Object Detection with Discriminatively Trained Part-Based Models

David Forsyth, *University of Illinois at Urbana-Champaign*

This installment of *Computer*'s series highlighting the work published in IEEE Computer Society journals comes from *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

How does a computer program find an object in a picture? Doing so is usually very easy for people, who don't seem perturbed by significant effects that present challenges for programs:

- *Lighting.* When an object is in bright light, it looks brighter than when it's in shadow, so a program can't just look at image intensity values.
- *Within-class variation.* Different instances of the same kind of object can look quite different to one another. For example, a green station wagon and a red convertible are both cars, so a program can't simply compare a picture to one example.
- *Aspect.* The same object can look very different when viewed at from different directions—pick up a book and compare its cover and its spine to see this effect. Again, this means that a program might need to have many examples of each type of object.
- *Deformation.* Many objects can change their appearance significantly without their identity changing. For example, you can move your limbs around, change clothes, paint your face, or have your hair cut. You will look very different indeed, but you will still be a person.

Brown University's Pedro Felzenszwalb and his colleagues described a method to find objects by managing these effects ("Object Detection with Discriminatively Trained Part-Based Models," vol. 32, no. 9, 2010, pp. 1627–1645). Specifically, they built an object detector using the "sliding window" method. To search for, say, cars, the user would first build a classifier, which is a decision rule, learned from data, that can decide whether an image window contains a car or not. The user then takes a subwindow of the image, describes it with features that are invariant to lighting effects, and then presents it to the classifier, which decides whether a car is present or not before moving on to the next subwindow. To find bigger (or smaller) cars, the method looks at subwindows of the same size in a smaller (or bigger) version of the image. To find different types of the same object, the user builds different classifiers, one for each type.

Both the features and search outline were well known by the time this paper appeared, but the key to the authors' success was in the structure of the classifier. As with earlier methods, it scores the whole subwindow (so testing for the overall boxy shape of a car). But it also scores distinct smaller patterns ("parts"—in our example, likely wheels and a windshield), with values that summarize whether the parts look as they should and are near where they should be. So the method accommodates the difference between cars with long and short wheelbases by a score that allows wheel parts to move freely in the horizontal direction and between side views and frontal views; Figure 1 shows this for horses. Both the scores for part appearance and

Published by the IEEE Computer Society

location are learned from data using a novel strategy.

At the time the paper appeared, the authors published the code both for learning and applying these models under very generous licensing terms; since then, they've updated the code base several times. There are many interesting natural variants of this method. You could, for example, use different features or numbers or types of parts; score part locations differently; speed up the computation of the scores using various approximations; apply the method to depth data; and so on. This is an excellent first paper to read in computer vision, because it's accessible and "makes sense." It's also the cornerstone upon which object detection is now based. ■

David Forsyth *is a professor of computer science at the University of Illinois at Urbana-Champaign.*

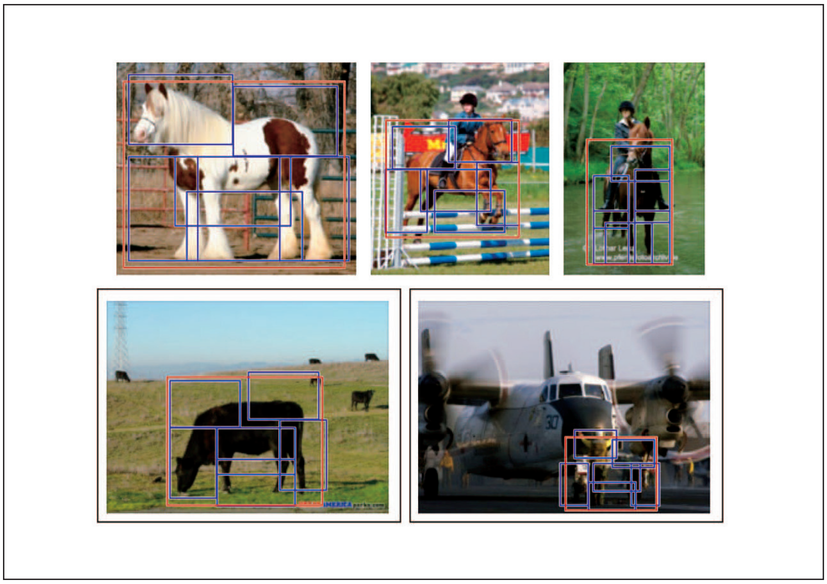

Figure 1. This excerpt from Figure 10 of the original paper shows responses from a horse detector. The red boxes are image windows where the method has detected a horse; the blue boxes are the part subwindows. The top row shows successful examples. Notice how horses look different when viewed from the side, at an angle, and from the front. The detector accommodates these changes by allowing the part locations to move. Object detection is difficult: the bottom row shows false detections. Cows look like horses, and so does just the right view of an aircraft's undercarriage.