# Object Detection

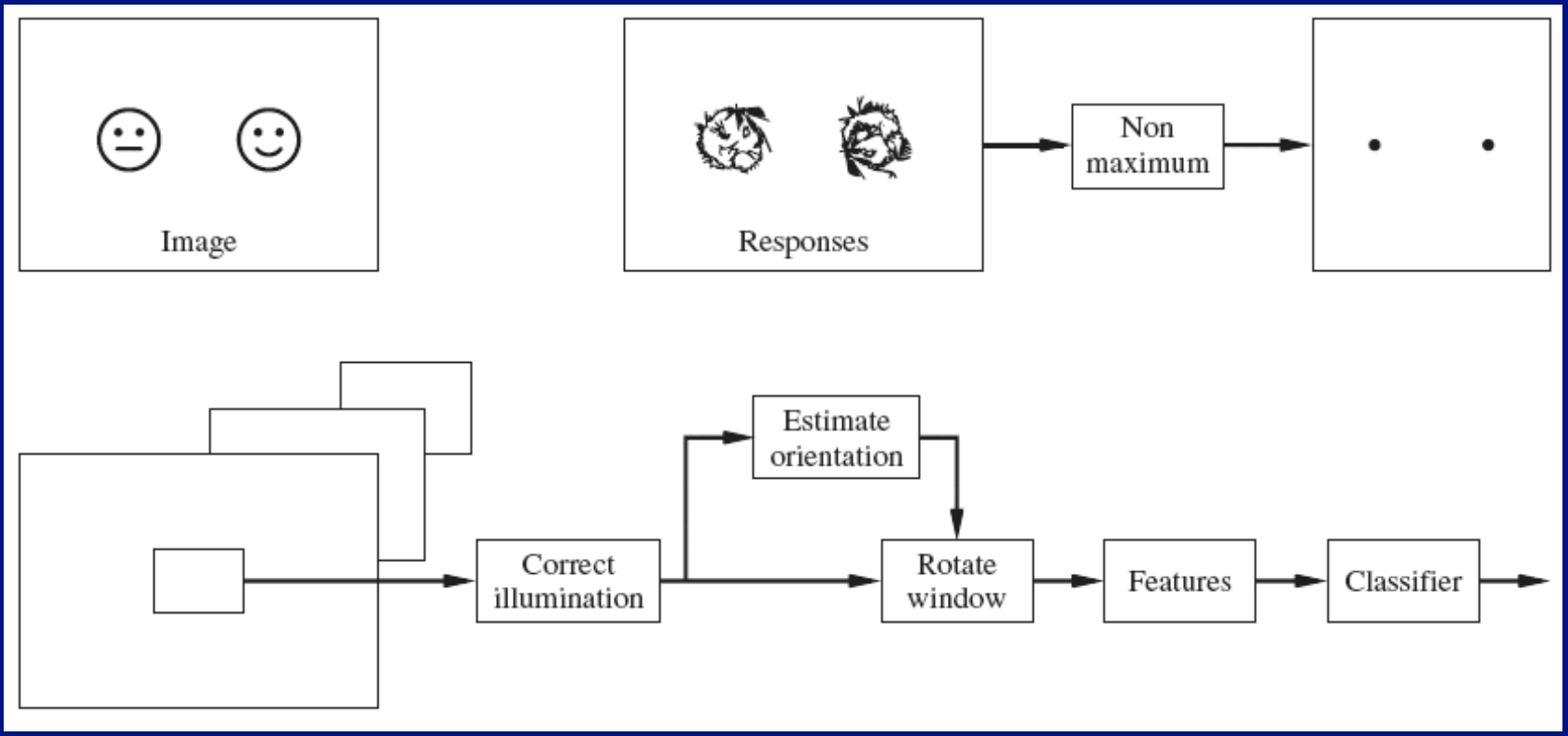
D.A. Forsyth

# Detection with a classifier

- Search
  - all windows
  - at relevant scales
- Prepare features
- Classify

- Issues
  - how to get only one response
  - speed
  - accuracy

# Detection with a classifier



Train a classifier on $n \times m$ image windows. Positive examples contain the object and negative examples do not.
Choose a threshold $t$ and steps $\Delta x$ and $\Delta y$ in the $x$ and $y$ directions

Construct an image pyramid.

For each level of the pyramid
  Apply the classifier to each $n \times m$ window, stepping by
  $\Delta x$ and $\Delta y$, in this level to get a response strength $c$.
  If $c > t$
    Insert a pointer to the window into a ranked list $\mathcal{L}$, ranked by $c$.

For each window $\mathcal{W}$ in $\mathcal{L}$, starting with the strongest response
  Remove all windows $\mathcal{U} \neq \mathcal{W}$ that overlap $\mathcal{W}$ significantly,
    where the overlap is computed in the original image by expanding windows
    in coarser scales.

$\mathcal{L}$ is now the list of detected objects.

**Algorithm 17.1:** Sliding Window Detection.

# Non-maximum suppression

- Compute “strength of response”
  - SVM value
  - LR value
- threshold
  - small values are not faces
- find largest value (over location, scale)
  - suppress nearby values
  - repeat

# Simplest, standard strategy

- Window features are HOG features
- Classifier is linear SVM

- ALWAYS try this first

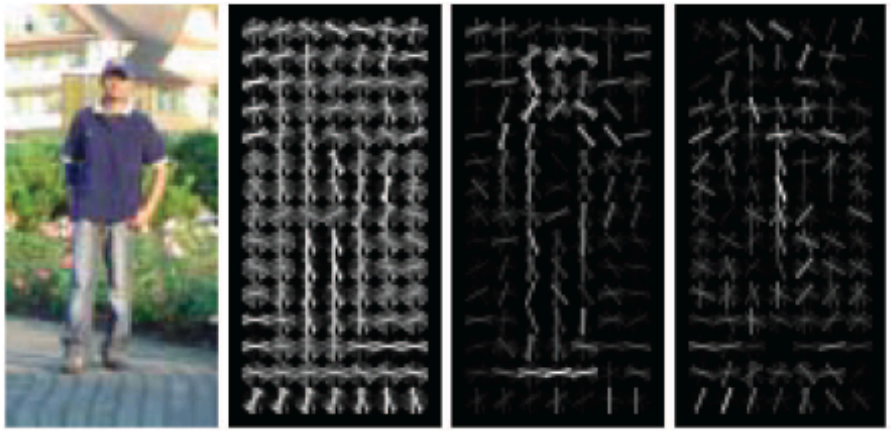
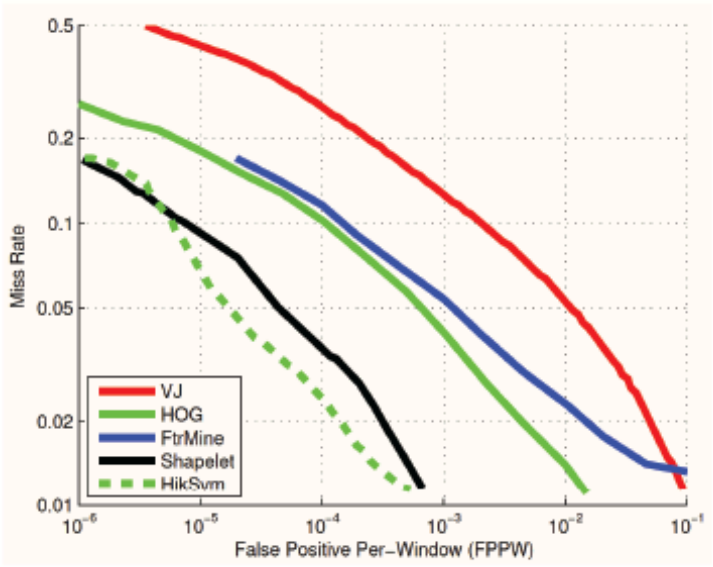


FIGURE 17.7: As Figure 17.6 indicates, a linear SVM works about as well as the best detector for a pedestrian detector. Linear SVMs can be used to visualize what aspects of the feature representation are distinctive. On the **left**, a typical pedestrian window, with the HOG features visualized on the **center left**, using the scheme of Figure 5.15. Each of the orientation buckets in each window is a feature, and so has a corresponding weight in the linear SVM. On the **center right**, the HOG features weighted by positive weights, then visualized (so that an important feature is light). Notice how the head and shoulders curve and the lollipop shape gets strong positive weights. On the **right**, the HOG features weighted by the absolute value of negative weights, which means a feature that strongly suggests a person is not present is light. Notice how a strong vertical line in the center of the window is deprecated (because it suggests the window is not centered on a person). *This figure was originally published as Figure 6 of "Histograms of Oriented Gradients for Human Detection," N. Dalal and W. Triggs, Proc. IEEE CVPR 2005, © IEEE, 2005.*

# Evaluation

- plot miss rate against false-positive per image rate
  - there are alternatives
    - total error rate
    - accuracy
    - loss



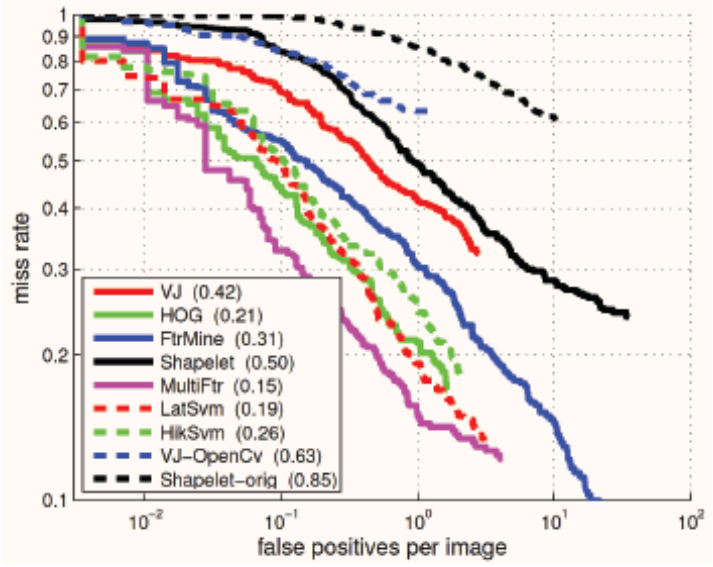




FIGURE 17.8: The FPPW statistic is useful for evaluating classifiers, but less so for evaluating systems. On the **left**, results on the INRIA pedestrian dataset for a variety of systems, plotted using miss rate against FPPW by Dollar *et al.* (2009). In this plot, curves that lie lower on the figure represent better performance (because they have a lower miss rate for a given FPPW rate). On the **right**, results plotted using miss rate against false positive per image (FPPI), a measure that takes into account the number of windows presented to the classifier. Again, curves that lie lower are better. Notice how different the ranking of the systems is. *This figure was originally published as Figure 8 of "Pedestrian Detection: A Benchmark" P. Dollár, C. Wojek, B. Schiele, and P. Perona, Proc. IEEE CVPR 2009 © IEEE 2009.*

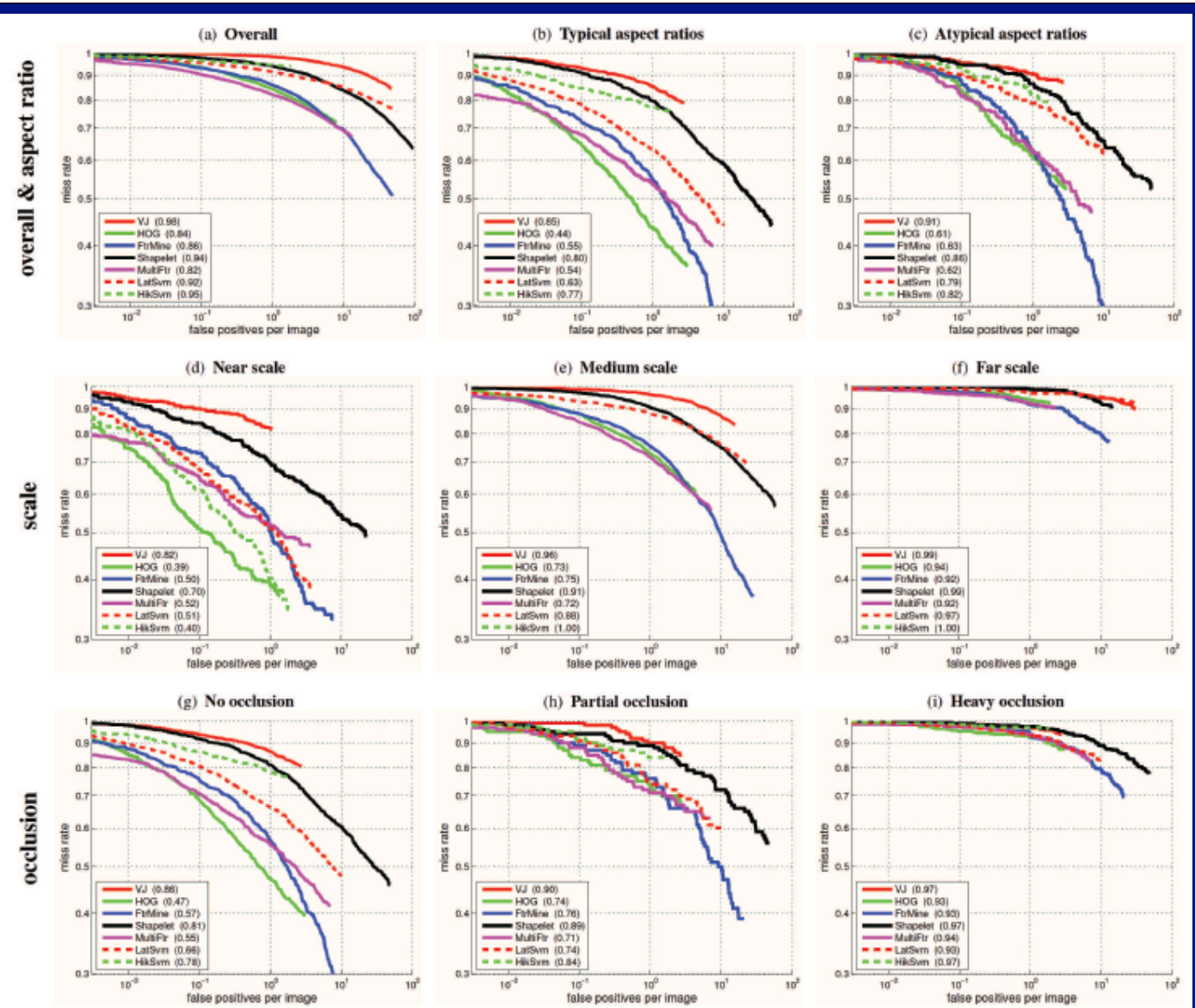



FIGURE 17.16: Performance of various pedestrian detectors for a test dataset (**top left**, labeled "overall") and for various special subsets of that test dataset, performed using the testbed of Dollar *et al.* (2009). Detectors that do well for some cases (for example, near-scale pedestrians) can do poorly for other cases (for example, medium-scale pedestrians). Some cases are hard for all detectors. *This figure was originally published as Figure 9 of "Pedestrian Detection: A Benchmark" P. Dollár, C. Wojek, B. Schiele, and P. Perona, Proc. IEEE CVPR 2009 © IEEE 2009.*

# Deformable objects

- Build several detectors for each object
  - to cover, say, different views
- Each detector computes a score that is a sum of
  - local part scores
    - each is linear SVM+HOG
  - position scores for the local part
- This can be learned from training data

http://people.cs.chicago.edu/~pff/latent/

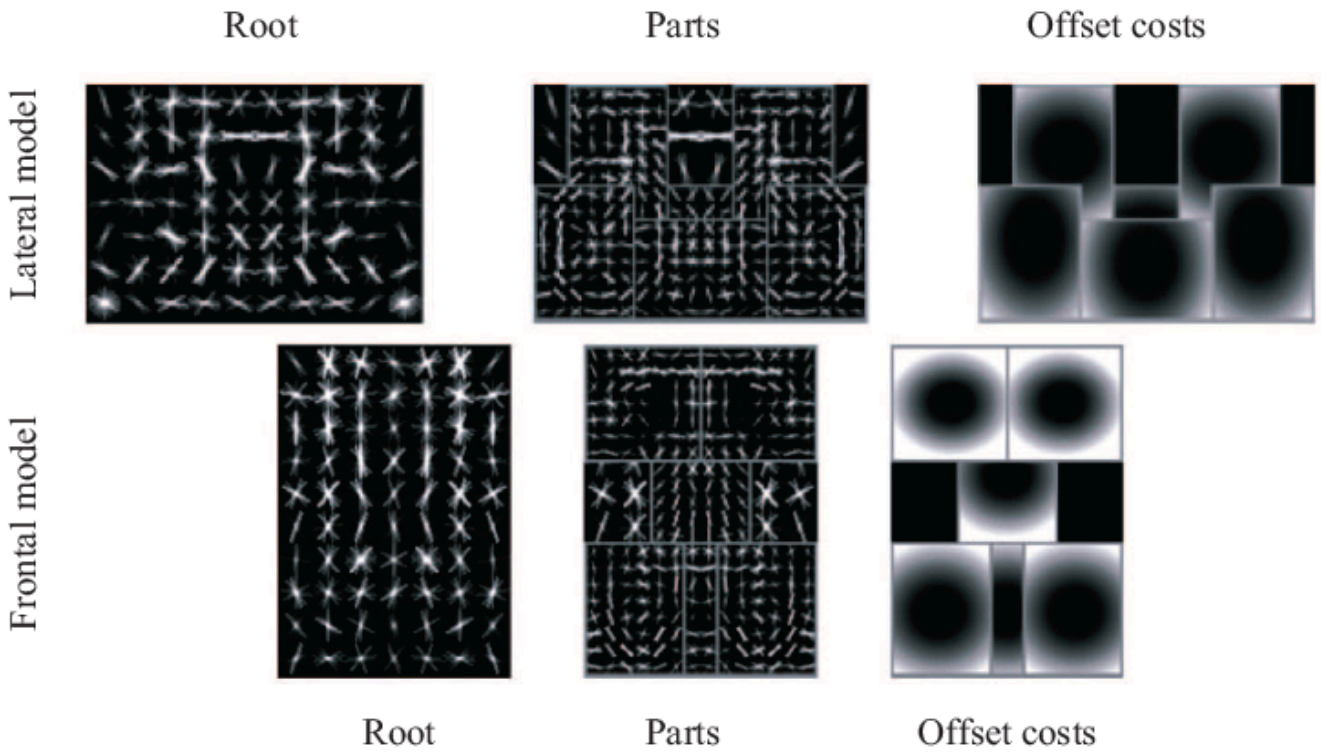



FIGURE 17.14: A model for a bicycle, built using the scheme of Felzenszwalb *et al.* (2010*b*). There are two components, corresponding to a frontal and a lateral view. Each component has a root and six parts. The root and the part appearance models are visualized with the scheme of Figure 5.15. Notice how the root for each view corresponds to a rough layout, but (for example) the wheels in the lateral view or the handlebar in the frontal view are hard to spot. This is because bicycles will not be in exactly the same place, or at exactly the same orientation, in each window. The parts can compensate for that, and the part models show quite clear wheels and handlebars. The offset costs are registered to the parts, and smaller values are darker. For example, the wheels in the lateral view can move somewhat apart, but it becomes expensive to separate them by too much, or place them too close together. The score for a particular image window is the maximum of the component scores, which are described in the text. *This figure was originally published as Figure 2 of "Object Detection Using Discriminatively Trained Part-based Models," by P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2010* © *IEEE 2010.*

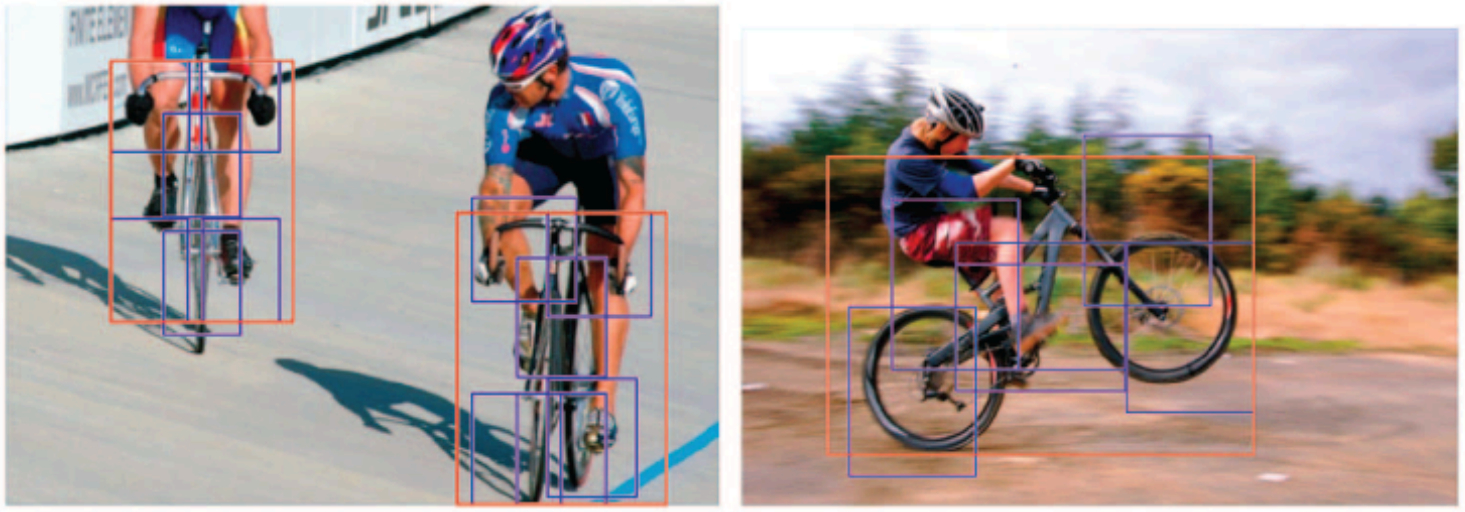


FIGURE 17.15: Examples of bicycles detected using the model of Figure 17.14. The large boxes are bicycle instances; the smaller boxes inside are the locations of the detected parts. The wheelie is not detected by rotating the box, but because the parts are allowed to move within the box. *This figure was originally published as Figure 2 of "Object Detection Using Discriminatively Trained Part-based Models," by P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2010* © *IEEE 2010.*

# Evaluation

- Work with boxes
  - Put boxes around targets
  - Detector reports boxes
  - hit if overlap is good enough
- Recall, precision, AP measures for boxes

| Category | 2007 | 2008 | 2009 | 2010 |
|---|---|---|---|---|
| aeroplane | 0.262 | 0.365 | 0.478 | 0.584 |
| bicycle | 0.409 | 0.420 | 0.468 | 0.553 |
| bird | 0.098 | 0.113 | 0.174 | 0.192 |
| boat | 0.094 | 0.114 | 0.158 | 0.210 |
| bottle | 0.214 | 0.282 | 0.285 | 0.351 |
| bus | 0.393 | 0.238 | 0.438 | 0.555 |
| car | 0.432 | 0.366 | 0.372 | 0.491 |
| cat | 0.240 | 0.213 | 0.340 | 0.477 |
| chair | 0.128 | 0.146 | 0.150 | 0.200 |
| cow | 0.140 | 0.177 | 0.228 | 0.315 |
| diningtable | 0.098 | 0.229 | 0.575 | 0.277 |
| dog | 0.162 | 0.149 | 0.251 | 0.372 |
| horse | 0.335 | 0.361 | 0.380 | 0.519 |
| motorbike | 0.375 | 0.403 | 0.437 | 0.563 |
| person | 0.221 | 0.420 | 0.415 | 0.475 |
| pottedplant | 0.120 | 0.126 | 0.132 | 0.130 |
| sheep | 0.175 | 0.194 | 0.251 | 0.378 |
| sofa | 0.147 | 0.173 | 0.280 | 0.330 |
| train | 0.334 | 0.296 | 0.463 | 0.503 |
| tvmonitor | 0.289 | 0.371 | 0.376 | 0.419 |
| # methods | 5 | 3 | 6 | 6 |
| # comp | 9 | 7 | 17 | 19 |

TABLE 17.1: Average precision of the best classification method for each category for the Pascal image classification challenge by year (per category; the method that was best at "person" might not be best at "pottedplant"), summarized from `http://pascallin.ecs.soton.ac.uk/challenges/VOC/`. On the bottom rows, the number of methods in each column and the total number of methods competing (so, for example, in 2007, only 2 of 17 total methods were best in category; each of the other 15 methods was beaten by something for each category). Notice that the average precision grows, but not necessarily monotonically (this is because the test set changes). Most categories now work moderately well.

# Speed+Accuracy

- There are methods to reject some windows early
  - using simple tests
- Key to good performance seems to be
  - very large numbers of examples
    - and clever methods to train with huge numbers of negatives
  - careful feature engineering

# Poselets - I

Detect body parts, using simple HOG feature+classifier strategy

FIGURE 18.7: Poselets are image patches of characteristic, relatively constrained appearance that suggest a restricted range of configurations. These are examples of image patches corresponding to four distinct poselets (associated with face, arms, whole body, and head) from Bourdev and Malik (2009). Notice how each could likely be found with current detectors in a relatively straightforward way. *This figure was originally published as Figure 1 of "Poselets: Body Part Detectors Trained Using 3D Human Pose Annotations," L. Bourdev and J. Malik, Proc. IEEE ICCV 2009,* © *2009 IEEE.*

# Poselets - II

Now the parts vote on where the torso is

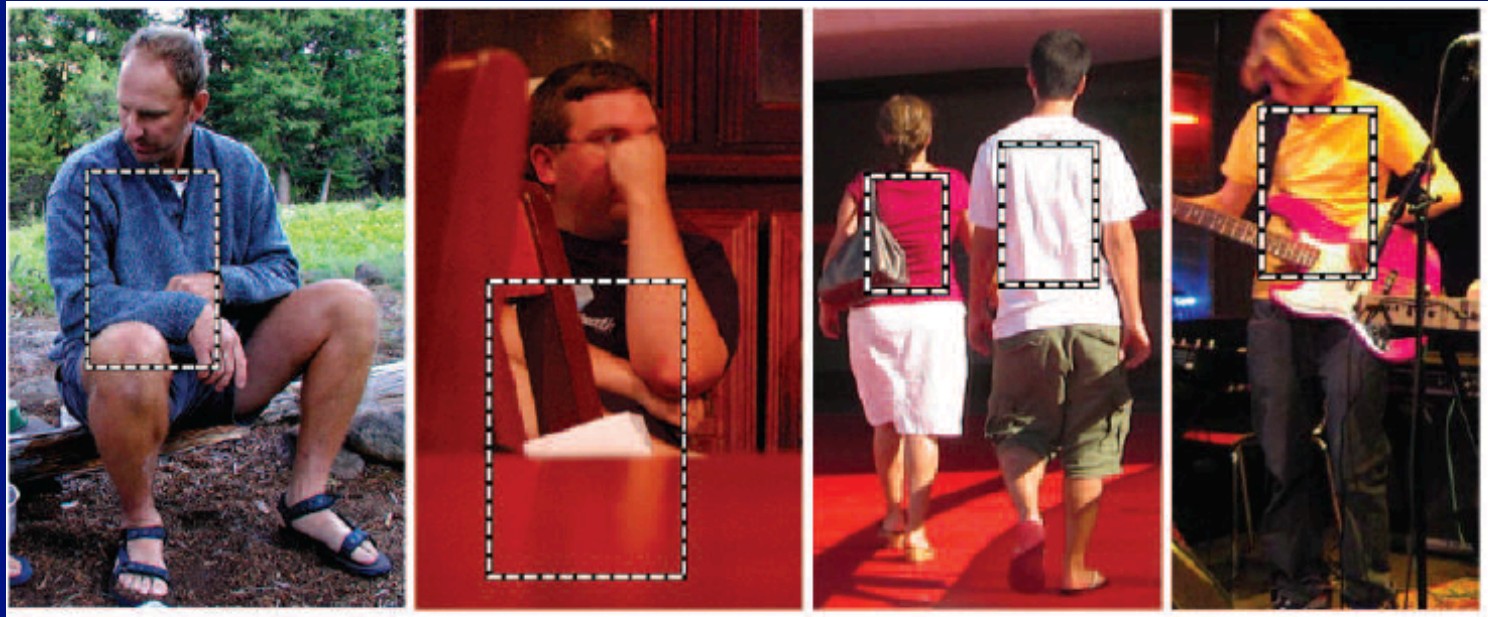


FIGURE 18.8: Bourdev and Malik (2009) show that poselets can be used to find, say, the torso of the body even though it might not be visible. Each detected poselet can cast a vote, whose value is determined discriminatively, for the location of the torso. The likely torso locations are then clustered, to identify groups of votes that agree. Finally, the strongest cluster gives a torso location, if it is strong enough. Notice how some of the marked torsos could not be identified by direct image information. *This figure was originally published as Figure 10 of "Poselets: Body Part Detectors Trained Using 3D Human Pose Annotations," L. Bourdev and J. Malik, Proc. IEEE ICCV 2009, © 2009 IEEE.*

# Visual phrases

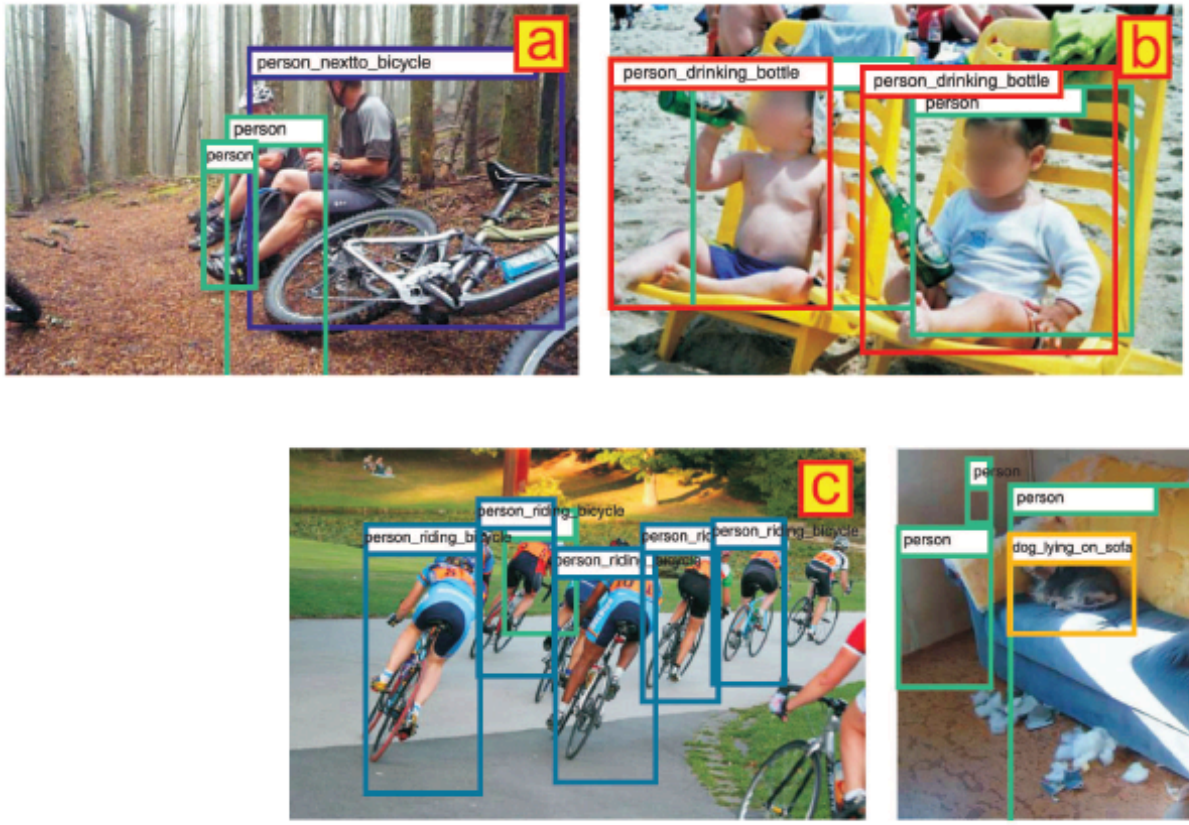



FIGURE 18.9: Visual phrases are composites of objects that are easier to detect than their components. Farhadi and Sadeghi (2011) demonstrate that some visual phrases exist and are useful. For example, it is much easier to detect a person drinking from a bottle than it is to detect a person, because a person drinking from a bottle has a more limited and more characteristic range of appearances. These figures show some examples of visual phrases, detected using the methods of Section 17.2. *This figure was originally published as Figure 1 of "Recognition using Visual Phrases," A. Farhadi and A. Sadeghi, Proc. IEEE CVPR 2011, © 2011 IEEE.*

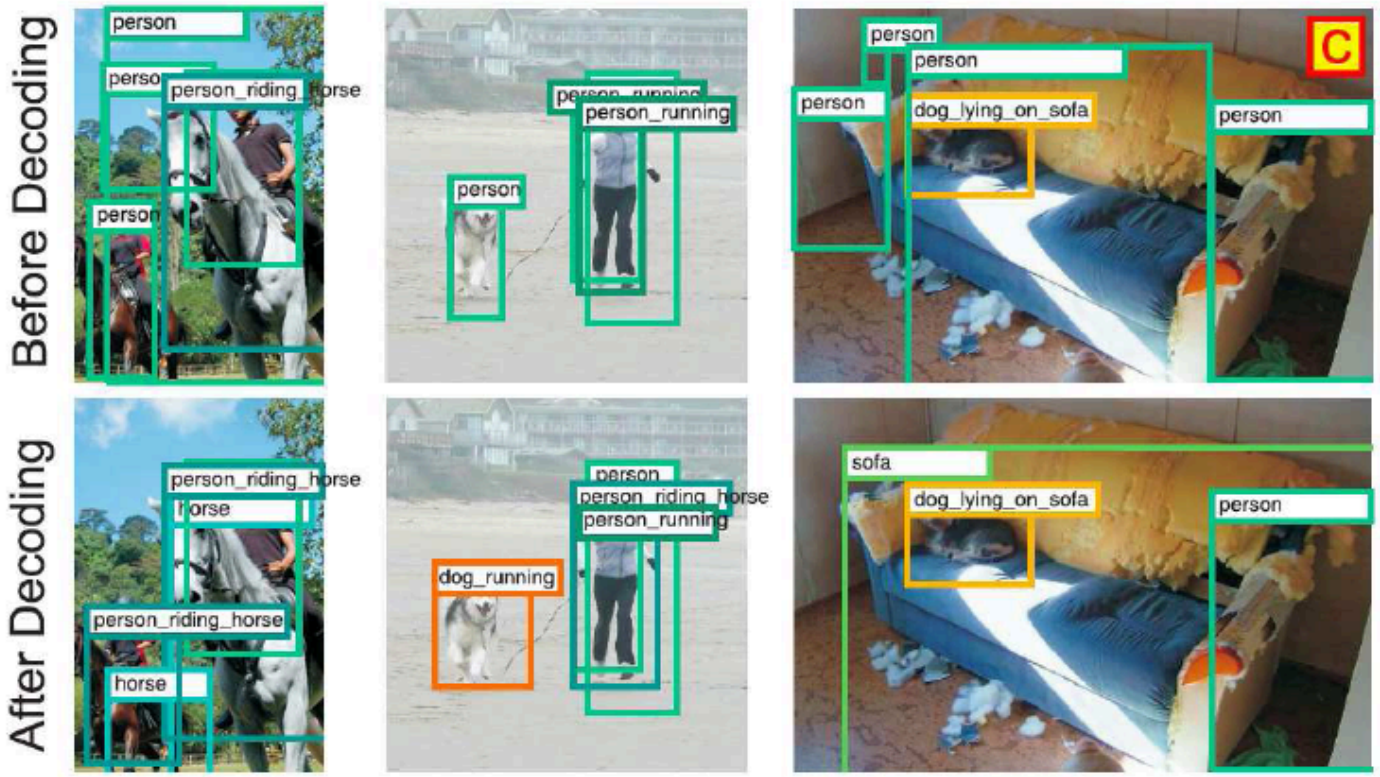



FIGURE 18.10: Detection systems that use visual phrases must be able to deal with ambiguous and possibly mutually exclusive detector responses. For example, when there is a person drinking from a bottle, there must also be a person and a bottle, and all three detectors might respond. Resolving what to report given a set of detector responses is called decoding by Farhadi and Sadeghi (2011). The **top row** shows some detector responses for each image before decoding (there are too many to show all; these are the stronger ones); the **bottom row** shows the detectors marked as correct by the decoding stage. This stage is able to use the local context of detector responses. For example, a strong response from a dog lying on sofa detector implies a sofa, and so the sofa can be believed; similarly, a believable sofa implies that many of the person detector responses are unlikely. Farhadi and Sadeghi (2011) demonstrate that decoding improves the performance of all detectors in the system, and that having visual phrase detectors and a decoding stage improves the performance of conventional object detectors, most likely by exposing contextual information that strengthens or reduces the plausibility of the detector response. *This figure was originally published as Figure 6 of "Recognition using Visual Phrases," A. Farhadi and A. Sadeghi, Proc. IEEE CVPR 2011, © 2011 IEEE.*

# Decoding

### 17.3.1 Datasets and Resources

**Pedestrian detection datasets:** There are multiple pedestrian datasets. The INRIA pedestrian dataset, used in Dalal and Triggs (2005), is published by Dalal and Triggs at `http://pascal.inrialpes.fr/data/human/`. The MIT pedestrian dataset, introduced in Papageorgiou and Poggio (2000), is published at `http://cbcl.mit.edu/software-datasets/PedestrianData.html`.

There is a set of pointers to implementations of systems and to datasets at `http://www.pedestrian-detection.com/`.

Dollár, Wojek, Schiele, and Perona publish several very large pedestrian datasets (including the Caltech training dataset, test dataset, and Japan dataset) at `http://www.vision.caltech.edu/Image_Datasets/CaltechPedestrians/`, and describe them in Dollar *et al.* (2009). This location also contains pointers to other pedestrian datasets.

Ess, Leibe, Schindler, and van Gool publish a dataset of tracked humans—who are likely pedestrians—at `http://www.vision.ee.ethz.ch/~aess/dataset/`; this dataset is described in detail in Ess *et al.* (2009).

Overett, Petersson, Brewer, Andersson, and Pettersson publish the NICTA pedestrian dataset at `http://nicta.com.au/research/projects/AutoMap/computer_vision_datasets`; this dataset is described in detail in Overett *et al.* (2008).

Wojek, Walk, and Schiele publish a dataset of pedestrians in motion at `http://www.mis.tu-darmstadt.de/tud-brussels`; this dataset is described in detail in Wojek *et al.* (2009).

There are several datasets associated with Daimler Chrysler which can be found at `http://www.gavrila.net/Research/Pedestrian_Detection/Daimler_Pedestrian_Benchmarks/daimler_pedestrian_benchmarks.html`.

Enzweiler and Gavrila publish the Daimler pedestrian benchmark dataset at this URL and it described in detail in Enzweiler and Gavrila (2009). Munder and Gavrila publish the Daimler pedestrian classification dataset at this URL and it is described in detail in Munder and Gavrila (2006). Enzweiler, Eigenstetter, Schiele, and Gavrila publish the Daimler multi-cue occluded pedestrian detection benchmark dataset at this URL and it is described in Enzweiler *et al.* (2010).

The computer vision center at the Universitat Autònoma de Barcelona publishes several pedestrian datasets at `http://www.cvc.uab.es/adas/index.php?`

section=other_datasets. There is a dataset of virtual pedestrians at this URL, published by Marín, Vázquez, Gerónimo, and López; it is described in detail in Marin *et al.* (2010). Gerónimo, Sappa, López, and Ponsa publish a dataset of pedestrians captured around Barcelona at this URL; this dataset is described in detail in Gerónimo *et al.* (2007).

Maji, Berg, and Malik publish pedestrian detector code that uses pyramid HOG features and an intersection kernel SVM at http://www.cs.berkeley.edu/~smaji/projects/ped-detector/. The code is described in Maji *et al.* (2008).

**Face detection codes and datasets:** The number of face detection datasets is so great that we provide pointers to pages that collect datasets. These pages provide pointers to codes as well. There is a collection of 12 datasets, including several well-known face datasets, at http://robotics.csie.ncku.edu.tw/Databases/FaceDetect_PoseEstimate.htm. Frischholz maintains a face detection home page, containing demonstrations, publications, datsets, and links, at http://www.facedetection.com/; many more face datasets appear in the dataset component at http://www.facedetection.com/facedetection/datasets.htm. Grgic and Delac supply codes and datasets for face recognition at http://face-rec.org/. Some sample codes, and further datasets can be found at http://vision.ai.uiuc.edu/mhyang/face-detection-survey.html.

**General object detection codes and datasets:** All datasets from the PASCAL challenge are published at http://pascallin.ecs.soton.ac.uk/challenges/VOC/, and described in detail by Everingham *et al.* (2010). Most of the strongest methods on this challenge are based on the detector we described in Section 17.2; code for training and testing this method is available at http://people.cs.uchicago.edu/~pff/latent/.

**Pb codes and data** are published by Arbelaez, Maire, Fowlkes, and Malik at http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html; there is a description in Arbelaez *et al.* (2011).

# Summary

- Simple detection strategy is quite effective
  - search windows, compute HOG, stick in classifier
- Can use somewhat more complex object models
  - makes things better
- For some objects, detection is quite effective