Learning to Localize Objects with Structured Output Regression

Matthew Blaschko and Christopher Lampert

ECCV 2008 – Best Student Paper Award

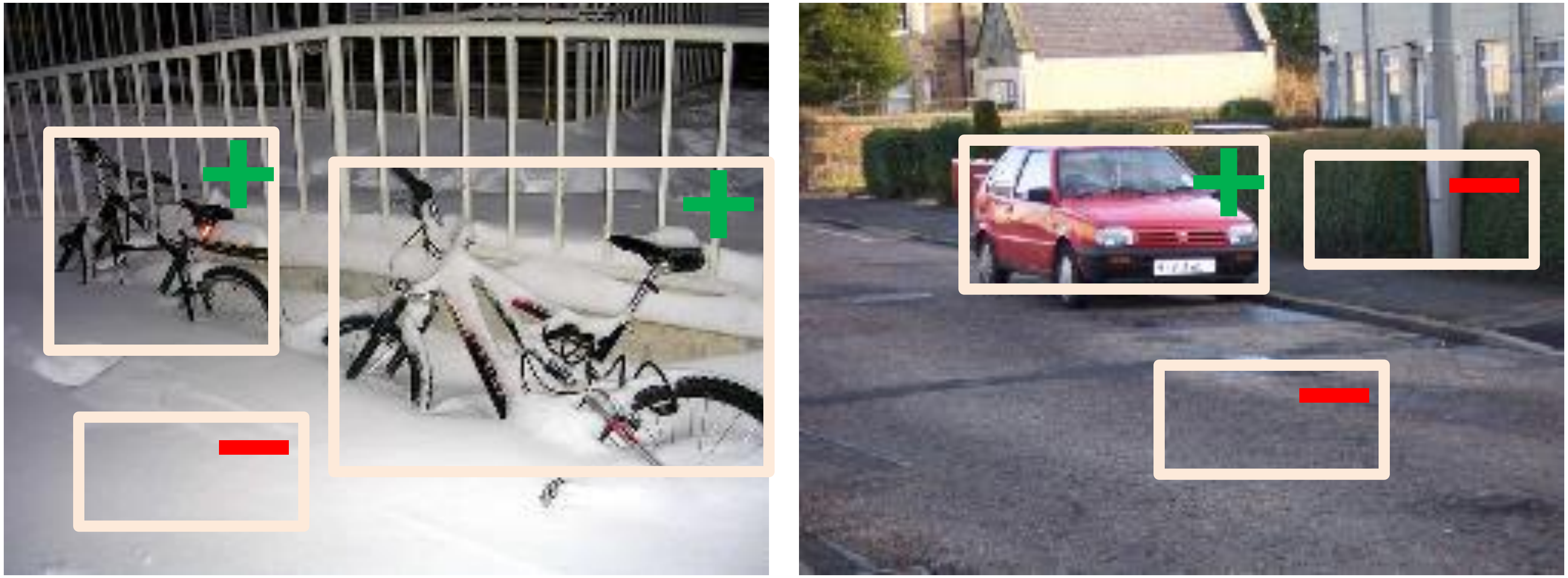
Presentation by Jaeyong Sung and Yiting Xie

Object Localization

- important task for image understanding

Object Localization

- important task for image understanding
- standard approach: binary training + sliding window

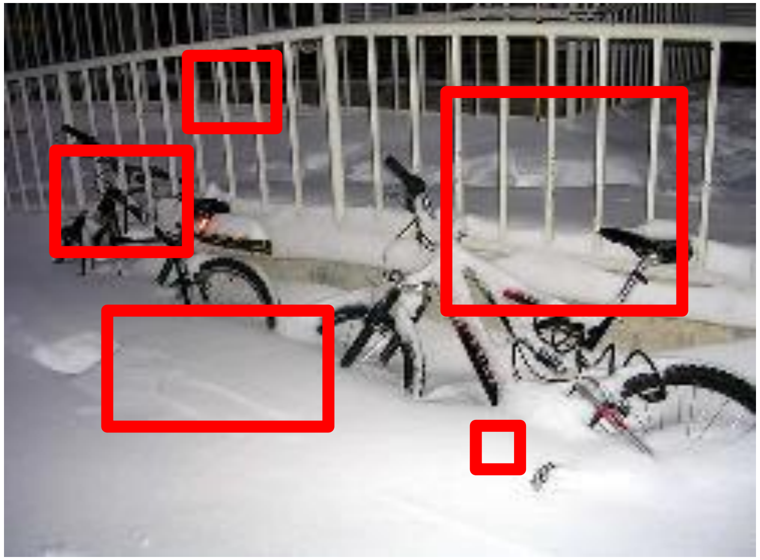


Object Localization

- important task for image understanding
- standard approach: binary training + sliding window

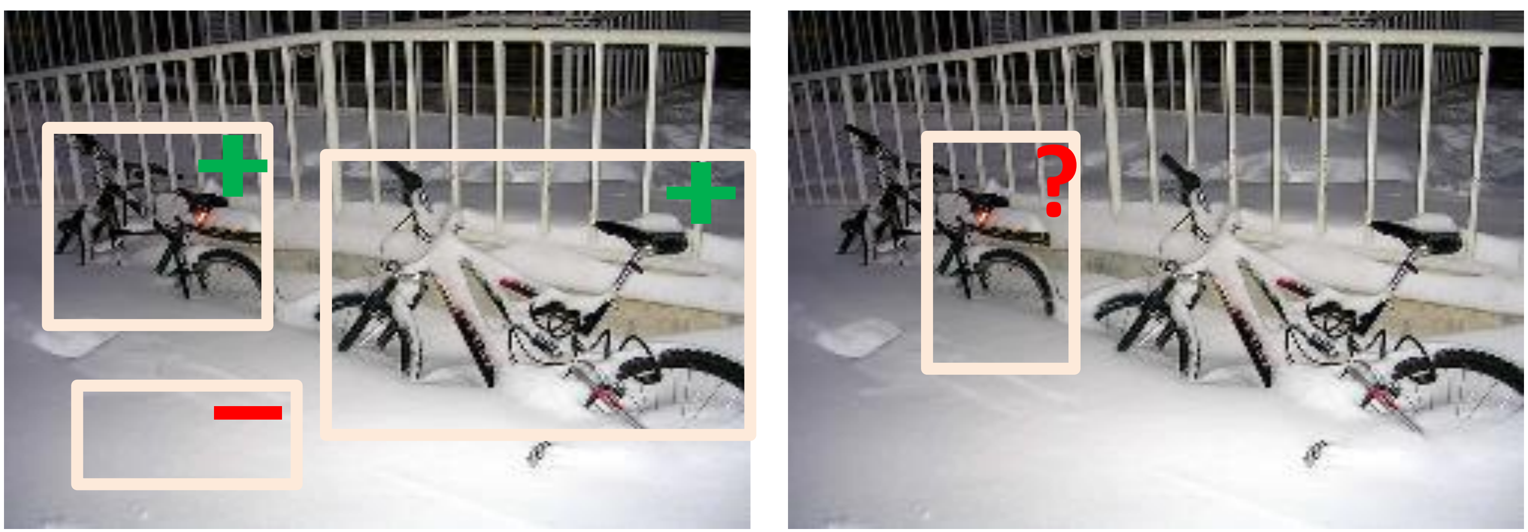
Object Localization

- Main disadvantages of sliding window
  - Inefficient to scan over the entire image
    - 320 x 240 image → one billion rectangular sub-images

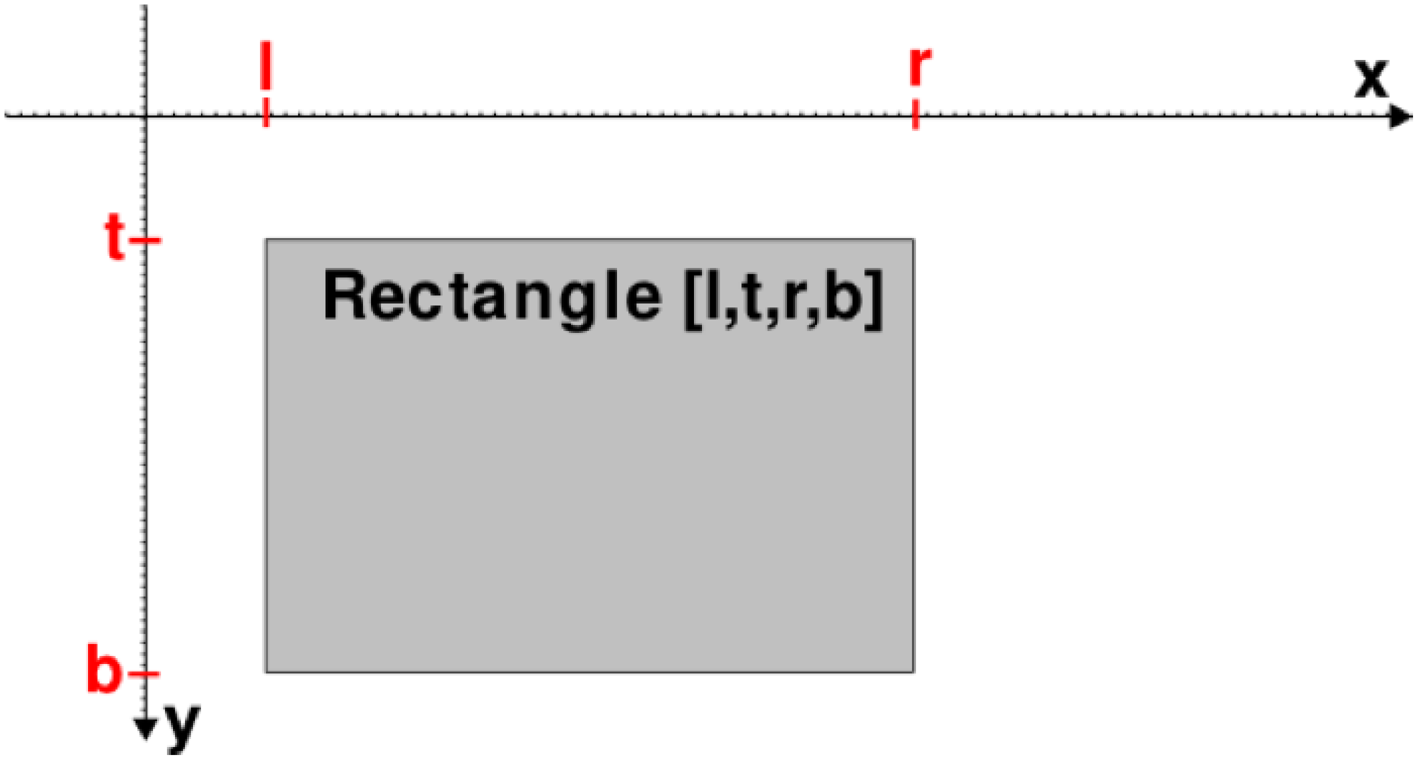


Object Localization

- Main disadvantages of sliding window
  - Inefficient to scan over the entire image
    - 320 x 240 image → one billion rectangular sub-images
  - Not clear how to optimally train a discriminant function
    - main contribution of this paper
    - utilizes structured learning



Parameterization of Bounding Box



$$Y \equiv \{(\omega, t, l, b, r) | \omega \in \{+1, -1\}, (t, l, b, r) \in \mathbb{R}^4\}$$

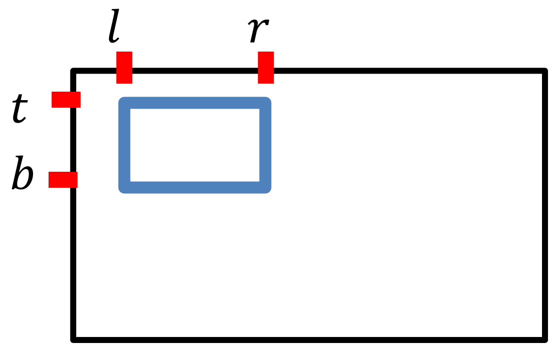
- If $\omega = -1$, the coordinate vector is ignored.

Structured Regression

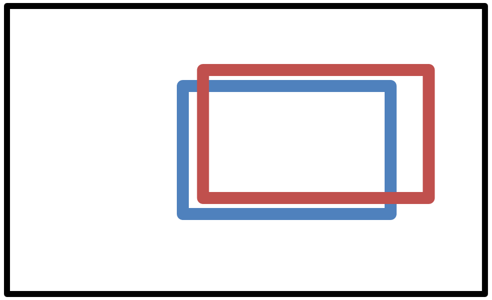
- a structured regression rather than classification

Structured Regression

- a structured regression rather than classification
- outputs are not independent of each other
  - right coordinate > left coordinate
  - bottom coordinate > top coordinate

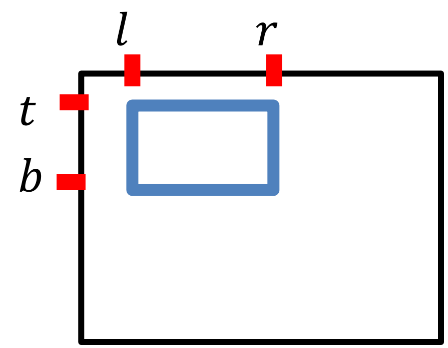


  - overlapping boxes should have similar objective



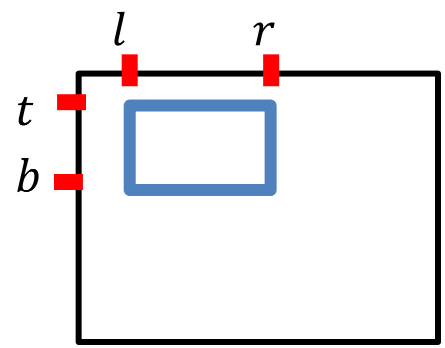
Object Localization as Structured Learning

- Given
  - Input images
    $\{x_1, \ldots, x_n\} \subset X$
  - Associated annotations
    $\{y_1, \ldots, y_n\} \subset Y$
    $$Y \equiv \left\{ (\omega, t, l, b, r) \,\middle|\, \begin{matrix} \omega \in \{+1, -1\}, \\ (t, l, b, r) \in \mathbb{R}^4 \end{matrix} \right\}$$



Object Localization as Structured Learning

- Given
  - Input images
    $\{x_1, \ldots, x_n\} \subset X$
  - Associated annotations
    $\{y_1, \ldots, y_n\} \subset Y$
    $$Y \equiv \left\{ (\omega, t, l, b, r) \,\middle|\, \begin{matrix} \omega \in \{+1, -1\}, \\ (t, l, b, r) \in \mathbb{R}^4 \end{matrix} \right\}$$



- Goal is to learn a mapping

$$g: X \to Y$$
$$g(x) = \underset{y}{\operatorname{argmax}} f(x, y)$$
$$f: X \times Y \to \mathbb{R}$$
$$f(x, y) = \langle w, \phi(x, y) \rangle$$

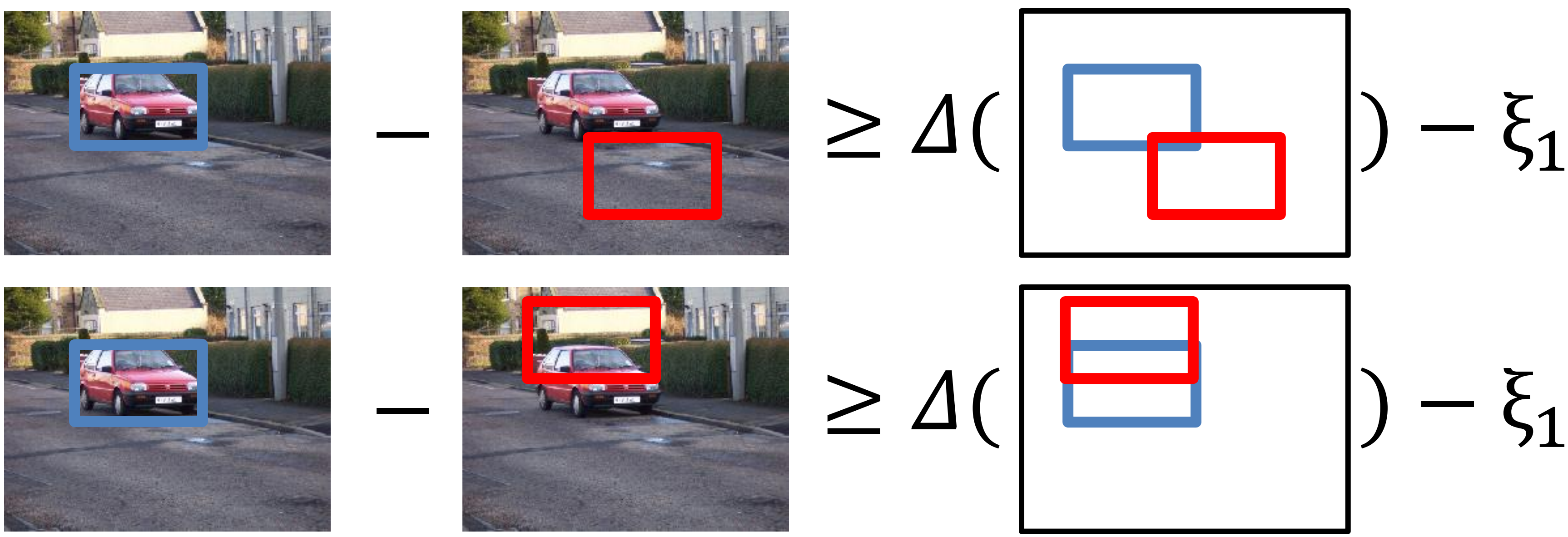
Object Localization as Structured Learning

- To train a discriminant function f

$$\min_{w,\xi} \ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{s.t.} \ \ \xi_i \geq 0, \quad \forall i$$

$$\langle w, \phi(x_i, y_i)\rangle - \langle w, \phi(x_i, y)\rangle \geq \Delta(y_i, y) - \xi_i, \quad \forall i, \forall y \in \mathcal{Y} \setminus y_i$$



Object Localization as Structured Learning

- To train a discriminant function f

$$\min_{w,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{s.t.} \quad \xi_i \geq 0, \quad \forall i$$

$$\langle w, \phi(x_i, y_i)\rangle - \langle w, \phi(x_i, y)\rangle \geq \Delta(y_i, y) - \xi_i, \quad \forall i, \forall y \in \mathcal{Y} \setminus y_i$$

⇓

$$\xi_i \geq \max_{y \in \mathcal{Y} \setminus y_i} \Delta(y_i, y) - \left(\langle w, \phi(x_i, y_i)\rangle - \langle w, \phi(x_i, y)\rangle\right), \quad \forall i$$

Object Localization as Structured Learning

- To train a discriminant function f

$$\min_{w,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{s.t. } \xi_i \geq 0, \quad \forall i$$

$$\langle w, \phi(x_i, y_i)\rangle - \langle w, \phi(x_i, y)\rangle \geq \Delta(y_i, y) - \xi_i, \quad \forall i, \forall y \in \mathcal{Y} \setminus y_i$$

⬇

$$\xi_i \geq \max_{y \in \mathcal{Y} \setminus y_i} \Delta(y_i, y) - (\langle w, \phi(x_i, y_i)\rangle - \langle w, \phi(x_i, y)\rangle), \quad \forall i$$

⬇

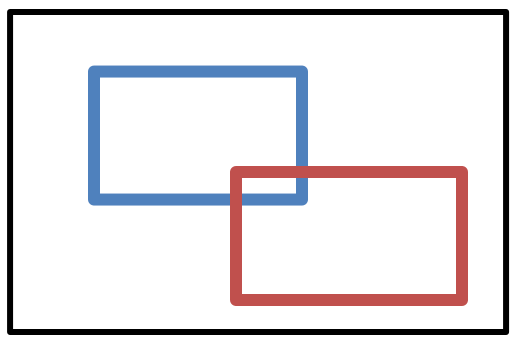
$$\max_{y \in \mathcal{Y} \setminus y_i} \Delta(y_i, y) + \langle w, \phi(x_i, y)\rangle$$

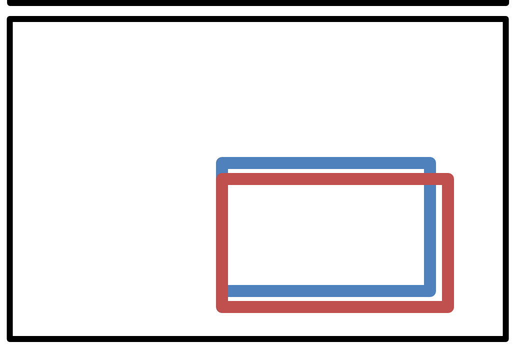
$$\max_{y \in \mathcal{Y} \setminus y_i} \Delta(y_i, y) + \sum_{j=1}^{n}\sum_{\tilde{y} \in \mathcal{Y}} \alpha_{j\tilde{y}} \left(k_x(x_j|_{y_j}, x_i|_y) - k_x(x_j|_{\tilde{y}}, x_i|_y)\right)$$

Loss Function

- Measure of overlap

$$\Delta(y_i, y) = \begin{cases} 1 - \frac{\text{Area}(y_i \bigcap y)}{\text{Area}(y_i \bigcup y)} & \text{if } y_{i\omega} = y_\omega = 1 \\ 1 - \left(\frac{1}{2}(y_{i\omega} y_\omega + 1)\right) & \text{otherwise} \end{cases}$$

$\Delta($

$) =$

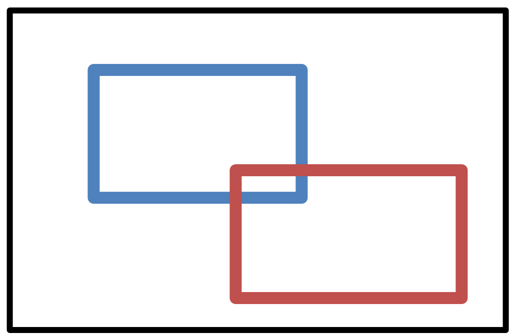
$\Delta($

$) =$

Loss Function

- Measure of overlap

$$\Delta(y_i, y) = \begin{cases} 1 - \frac{\text{Area}(y_i \bigcap y)}{\text{Area}(y_i \bigcup y)} & \text{if } y_{i\omega} = y_{\omega} = 1 \\ 1 - \left(\frac{1}{2}(y_{i\omega} y_{\omega} + 1)\right) & \text{otherwise} \end{cases}$$

$\Delta($



$) \approx 1$

$\Delta($

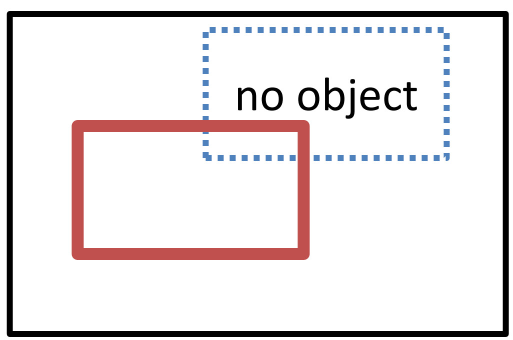
$) \approx 0$

Loss Function

- Measure of overlap

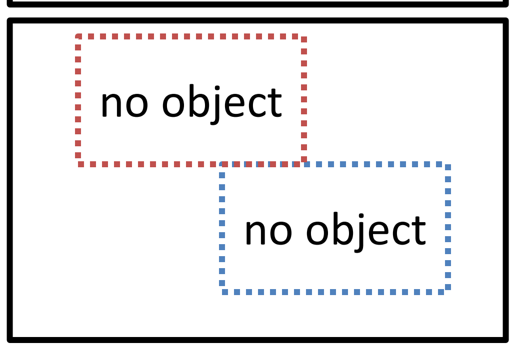
$$\Delta(y_i, y) = \begin{cases} 1 - \frac{\text{Area}(y_i \bigcap y)}{\text{Area}(y_i \bigcup y)} & \text{if } y_{i\omega} = y_\omega = 1 \\ 1 - \left(\frac{1}{2}(y_{i\omega} y_\omega + 1)\right) & \text{otherwise} \end{cases}$$

Δ (



) =

Δ (



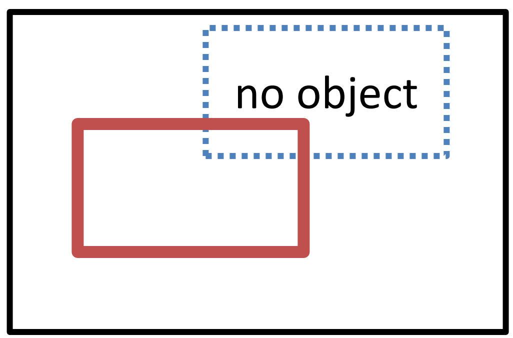
) =

Loss Function

- Measure of overlap

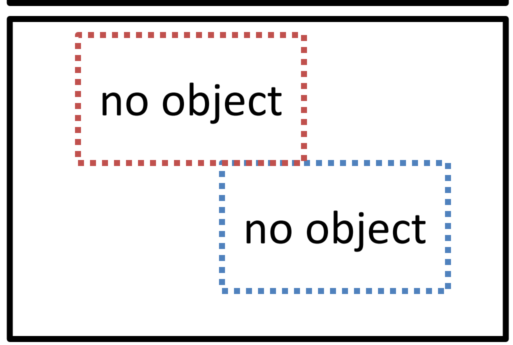
$$\Delta(y_i, y) = \begin{cases} 1 - \frac{\text{Area}(y_i \bigcap y)}{\text{Area}(y_i \bigcup y)} & \text{if } y_{i\omega} = y_\omega = 1 \\ 1 - \left(\frac{1}{2}(y_{i\omega} y_\omega + 1)\right) & \text{otherwise} \end{cases}$$

$\Delta($



$) = 1$

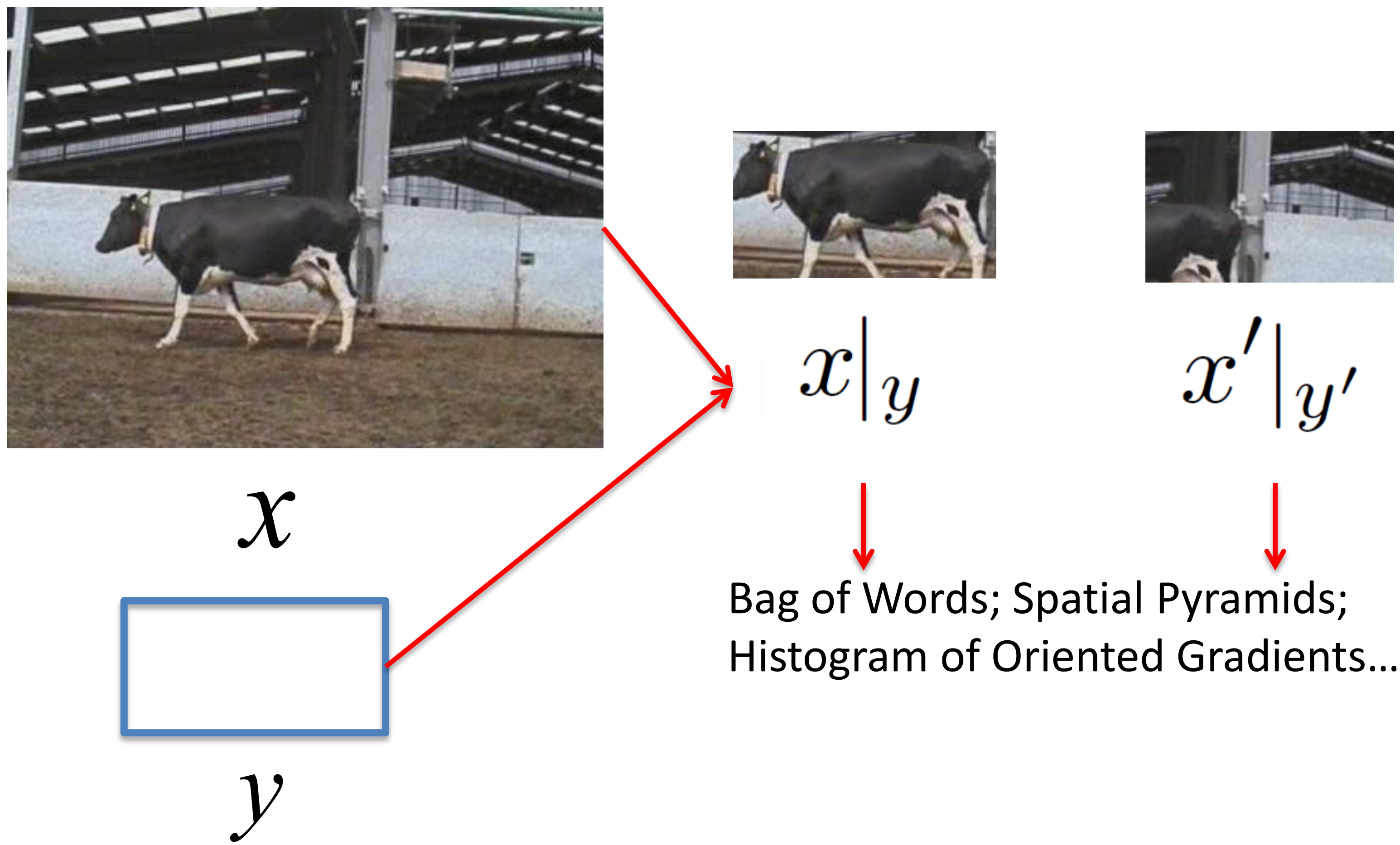
$\Delta($



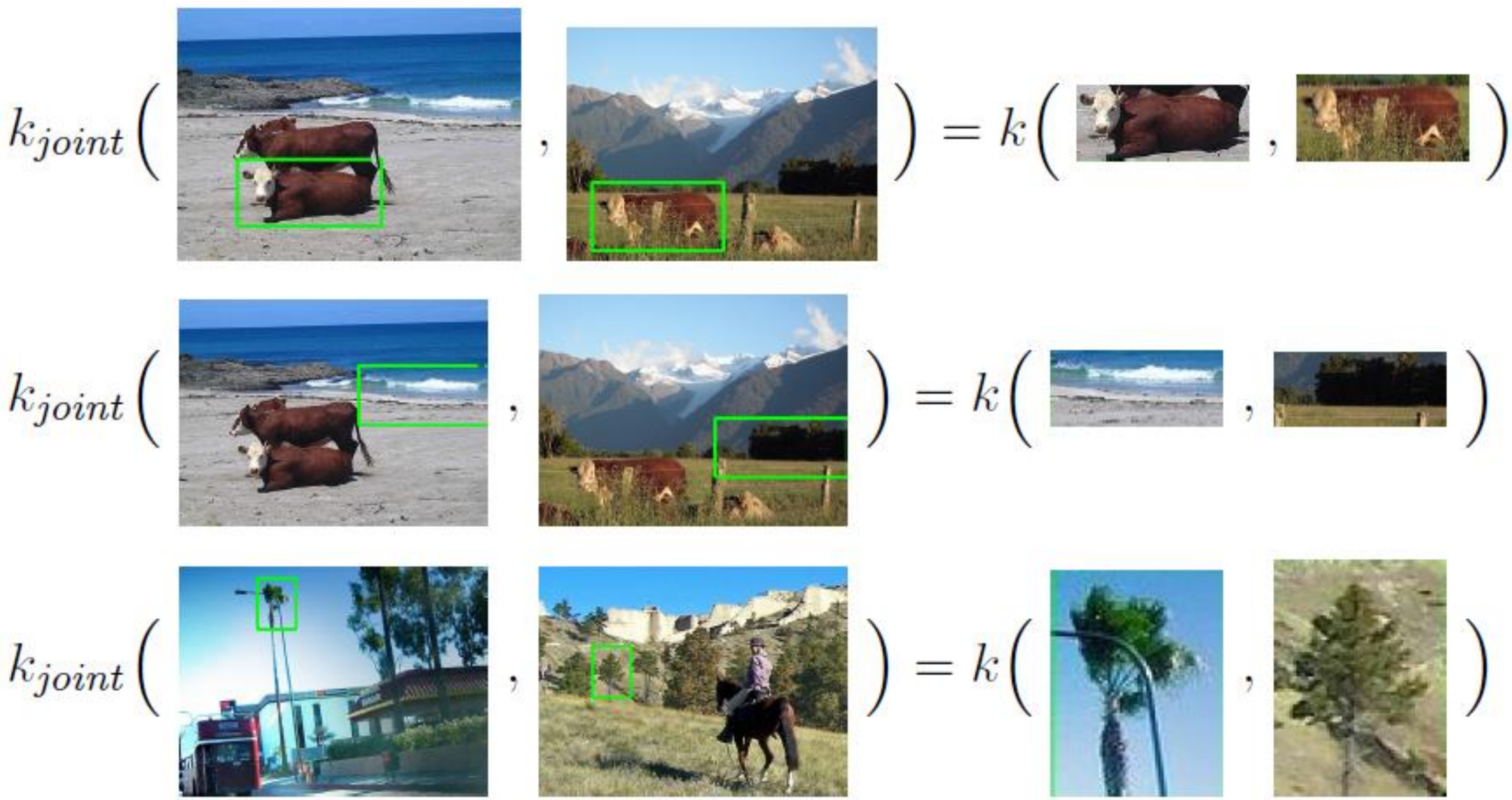
$) = 0$

Joint Kernel Map

$$k((x,y),(x',y')) = k_x(x|_y, x'|_{y'})$$

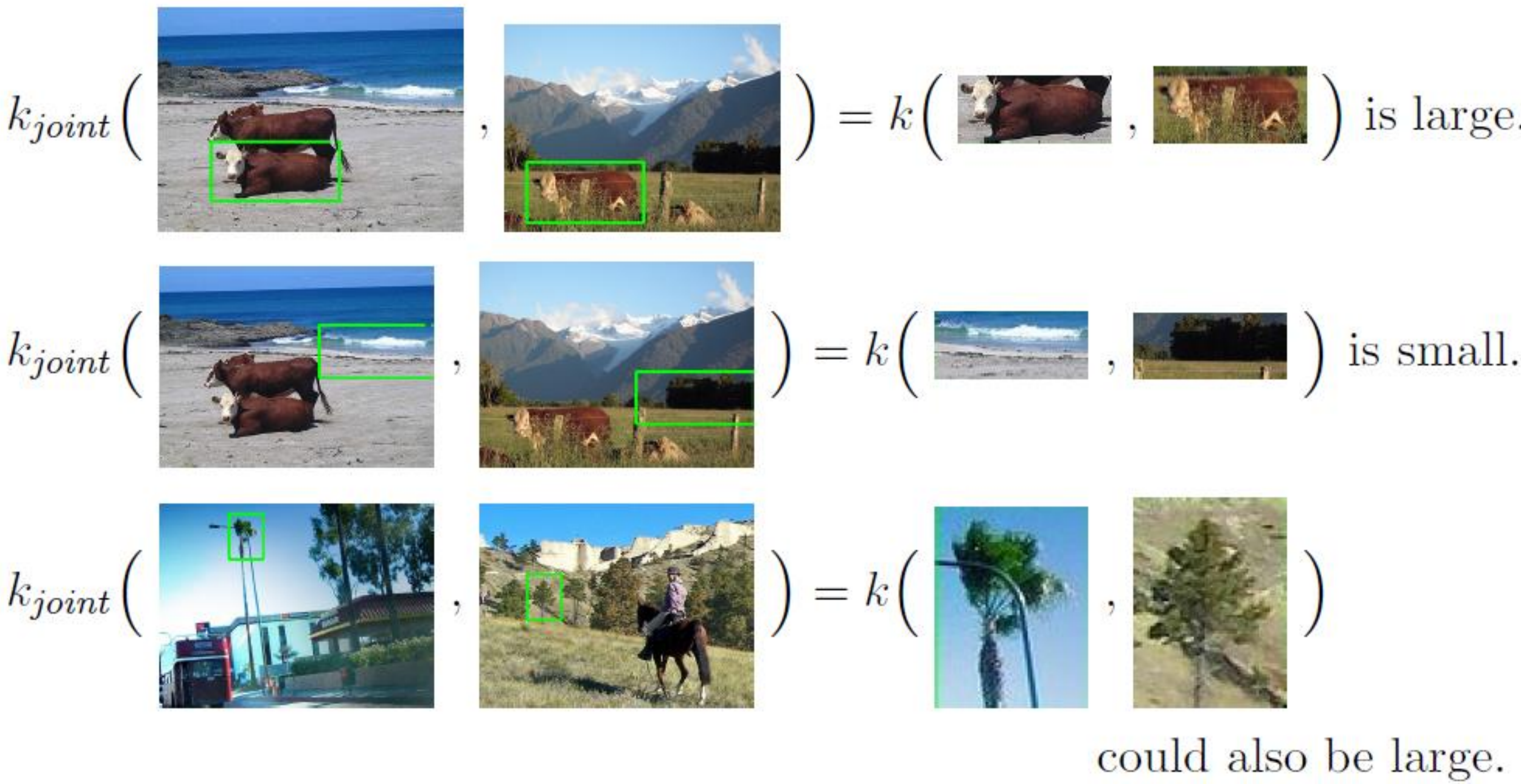


Joint Kernel Map for Localization



Slide from Blaschko and Lampert

Joint Kernel Map for Localization



Slide from Blaschko and Lampert

Maximization Step

- Training stage: $\max \Delta(y_i, y) + \langle w, \phi(x_i, y) \rangle$
- Testing stage: $\arg\max_{y \in Y} \langle w, \phi(x_i, y) \rangle$

- Exhaustive search computationally infeasible

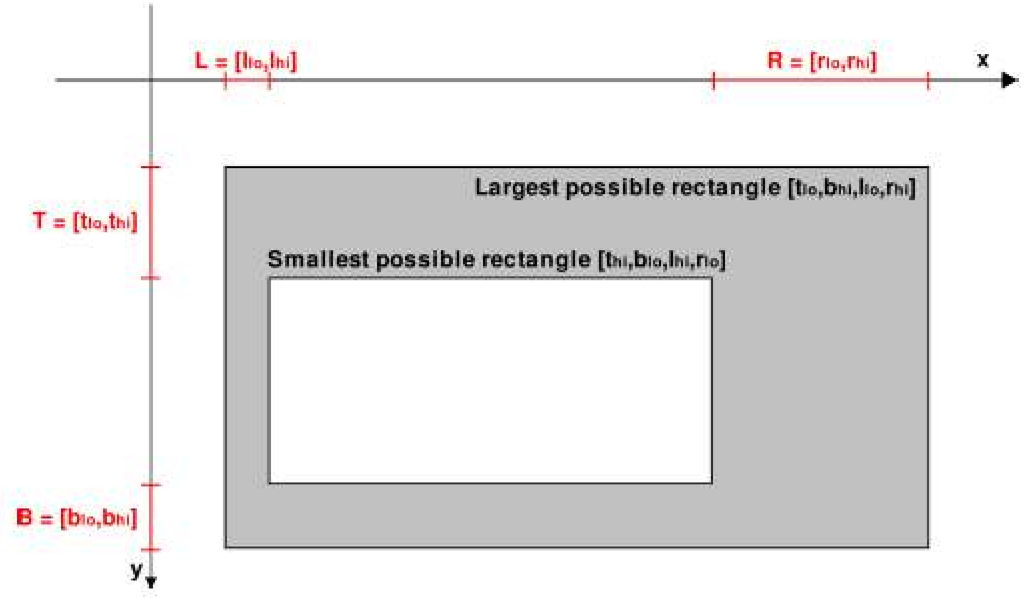
- Branch-and-bound optimization algorithm

Branch-and-bound: bounding box splitting

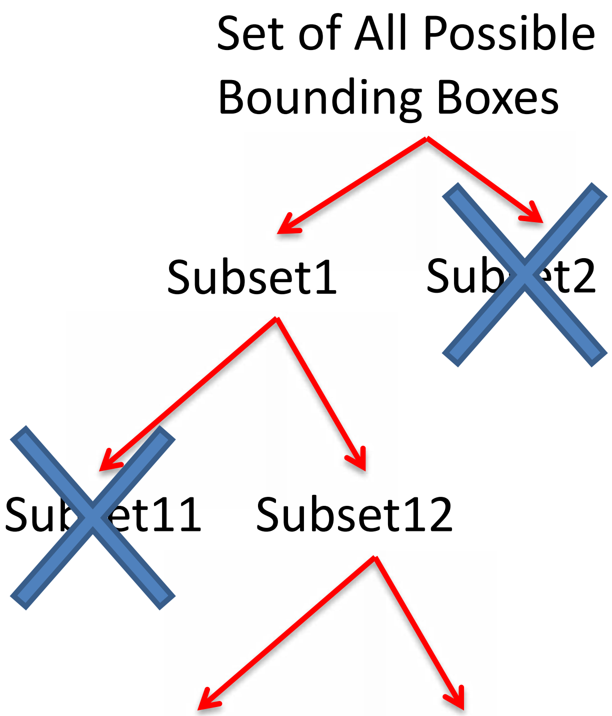
Branch-and-Bound works with subsets of the search space.

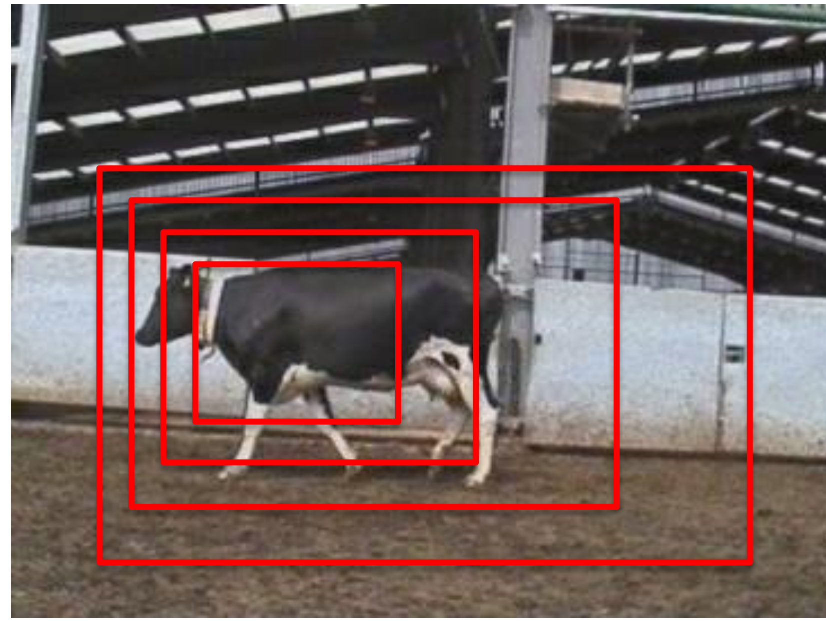
- Instead of four numbers $y = [l, t, r, b]$, store four intervals $Y = [L, T, R, B]$:

$$L = [\underline{l}, \overline{l}]$$
$$T = [\underline{t}, \overline{t}]$$
$$R = [\underline{r}, \overline{r}]$$
$$B = [\underline{b}, \overline{b}]$$



Branch-and-bound: branch step

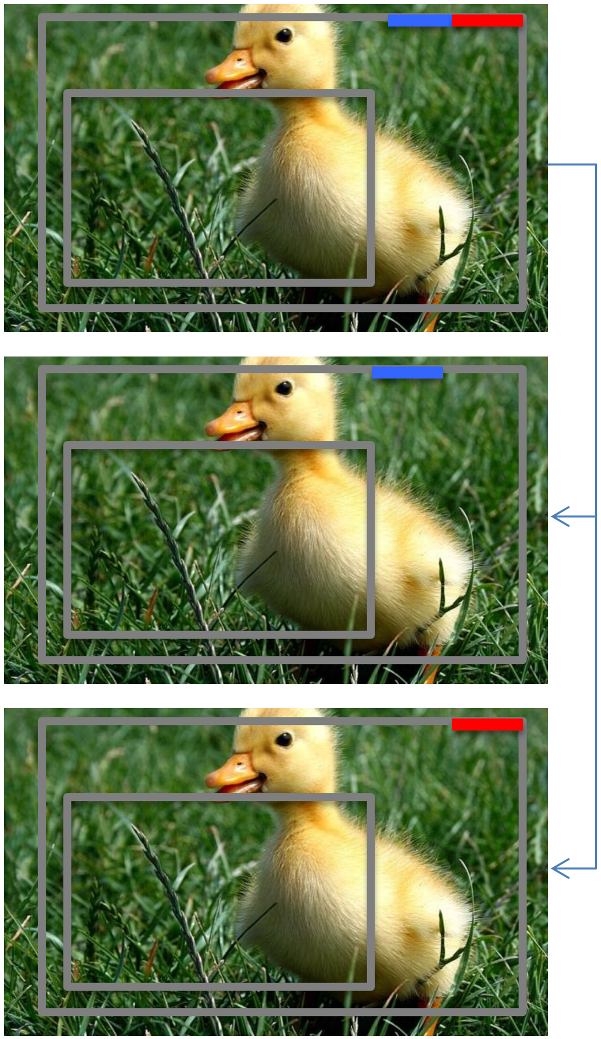




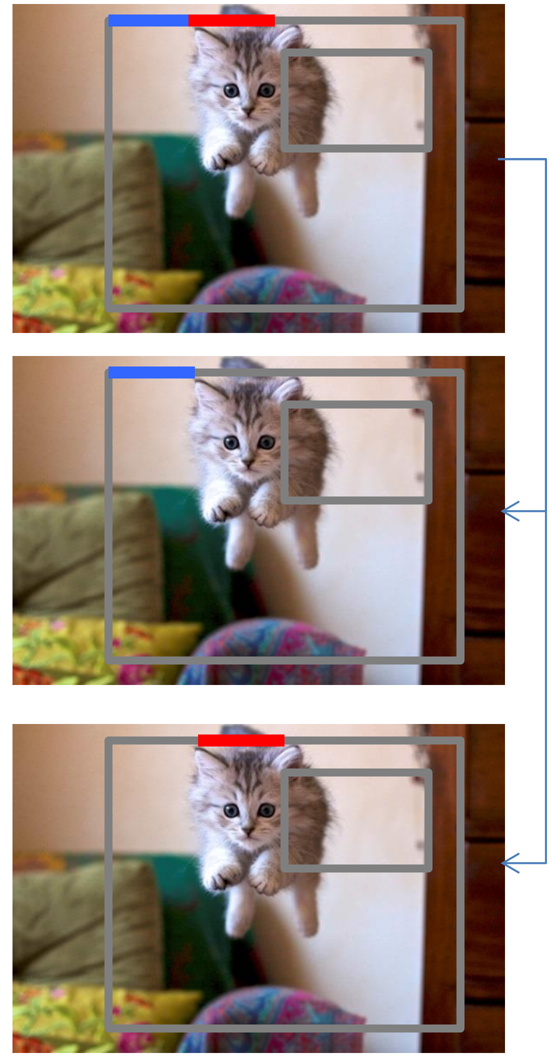
- Branching can be done by splitting image coordinates (left/right; top/bottom)
- Branch-and-bound is efficient because only the upper bound of a branch (a set of boxes) needs to be computed!

Each branch corresponds to a set of bounding boxes

Branch-and-bound: splitting examples

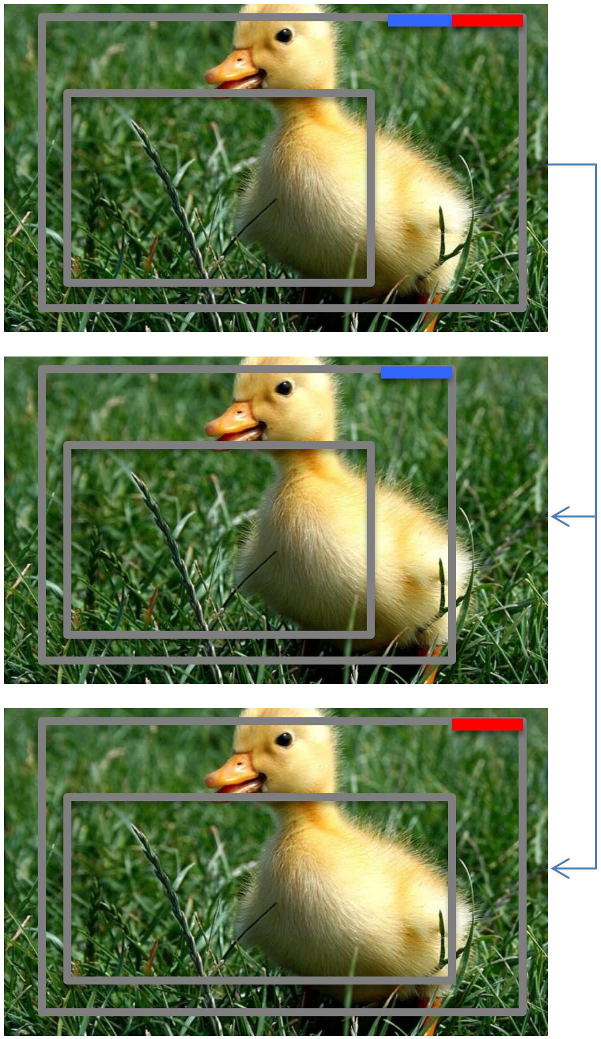


Splitting right coordinates

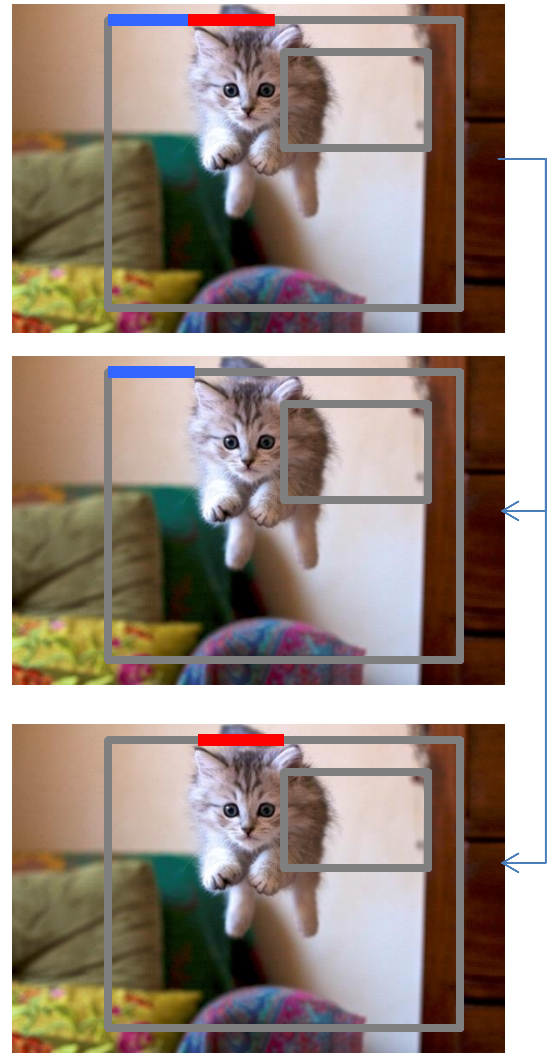


Splitting left coordinates

Branch-and-bound: splitting examples

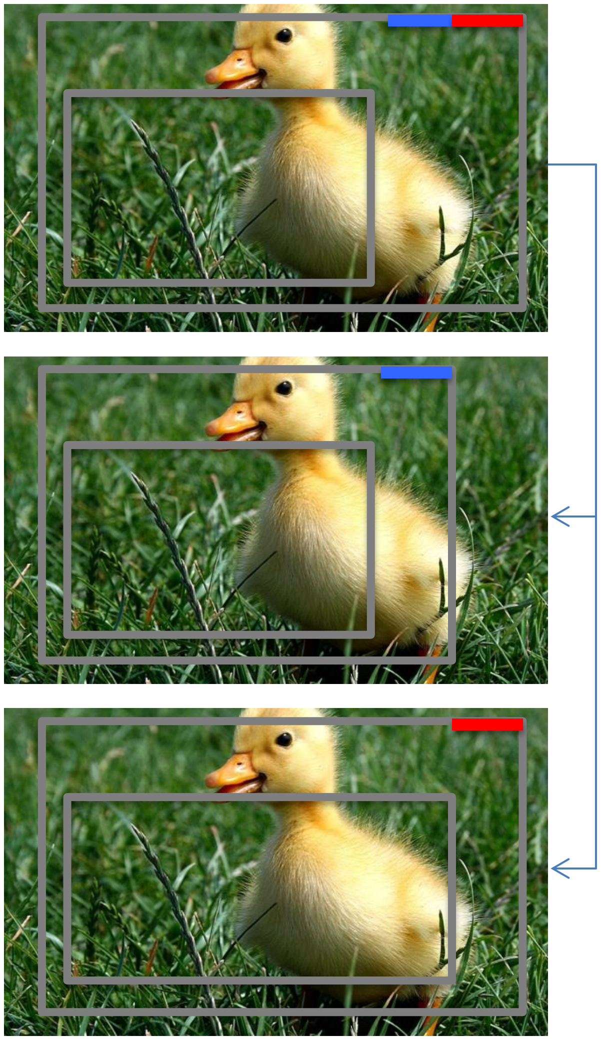


Splitting right coordinates

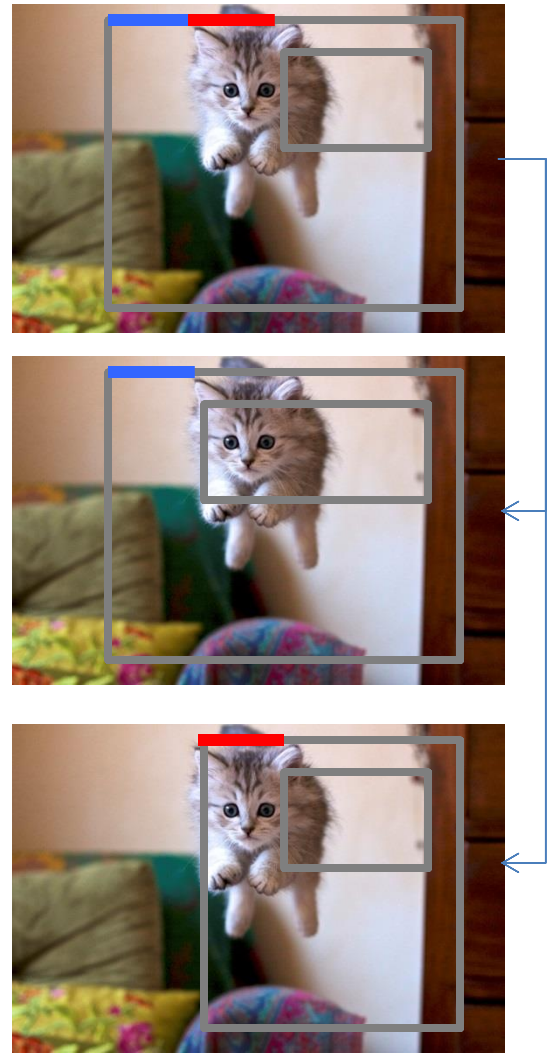


Splitting left coordinates

Branch-and-bound: splitting examples

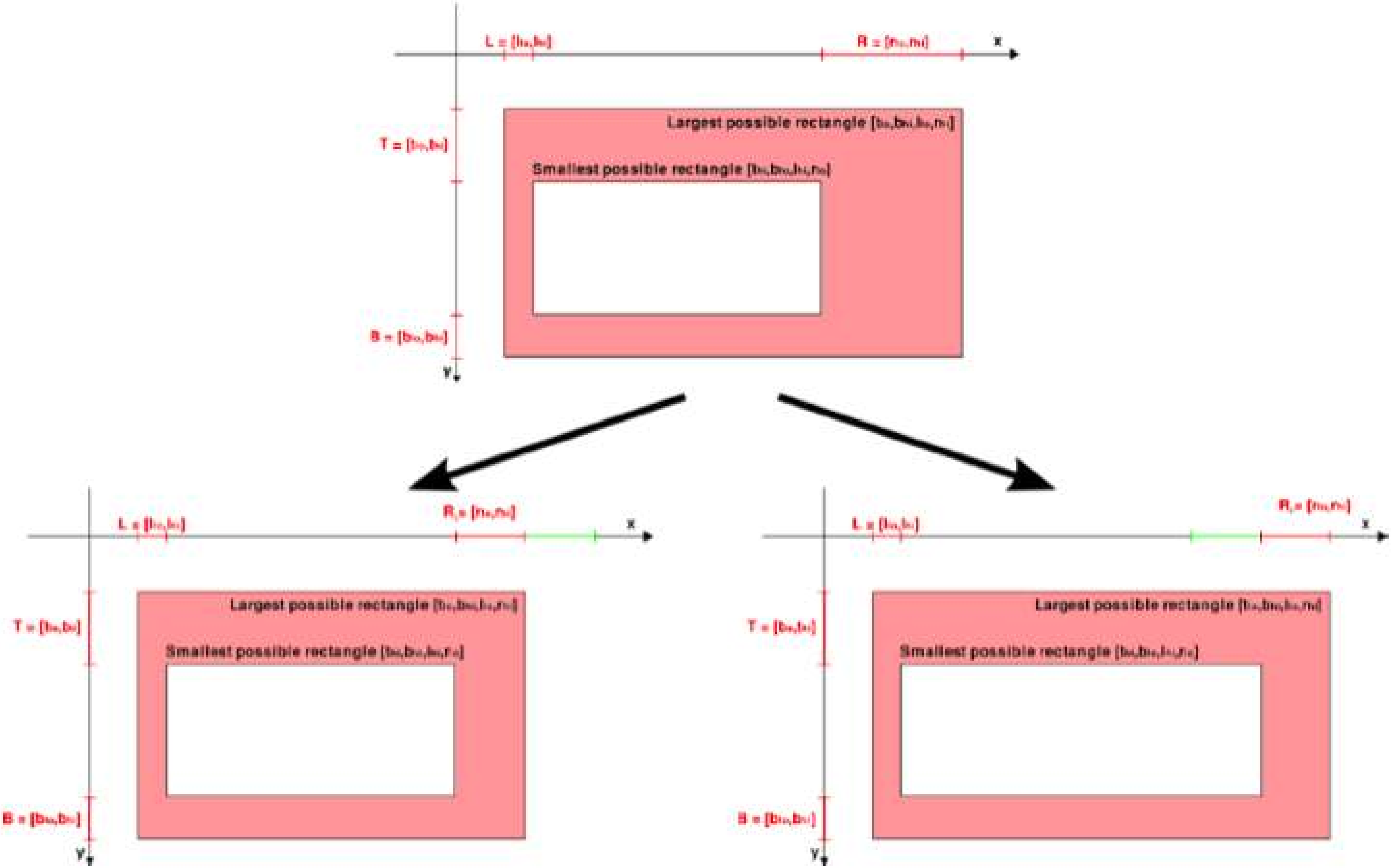


Splitting right coordinates



Splitting left coordinates

Branch-and-bound: bounding box splitting

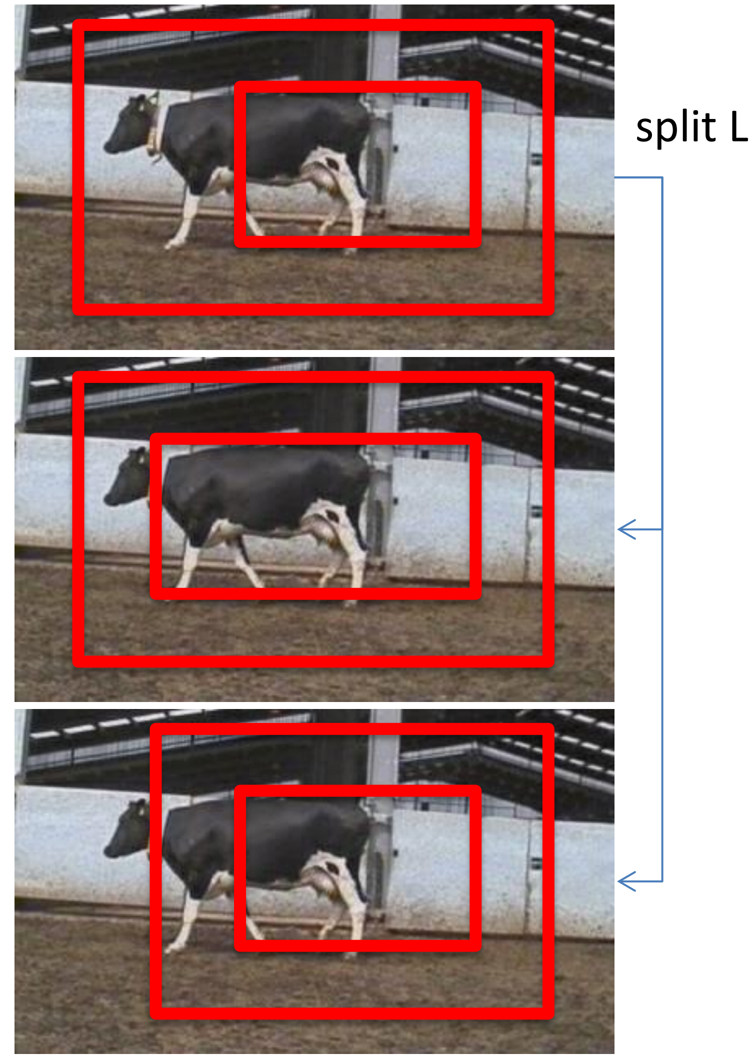


Branch-and-bound: quality function

A quality function to compute the upper bound for a set of boxes:

$$\hat{f}(R) = f^{+}(R_{max}) + f^{-}(R_{min})$$

- f^{+}: All positive features
- R_{max}: Maximum bounding box in a set
- f^{-}: All negative features
- R_{min}: Minimum bounding box in a set



Branch-and-bound: bound step

1. For each branching step, only keep the branch (set of boxes) with higher upper bound.

2. Create sub-branch for the current branch. Repeat 1 until there is only one box left.

Experiment: Dataset

- TU Darmstadt cows
  - 111 training images
  - 557 test images
- PASCAL VOC 2006
  - 5,304 images of 10 classes
  - Evenly split into a train/validation and a test part

Experiment: Setup

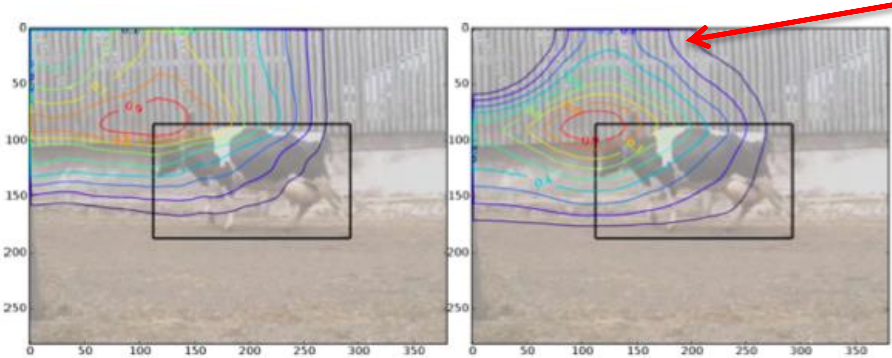
- Local SURF descriptors from feature points
  - 10,000 descriptors from training images
  - 3,000 entry visual codebook
- SVM^{struct} package was used.
- Benchmark against standard sliding window approach
  - Binary training
  - Linear image kernel over bag-of-visual-word histogram

Results: TU Darmstadt Cows

Performance at equal error rate (EER).

	Performance at ERR
Implicit Shape Model (ISM)	96.1%
Local Kernels (LK)	95.3%
LK + ISM	97.1%
Binary training	97.3%
Structured training	**98.2%**

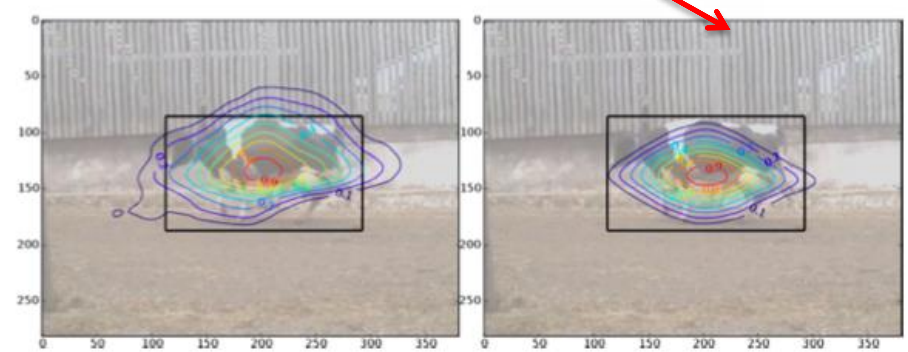
Tighter contour



Binary

Structured

Bottom right corner fixed



Binary

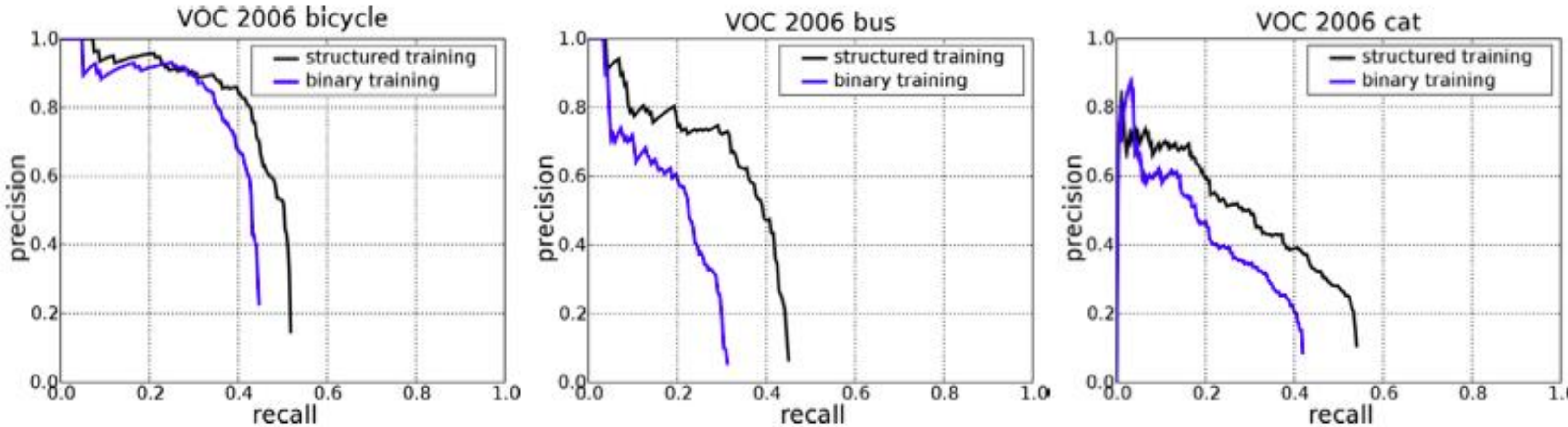
Structured

Box dimension fixed

Results: PASCAL VOC 2006

Precision-recall curves and example detections

Precision=TP/(TP+FP)
Recall=TP/(TP+FN)



Results: PASCAL VOC 2006

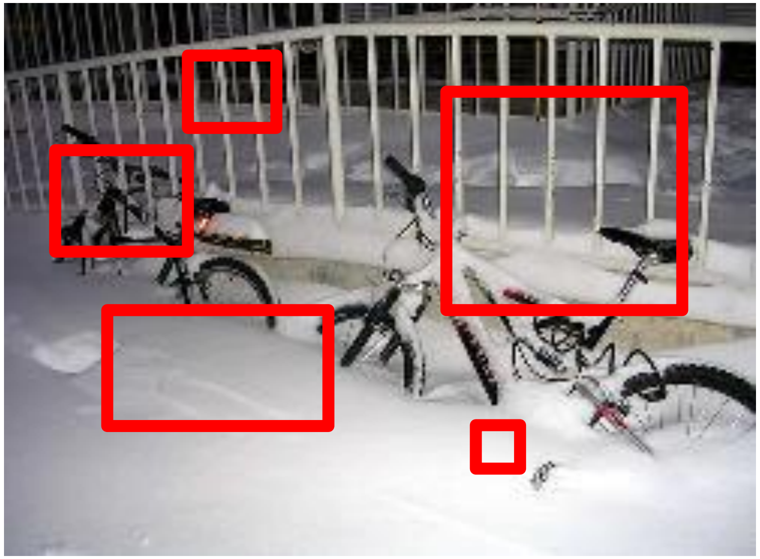
Average Precision Scores on the 10 categories of PASCAL VOC 2006

	bike	bus	car	cat	cow	dog	horse	m.bike	person	sheep
structured training	.472	**.342**	.336	**.300**	**.275**	.150	**.211**	**.397**	.107	.204
binary training	.403	.224	.256	.228	.114	**.173**	.137	.308	.104	.099
best in competition	.440	.169	.444	.160	.252	.118	.140	.390	.164	**.251**
post competition	**.498**†	.249‡	**.458**†	.223*	—	.148*	—	—	**.340**$^{+}$	—

Discussion and Conclusion

Structured training often exceeds state-of-the art performance.

- It has access to all possible bounding boxes.
- It is able to better handle partial detection problem.



Demo!