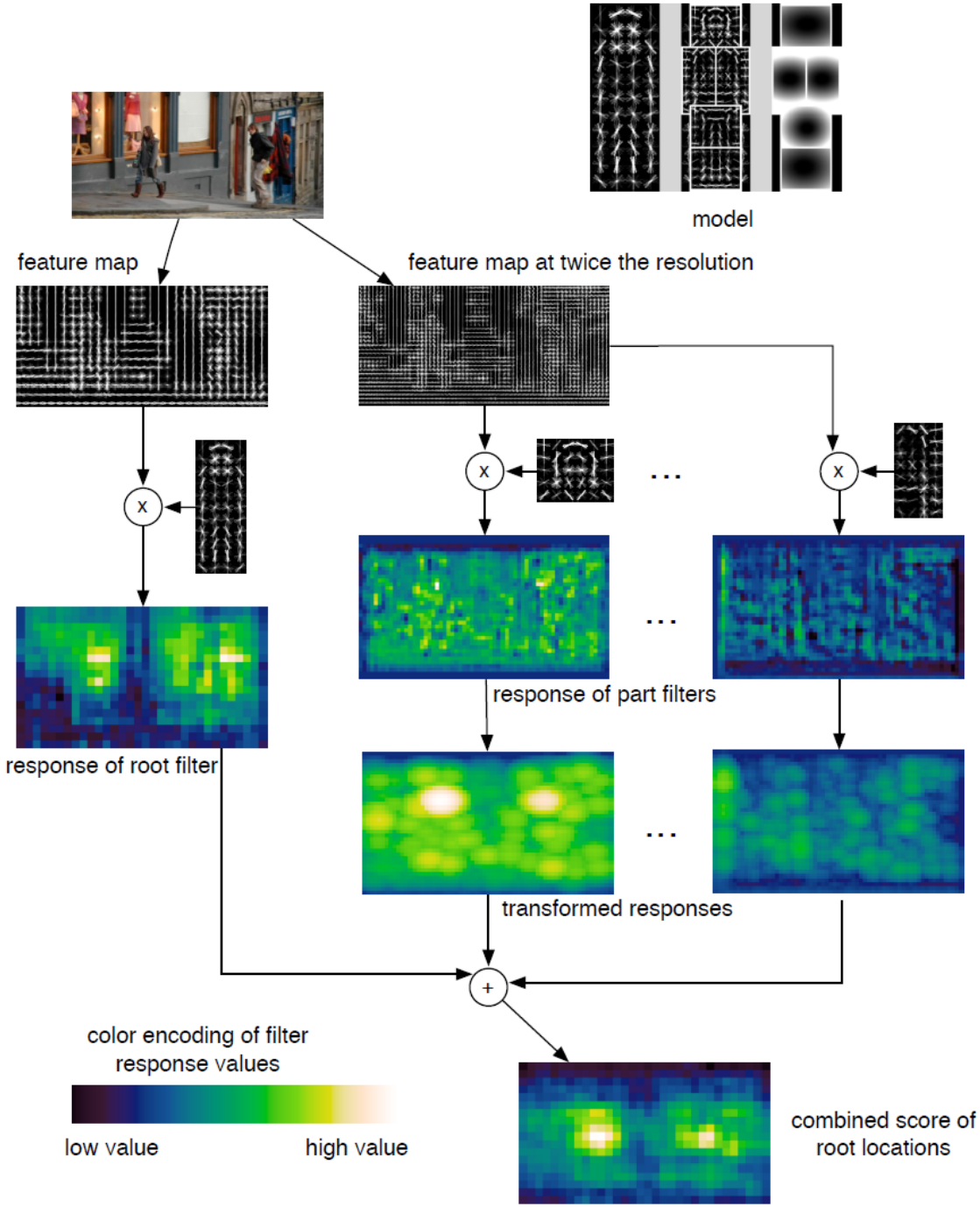
- $score(\mathbf{p}_0, ..., \mathbf{p}_n) = \sum_{i=0}^{n} F_i' . \phi(H, \mathbf{p}_i) - \sum_{i=1}^{n} d_i . \phi_d(dx_i, dy_i) + b$
- $(dx_i, dy_i) = (x_i, y_i) - (2(x_0, y_0) + v_i)$
- $\phi_d(dx, dy) = (dx, dy, dx^2, dy^2)$

Link to Latent SVM

The score can be written as $\beta . \Psi(H, z)$ with:

- $\beta = (F_0', ..., F_n', d_1, ..., d_n, b)$
- $\Psi(H, z) = (\phi(H, \mathbf{p}_0), ..., \phi(H, \mathbf{p}_n), -\phi_d(dx_1, dy_1), ..., -\phi_d(dx_n, dy_n), 1)$

Outline


model
feature map
feature map at twice the resolution
response of part filters
response of root filter
transformed responses
color encoding of filter response values
low value
high value
combined score of root locations

Outline

Mixture of models

Mixture M of m models

- $M = (M_1, ..., M_m)$
- $z = (c, \mathbf{p}_0, ..., \mathbf{p}_{n_c}) = (c, z')$
- $score(z) = \beta_c.\Psi(H, z')$

Score as $\beta.\Psi(H, z)$

- $\beta = (\beta_1, ..., \beta_m)$
- $\Psi(H, z) = (0, ..., 0, \Psi(H, z'), 0, ..., 0)$

Outline

Problem

- Classifier that score an example x with:

$$f_\beta(x) = \max_{z \in Z(x)} \beta.\Phi(x, z)$$

- $Z(x)$: set of possible latent values for x
- As for SVM, we learn β by minimizing

$$L_D(\beta) = \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n} \max\Big(0, 1 - y_i f_\beta(x_i)\Big)$$

with $D = \Big(\langle x_1, y_1\rangle, ..., \langle x_n, y_n\rangle\Big)$ a set of labeled examples

Outline

Semi-convexity

$$L_D(\beta) = \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n} \max\left(0, 1 - y_i f_\beta(x_i)\right)$$

A latent SVM is semi-convex

- The loss function is convex in β for negative examples.

Semi-convexity

$$L_D(\beta) = \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n} \max\left(0, 1 - y_i f_\beta(x_i)\right)$$

A latent SVM is semi-convex

- The loss function is convex in β for negative examples.

SVM

Convexity:

- f_β linear in β
- Hinge loss is convex as maximum of two convex functions

Semi-convexity

$$L_D(\beta) = \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n} \max\left(0, 1 - y_i f_\beta(x_i)\right)$$

A latent SVM is semi-convex

- The loss function is convex in β for negative examples.

SVM

Convexity:

- f_β linear in β
- Hinge loss is convex as maximum of two convex functions

LSVM

- f_β is convex as maximum of convex functions
- For negative examples, the loss function is convex

Semi-convexity

$$L_D(\beta) = \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n} \max\left(0, 1 - y_i f_\beta(x_i)\right)$$

A latent SVM is semi-convex

- The loss function is convex in β for negative examples.
- $L_D(\beta)$ is convex when latent variables are specified for positive examples.

SVM

Convexity:

- f_β linear in β
- Hinge loss is convex as maximum of two convex functions

LSVM

- f_β is convex as maximum of convex functions
- For negative examples, the loss function is convex
- If there is a single possible latent value for each positive example: f_β is linear

Outline

Optimization

- Z_p specifying a latent value for each positive example
- $D(Z_p)$ derived from D by restricting latent values for positive examples

Optimization

- Z_p specifying a latent value for each positive example
- $D(Z_p)$ derived from D by restricting latent values for positive examples
- $L_D(\beta) = \min_{Z_p} L_D(\beta, Z_p) = \min_{Z_p} L_{D(Z_p)}(\beta)$
- $L_D(\beta) \leqslant L_D(\beta, Z_p)$

Optimization

- Z_p specifying a latent value for each positive example
- $D(Z_p)$ derived from D by restricting latent values for positive examples
- $L_D(\beta) = \min_{Z_p} L_D(\beta, Z_p) = \min_{Z_p} L_{D(Z_p)}(\beta)$
- $L_D(\beta) \leqslant L_D(\beta, Z_p)$

Algorithm

- **Relabel positive examples:** Optimize $L_D(\beta, Z_p)$ over Z_p, i.e. select

$$z_i = \underset{z \in Z(x_i)}{\operatorname{argmax}}\, \beta.\Phi(x_i, z)$$

.

- **Optimize β:** Stochastic gradient descent

Stochastic gradient descent

Gradient descent

$$L_D(\beta) = \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n} \max\left(0, 1 - y_i f_\beta(x_i)\right)$$

$$\nabla L_D(\beta) = \beta + C\sum_{i=1}^{n} h(\beta, x_i, y_i) \qquad \text{(subgradient)}$$

$$h(\beta, x_i, y_i) = \begin{cases} 0 \text{ if } y_i f_\beta(x_i) \geqslant 1 \\ -y_i \phi(x_i, z_i(\beta)) \text{ otherwise} \end{cases}$$

Issue

Too costly to go over data to compute the exact gradient.

Stochastic gradient descent

Approximation of the gradient

Approximate $\sum_{i=1}^{n} h(\beta, x_i, y_i)$ by $nh(\beta, x_i, y_i)$ for one example $\langle x_i, y_i \rangle$

Algorithm

- Let α_t be the learning rate for iteration t
- Let i be a random example
- Let $z_i = \text{argmax}_{z \in Z(x_i)} \beta.\phi(x_i, z)$
- Update $\beta = \beta - \alpha_t \nabla_i L_D(\beta)$

Convergence

- Learning rate $\alpha_t = \frac{1}{t}$
- Depends on the number of training examples

Outline

Recall: Data-mining hard examples in SVM

Many negative instances

Bootstrapping

Collect hard negatives as incorrectly classified examples from a previous model.

Recall: Data-mining hard examples in SVM

Many negative instances

Bootstrapping

Collect hard negatives as incorrectly classified examples from a previous model.

Hard and easy examples

- $H(\beta, D) = \{\langle x, y\rangle \in D \mid y f_\beta(x) < 1\}$
- $E(\beta, D) = \{\langle x, y\rangle \in D \mid y f_\beta(x) > 1\}$

$$\beta^*(D) = \operatorname*{argmin}_\beta L_D(\beta) \qquad \text{(unique)}$$

Goal

Find a subset $C \subseteq D$ such that $\beta^*(C) = \beta^*(D)$.

Recall: Data-mining hard examples in SVM

Algorithm

- $\beta_t = \beta^*(C_t)$ (model training)
- If $H(\beta_t, D) \subseteq C_t$, stop and return β_t
- Remove easy examples
- Add hard examples

Recall: Data-mining hard examples in SVM

Algorithm

- $\beta_t = \beta^*(C_t)$ (model training)
- If $H(\beta_t, D) \subseteq C_t$, stop and return β_t
- Remove easy examples
- Add hard examples

Proof

- Let $C \subseteq D$. If $H(\beta, D) \subseteq C$, $\beta^*(C) = \beta^*(D)$.
- The algorithm terminates.

Data-mining hard examples in LSVM

- Feature cache F: set of (i, v) with $1 \leqslant i \leqslant n$ and $v = \phi(x_i, z)$ with $z \in Z(x_i)$ (one for positive examples)
- $I(F)$ examples indexed by F
- $L_F(\beta) = \frac{1}{2}\|\beta\|^2 + C \sum_{i \in I(F)} \max(0, 1 - y_i(\max_{(i,v) \in F} \beta.v))$

Data-mining hard examples in LSVM

- Feature cache F: set of (i, v) with $1 \leqslant i \leqslant n$ and $v = \phi(x_i, z)$ with $z \in Z(x_i)$ (one for positive examples)
- $I(F)$ examples indexed by F
- $L_F(\beta) = \frac{1}{2}\|\beta\|^2 + C \sum_{i \in I(F)} \max(0, 1 - y_i(\max_{(i,v) \in F} \beta.v))$

Modified stochastic gradient descent

- Let α_t be the learning rate for iteration t
- Let $i \in I(F)$ be a random example indexed by F
- Let $v_i = \text{argmax}_{v \in V(i)} \beta.v$
- Update $\beta = \beta - \alpha_t \nabla_i L_D(\beta)$

Data-mining hard examples in LSVM

- Feature cache F: set of (i, v) with $1 \leqslant i \leqslant n$ and $v = \phi(x_i, z)$ with $z \in Z(x_i)$ (one for positive examples)
- $I(F)$ examples indexed by F
- $L_F(\beta) = \frac{1}{2}\|\beta\|^2 + C \sum_{i \in I(F)} \max(0, 1 - y_i(\max_{(i,v) \in F} \beta.v))$

Modified stochastic gradient descent

- Let α_t be the learning rate for iteration t
- Let $i \in I(F)$ be a random example indexed by F
- Let $v_i = \text{argmax}_{v \in V(i)} \beta.v$
- Update $\beta = \beta - \alpha_t \nabla_i L_D(\beta)$

- We would like to find a small F such that $\beta^*(F) = \beta^*(D(Z_p))$
- $H(\beta, D) = \{(i, \Phi(x_i, z_i)) \mid z_i = \text{argmax}_{z \in Z(x_i)} \beta.\Phi(x_i, z) \text{ and } y_i(\beta.\Phi(x_i, z_i)) < 1\}$
- $E(\beta, D) = \{(i, v) \in F | y_i(\beta.v) > 1\}$
- Same procedure

Outline

Data:
Positive examples $P = \{(I_1, B_1), \ldots, (I_n, B_n)\}$
Negative images $N = \{J_1, \ldots, J_m\}$
Initial model β

Result: New model β

```
1  F_n := ∅
2  for relabel := 1 to num-relabel do
3      F_p := ∅
4      for i := 1 to n do
5          Add detect-best(β,I_i,B_i) to F_p
6      end
7      for datamine := 1 to num-datamine do
8          for j := 1 to m do
9              if |F_n| ≥ memory-limit then break
10             Add detect-all(β,J_j,−(1+δ)) to F_n
11         end
12         β :=gradient-descent(F_p ∪ F_n)
13         Remove (i,v) with β·v < −(1+δ) from F_n
14     end
15 end
```

Outline

Initialization

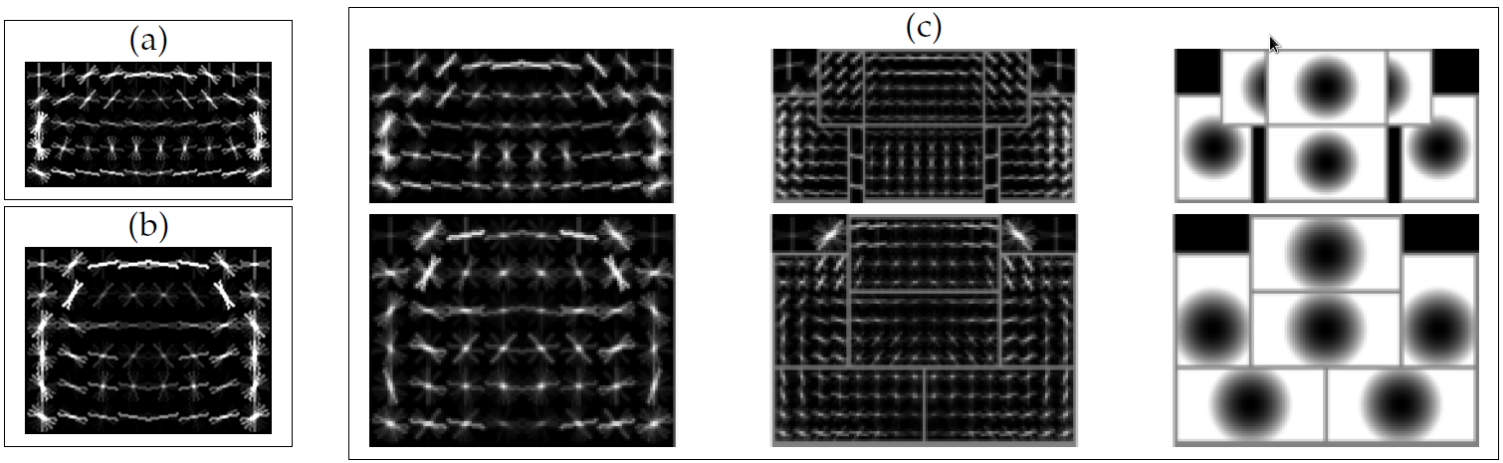
- *Phase 1: Initializing root filters*
  - Split positive examples according to aspect ratio in m groups
  - Decide area of F_i according to aspect ratio and area
  - Train the root filter F_i with classical SVM

Initialization

- *Phase 1: Initializing root filters*
  - Split positive examples according to aspect ratio in m groups
  - Decide area of F_i according to aspect ratio and area
  - Train the root filter F_i with classical SVM
- *Phase 2: Merging components*
  - Train mixture of models with no part (latent: component and $\mathbf{p}_0$)

Initialization

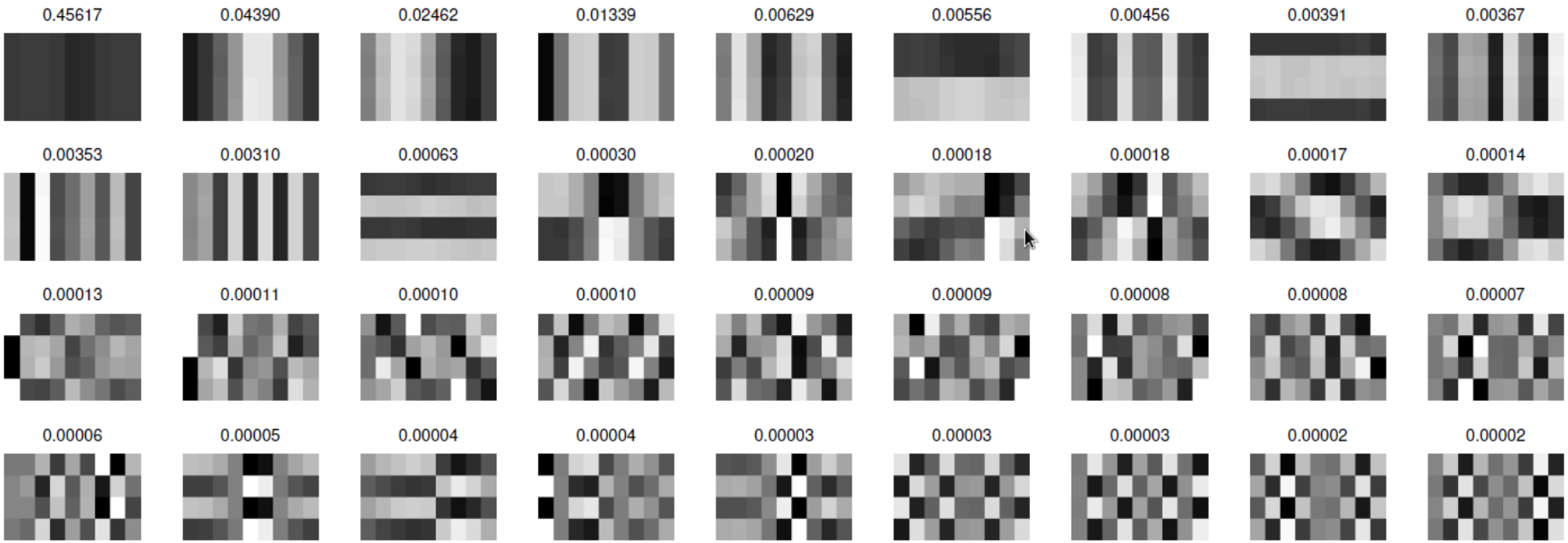
- *Phase 1: Initializing root filters*
  - Split positive examples according to aspect ratio in m groups
  - Decide area of F_i according to aspect ratio and area
  - Train the root filter F_i with classical SVM
- *Phase 2: Merging components*
  - Train mixture of models with no part (latent: component and $\mathbf{p}_0$)
- *Phase 3: Initializing part filters*
  - Initialize 6 rectangles to cover high energy of the root filter
  - Anchor at center or symmetric according to vertical axis
  - $d_i = (0, 0, 1, 1)$



Outline

Features

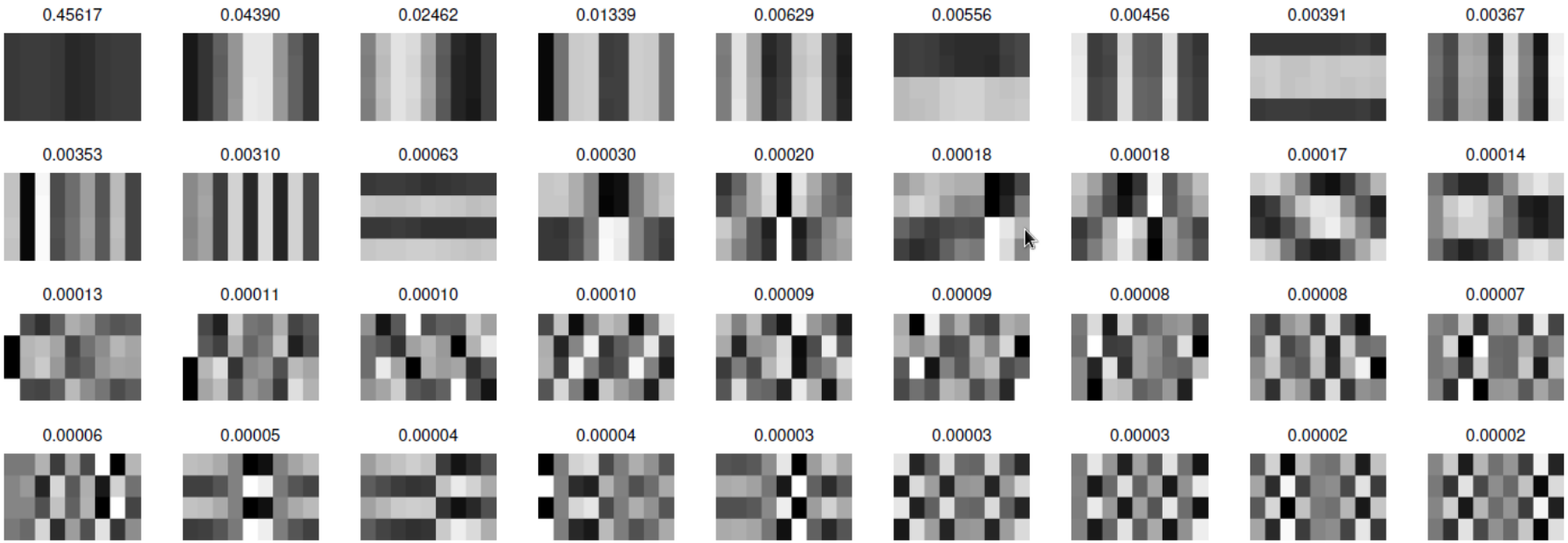
36-dimensional HOG descriptors (9 orientations, 4 normalizations, non-contrast sensitive)



11 main eigenvectors
Projection is costly

Features

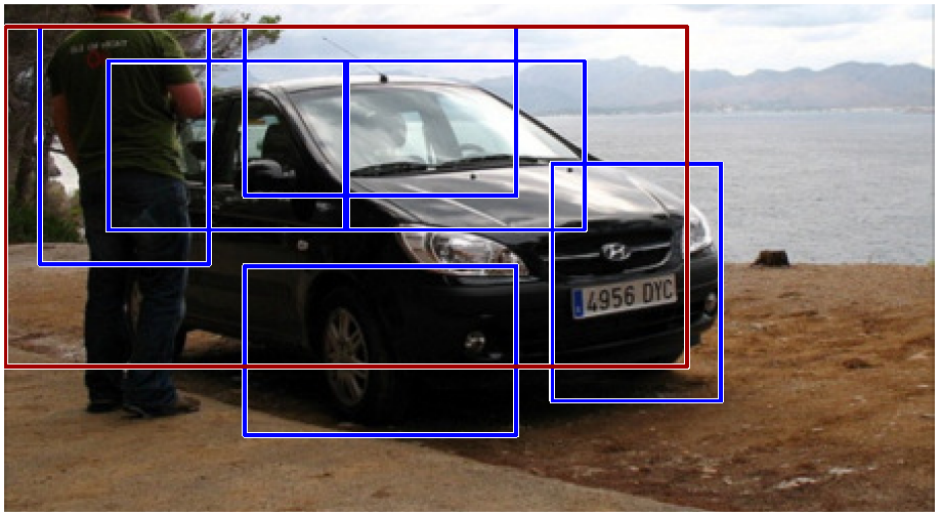
36-dimensional HOG descriptors (9 orientations, 4 normalizations, non-contrast sensitive)



11 main eigenvectors
Projection is costly
Similar results when using 9+4 basis vectors
(18+9)+4=31 for contrast and non-contrast sensitive

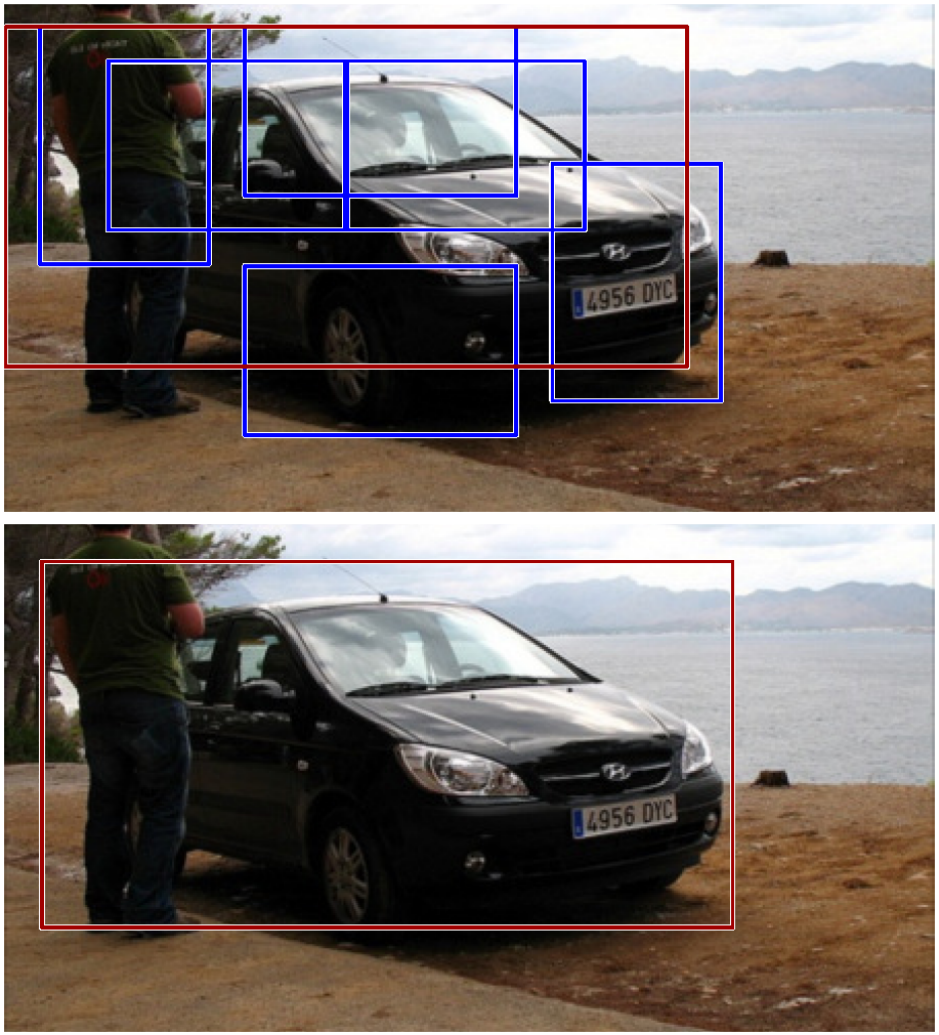
Outline

Bounding box prediction



- Input z: position of each part + root width
- Outputs $(x1, y1, x2, y2)$: position of the prediction bounding box

Bounding box prediction



- Input z: position of each part + root width
- Outputs $(x1, y1, x2, y2)$: position of the prediction bounding box
- How: for each component of a mixture, 4 linear functions learned by linear least-square regression on training data

Outline

Non-Maxima Suppression

Issue

For one object, there are a lot of overlapping detections

Solution

- Sort detections by score
- Add the detection one by one and skip if there exists a detection with overlap of at least 50%

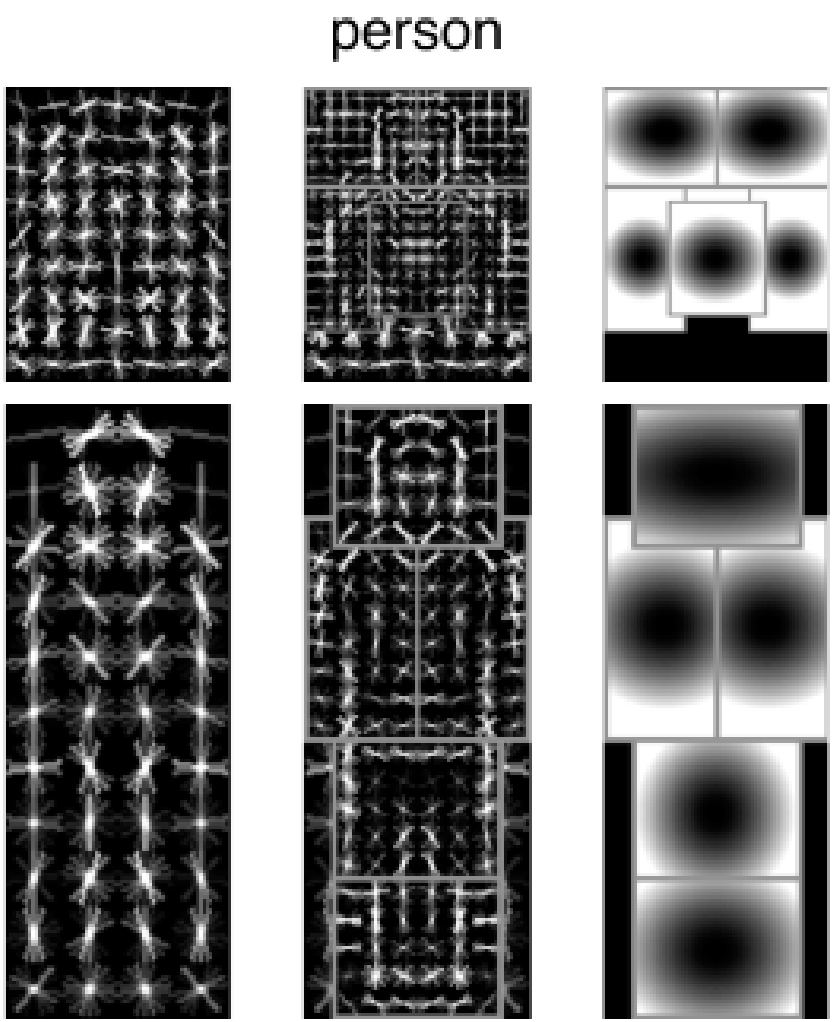
Outline

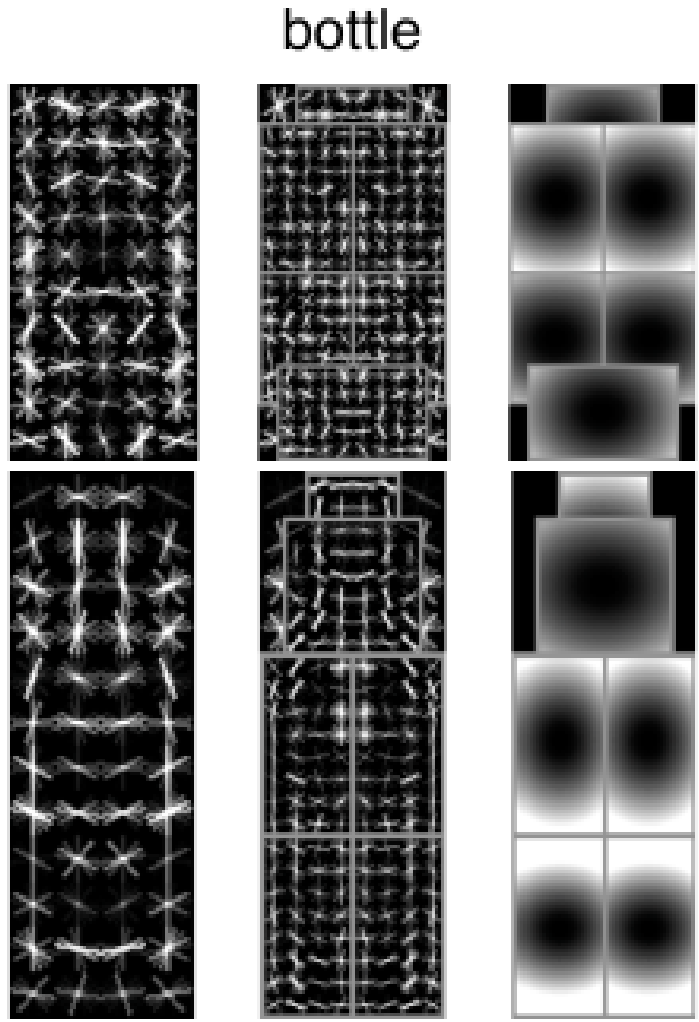
Contextual information

- Rescore the detection score
- Best score for k other categories: $c(I) = (s_1, s_2, ..., s_k)$
- For a detection (B, s), classify $g = (\sigma(s), \frac{x_1}{w}, \frac{y_1}{h}, \frac{x_2}{w}, \frac{y_2}{h}, c(I))$
- Learned from training data

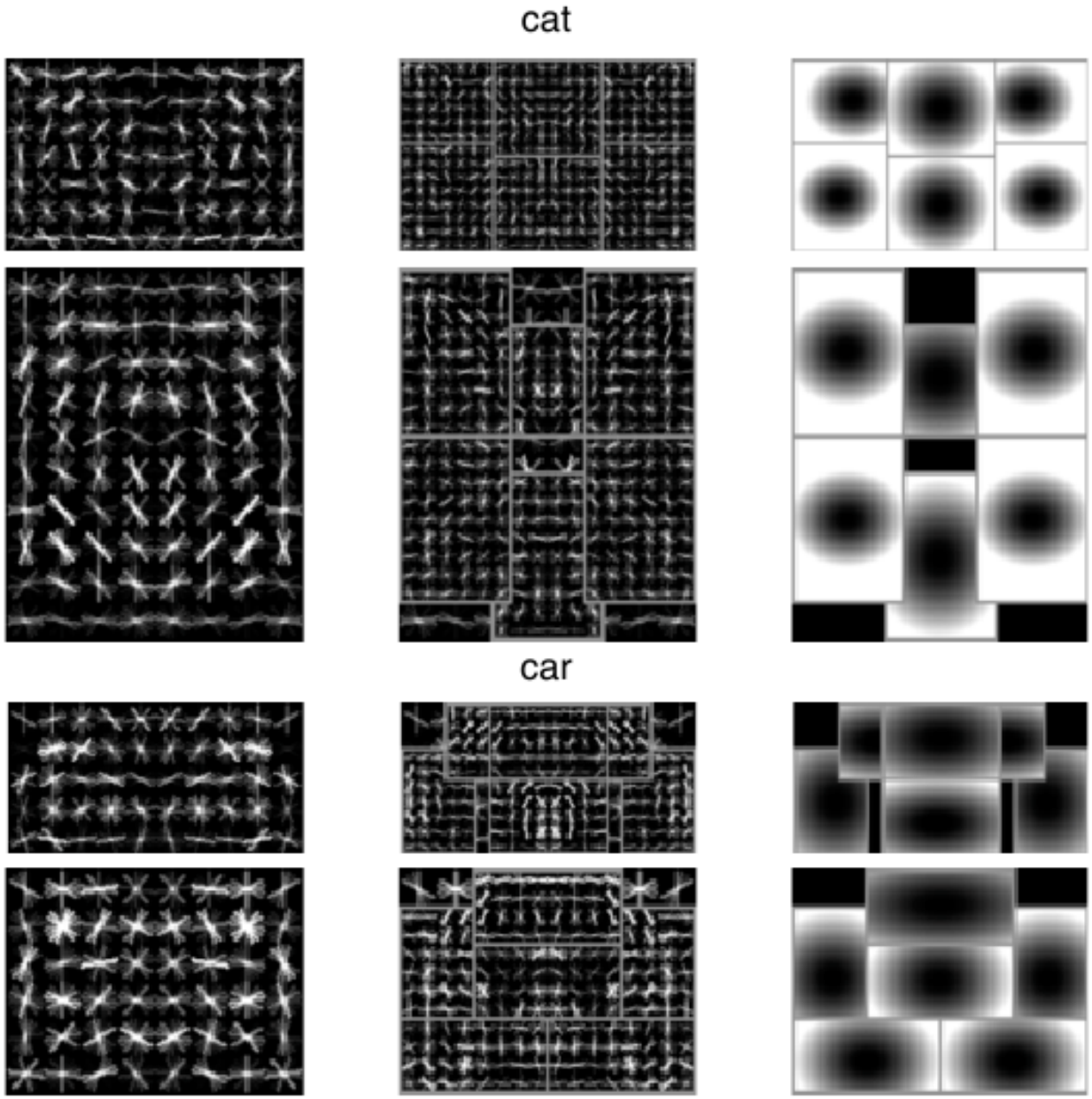
Outline

Models





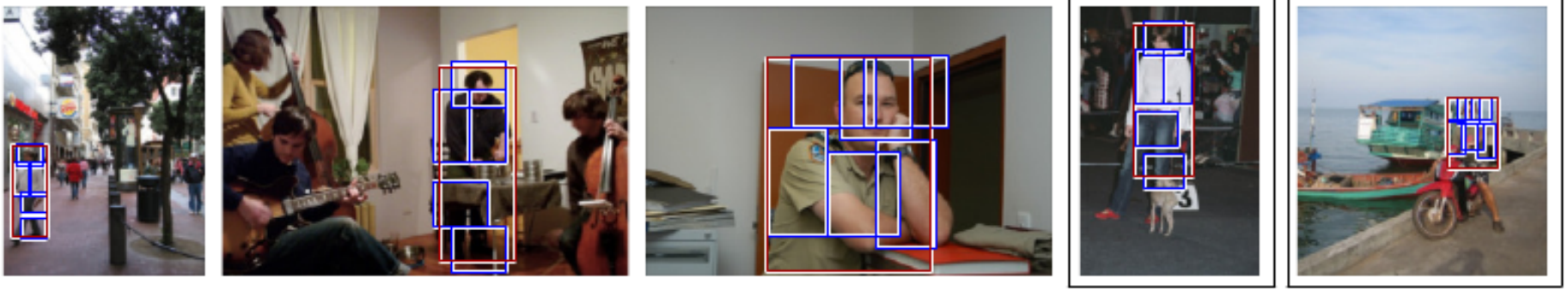
Models



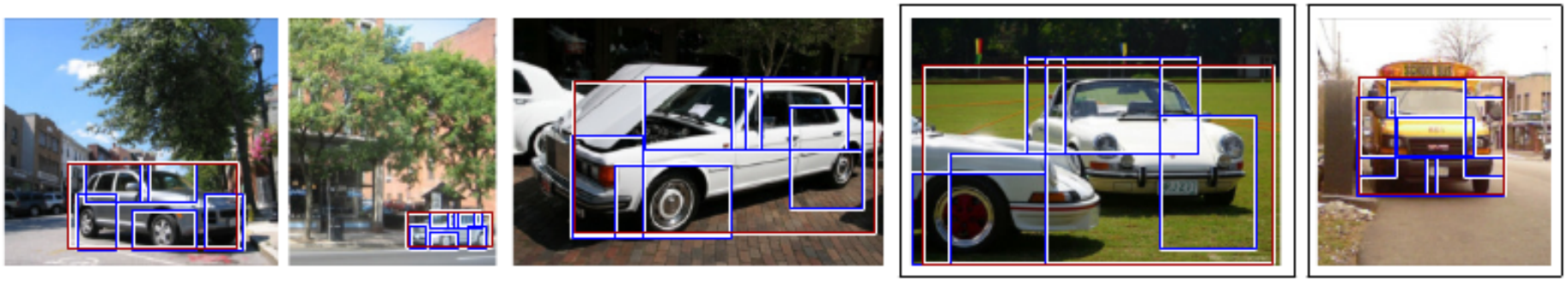
Outline

Detections

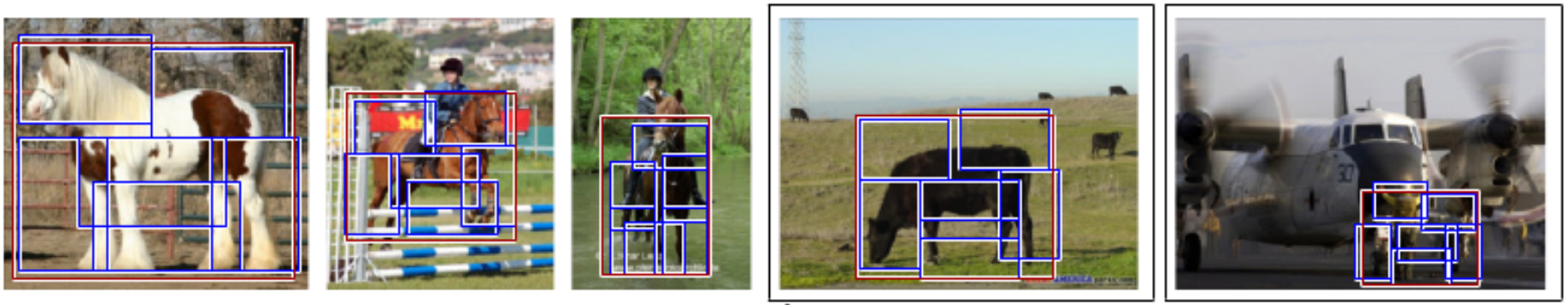
person



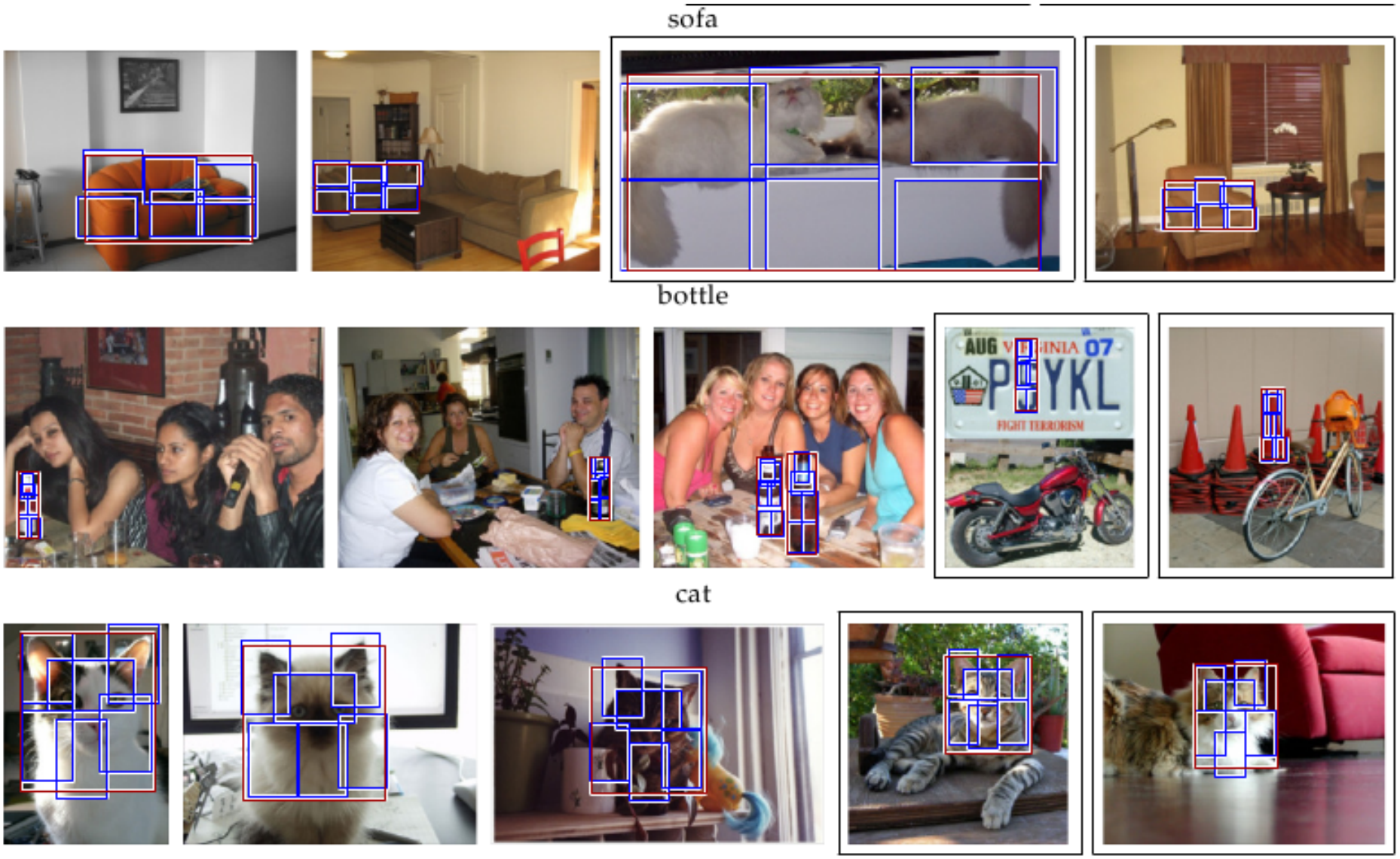
car



horse



Detections



Conclusion

Recent extensions

- Speeding-up detection using cascade classifier
- Grammar models

Conclusion

Recent extensions

- Speeding-up detection using cascade classifier
- Grammar models

Thanks for your attention.

Any question ?