BIL-722 ADVANCED TOPICS IN COMPUTER VISION

Çağdaş Baş, N10266943

Paper: Searching for objects driven by context

Authors: Bogdan Alexe, Nicolas Heess, Yee Whye Teh, Vittorio Ferrari

PURPOSE: OBJECT DETECTION

- Among many problems, all the methods exhaustively search the object with help of the sliding windows approach.
  - All the methods evaluates all the possible windows.
  - This process is very slow and also unnatural.
- Cognitive search shows that humans don’t do that. Instead search intelligently.

PROPOSITION: INTELLIGENT SEARCH

- Learn an object's relative position to its surroundings.
- An ideal search strategy would be like this:
  1. W_1 is sky, cars occur below sky so look below.
  2. W_2 is road, cars occur on the road, look just below the road
  3. There is a car part inside W_3, look surrounding patches.
  4. W_4 is a car.

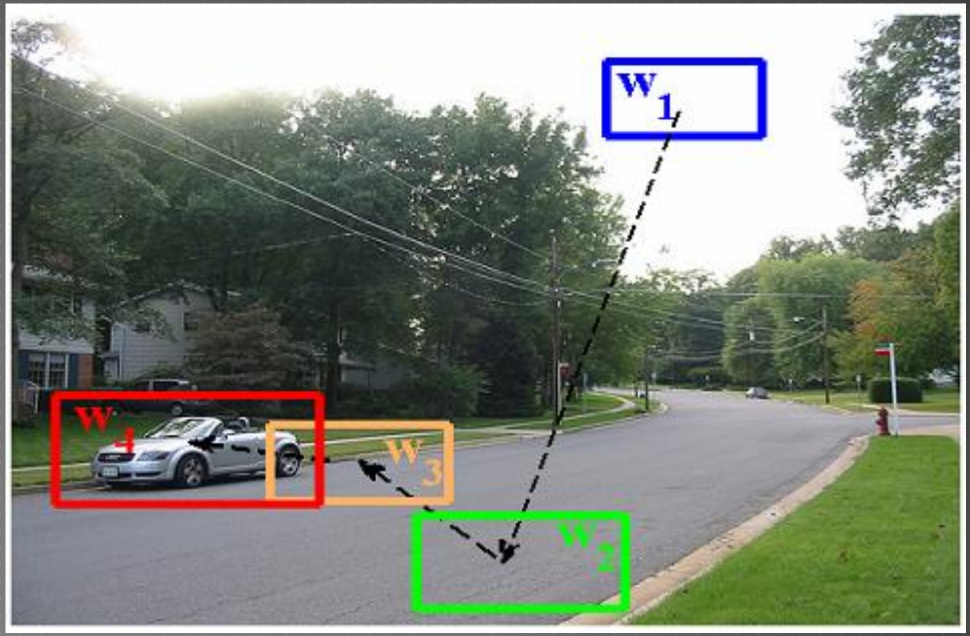


Figure Credit: Alexe Bogdan

OVERVIEW OF THE METHOD

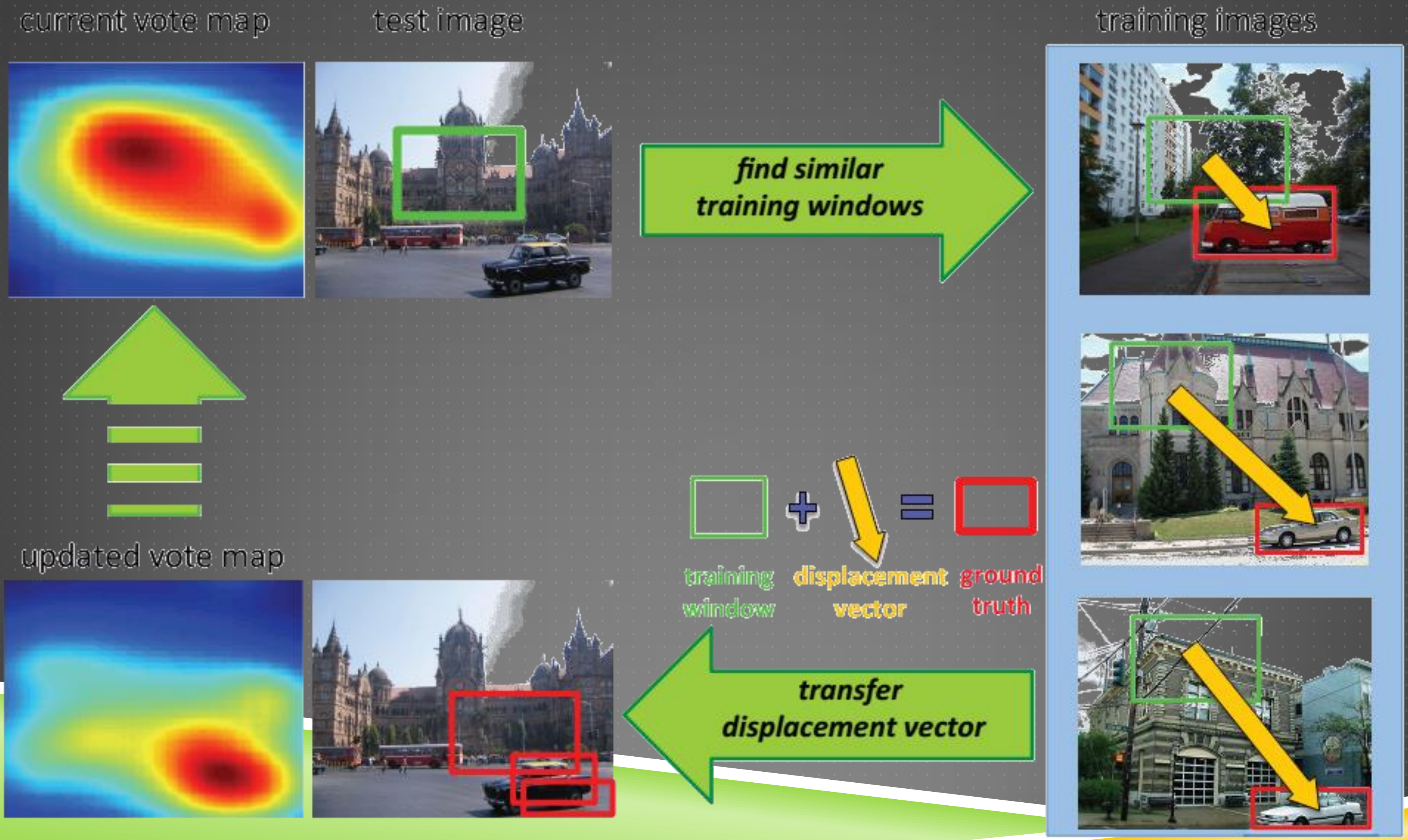


Figure Credit: Alexe Bogdan

ALGORITHM IN A NUTSHELL

1. Method randomly picks one window at the beginning.
2. Search Policy π^S:
   1. Similar position/appearance duo searched in the training set.
   2. Each of these similar patches votes for a new position.
   3. Method accumulates these votes as probability maps and decides where to look next.
3. Output Policy π^O:
   1. If current window similar enough to a car, search is over.

ALGORITHM IN DETAIL: FEATURE VECTOR

- A window is represented by these vector:

$$w^l = \left(x^l, y^l, s^l\right), y^t$$

Position

Scale

Feature vectors

- Window features y^t consists of:
  - Normalized location and scale of the window
  - HOG Histogram of the window
  - Classifier score
  - Displacement vector:
    - Intersection over union with the ground truth box
    - Normalized Hamming distance to the ground truth box
    - Absolute difference in the window classifier with the ground truth box

ALGORITHM IN DETAIL: SEARCH POLICY

- Extract uniformly distributed windows from all the training images, store features.
- For a test image:

1. Select a window, find it's K-NN from training windows.
2. Map new window and acquire the new probability map.
3. Choose next window with the highest probability:

$$\mathbf{w}^{t+1} = \pi^{S}(\mathbf{w}^1, \mathbf{y}^1, \ldots, \mathbf{w}^t, \mathbf{y}^t) = \arg\max_{\mathbf{w}} M^t(\mathbf{w} | \mathbf{w}^1, \mathbf{y}^1, \ldots, \mathbf{w}^t, \mathbf{y}^t; \Theta).$$

ALGORITHM IN DETAIL: SEARCH POLICY (2)

- Calculate probability map with the new window in test image w^t

$$\tilde{m}(\mathbf{w};\mathbf{w}^t,\mathbf{y}^t,\Theta) = \sum_{l=1}^{L} \underbrace{K_F(\mathbf{y}^t,\mathbf{y}^l;\Theta^F)}_{\text{Feature similarity kernel}} \cdot \underbrace{K_S(\mathbf{w},\mathbf{w}^t \oplus \mathbf{d}^l;\Theta^S)}_{\text{Spatial Smoothing Kernel}}.$$

  - w^t: Current window in test image.
  - w^l:Window from training set.

ALGORITHM IN DETAIL: SEARCH POLICY (3)

- Normalize each probability map and integrate all the past maps.

$$\tilde{m}(\mathbf{w};\mathbf{w}^t,\mathbf{y}^t,\Theta)=\sum_{l=1}^{L}\underbrace{K_F(\mathbf{y}^t,\mathbf{y}^l;\Theta^F)}_{\text{Feature similarity kernel}}\cdot\underbrace{K_S(\mathbf{w},\mathbf{w}^t\oplus\mathbf{d}^l;\Theta^S)}_{\text{Spatial Smoothing Kernel}}.$$

- Integrate all maps to form the overall probability map using exponentially decaying mixture.

$$M^t(\mathbf{w}|\mathbf{w}^1,\mathbf{y}^1,\ldots,\mathbf{w}^t,\mathbf{y}^t;\Theta)=\sum_{t'=1}^{t}a(t,t')m(\mathbf{w}|\mathbf{w}^{t'},\mathbf{y}^{t'},\Theta),$$

ALGORITHM IN DETAIL: OUTPUT POLICY

- After T iteration, output a single window which has highest classification score amongst all:

$$w^{out} = arg\max_{t} c(w^t)$$

- This is a downside. Method assumes that there is only one instance in the image.

ALGORITHM IN DETAIL: LEARNING WEIGHTS

- There is a weight for each class in similarity kernel stage.
- This weights defines each patch's importance for each object class.

$$\mathcal{J}(\Theta^F; \hat{\mathbf{h}}) = \sum_{t=1}^{T} \sum_{\mathbf{w}} M^t(\mathbf{w} | \hat{\mathbf{w}}^1, \hat{\mathbf{y}}^1, \ldots, \hat{\mathbf{w}}^t, \hat{\mathbf{y}}^t; \Theta) \cdot K_S(\mathbf{w}, \mathbf{w}^b_{\mathrm{GT}})$$

OBJECT CLASSIFIER

- An object classifier is trained for each class.
  - For each class, one root HOG filter and several part HOG filters are trained.
  - Root and part filters summed with weights according to Felzenswab's work.
- For each class, training split is used for classifier learning.

EXPERIMENTS

- Experiments conducted on PASCAL VOC 2010 dataset.
  - A highly challenging dataset which contains 20 object classes witch bounding box annotations.
- Validation set is used for testing.
- Mean Average Precision over all classes and detection rate and number of windows evaluated by the detector used as performance measures.

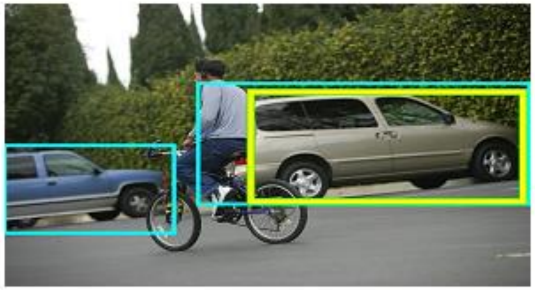
EXPERIMENTS: QUANTITATIVE

	Sliding Window [12]	Our	Random Chance	Objectness [1]	Selective Search [29]
mAP	0.266	**0.293**	0.070	0.259	0.261
DR	0.372	**0.409**	0.124	0.366	0.370
#win	25000	**100**	100	1046	408

EXPERIMENTS: QUALITATIVE

(motorbike-left)

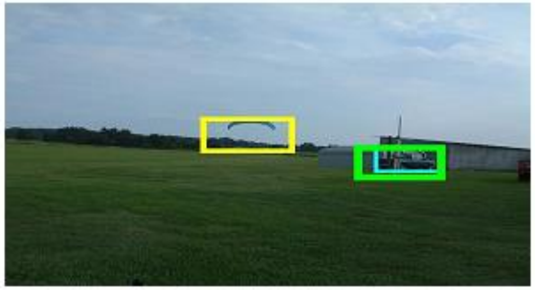
(horse-right-rear)



(car-right)

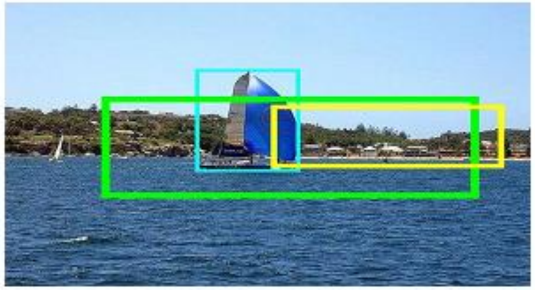
(boat-rear)

(cow-left)



(car-right)

(chair-frontal)

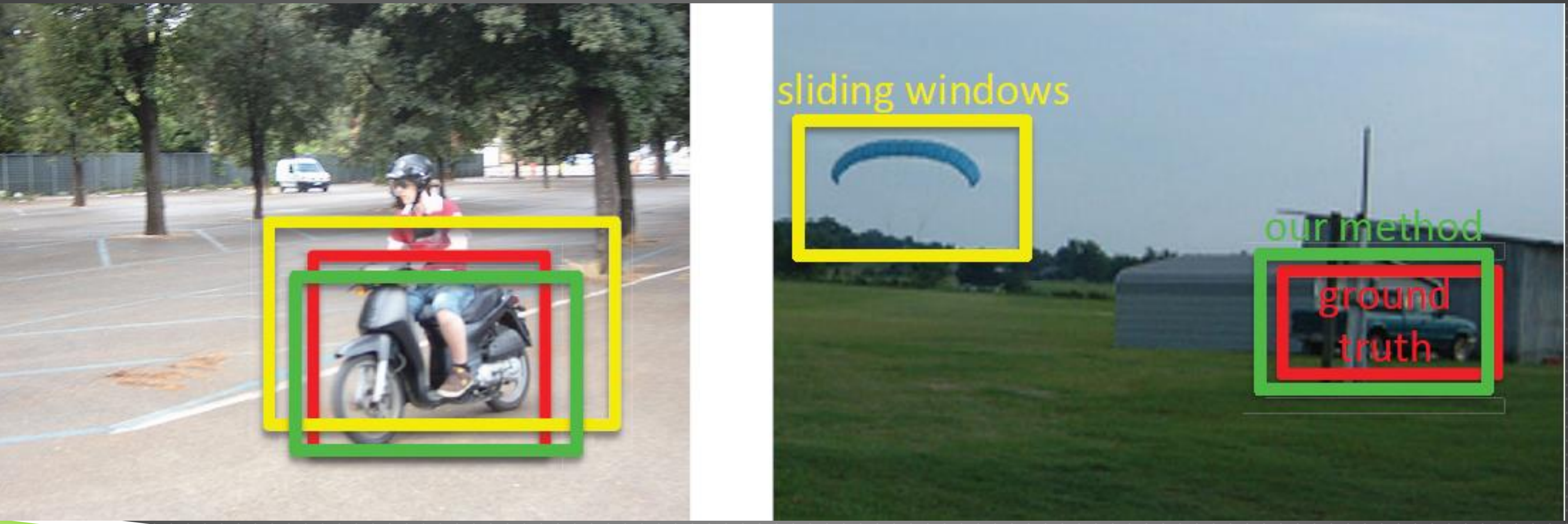


(boat-right)

(tvmonitor-frontal)

EXPERIMENTS: QUALITATIVE

- Comparison of ? With Felzenszwalb et al. PAMI 2010



EXPERIMENTS: PERFORMANCE

- Experiments run on a Intel i7 processor powered PC.
- It can be seen that compared window count is significantly lower than the usual deformable part model approach.
- It is said that deformable part model approach takes 92s while proposed method takes only 2s.

	Sliding Window [12]	Our	Random Chance	Objectness [1]	Selective Search [29]
mAP	0.266	**0.293**	0.070	0.259	0.261
DR	0.372	**0.409**	0.124	0.366	0.370
#win	25000	**100**	100	1046	408

PROS - CONS

- Pros:
  - Fast and logical search
  - Can be applied with any classifier/feature
- Cons:
  - Assumes only one instance exists.
  - Dataset dependent?

THANKS