Object Detection

Yali Amit and Pedro Felzenszwalb, University of Chicago

Related Concepts

- Object Recognition
- Image Classification

Definition

Object detection involves detecting instances of objects from a particular class in an image.

Background

The goal of object detection is to detect all instances of objects from a known class, such as people, cars or faces in an image. Typically only a small number of instances of the object are present in the image, but there is a very large number of possible locations and scales at which they can occur and that need to somehow be explored.

Each detection is reported with some form of *pose* information. This could be as simple as the location of the object, a location and scale, or the extent of the object defined in terms of a bounding box. In other situations the pose information is more detailed and contains the parameters of a linear or non-linear transformation. For example a face detector may compute the locations of the eyes, nose and mouth, in addition to the bounding box of the face. An example of a bicycle detection that specifies the locations of certain parts is shown in Figure 1. The pose could also be defined by a three-dimensional transformation specifying the location of the object relative to the camera.

Object detection systems construct a model for an object class from a set of training examples. In the case of a fixed rigid object only one example may be needed, but more generally multiple training examples are necessary to capture certain aspects of class variability.

Object detection methods fall into two major categories, *generative* [1,2,3,4,5] and *discriminative* [6,7,8,9,10]. The first consists of a probability model for the pose variability of the objects together with an appearance model: a probability model for the image appearance conditional on a given pose, together with a model for background, i.e. non-object images. The model parameters can be estimated from training data and the decisions are based on ratios of posterior probabilities. The second typically builds a classifier that can discriminate between images (or sub-images) containing the object and those not containing the object. The parameters of the classifier are selected to minimize mistakes on the training data, often with a regularization bias to avoid overfitting.

Other distinctions among detection algorithms have to do with the *computational* tools used to scan the entire image or search over possible poses, the type

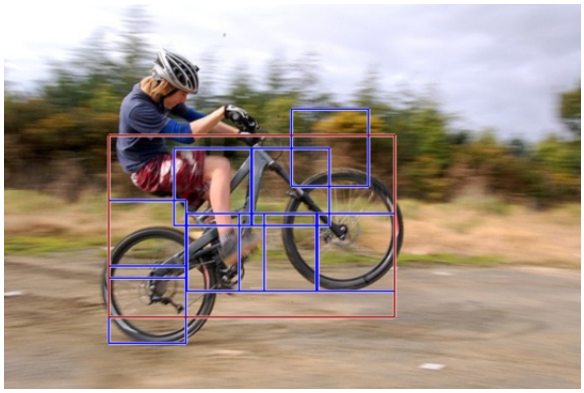

Fig. 1. A bicycle detection specified in terms of the locations of certain parts.

of image *representation* with which the models are constructed, and what type and how much training data is required to build a model.

Theory

Images of objects from a particular class are highly variable. One source of variation is the actual imaging process. Changes in illumination, changes in camera position as well as digitization artifacts, all produce significant variations in image appearance, even in a static scene. The second source of variation is due to the intrinsic appearance variability of objects within a class, even assuming no variation in the imaging process. For example, people have different shapes and wear a variety of clothes, while the handwritten digit 7 can be written with or without a line through the middle, with different slants, stroke widths, etc. The challenge is to develop detection algorithms that are *invariant* with respect to these variations and are computationally efficient.

Invariance

The brute force (naive) approach to invariance assumes training data is plentiful and represents the entire range of object variability. Invariance is implicitly learned from the data while training the models.

When training data is limited it is necessary to build invariance into the models. There are two complementary methods to achieve this. One involves computing invariant functions and features, the other involves searching over latent variables. Most algorithms contain a combination of these approaches. For example many algorithms choose to apply local transformations to pixel intensities in such a way that the transformed values are invariant to a range of illumination conditions and small geometric variations. These local transformations lead to *features* and the array of feature values is the *feature map*. More significant transformations are often handled through explicit search of latent variables or by learning the remaning variability from training data.

Fig. 2. The output of a face detection algorithm.

Invariant functions and features This method constructs functions of the data that are invariant with respect to the types of variability described above and can still distinguish between object and background images. This may prove difficult if object variability is extensive. Invariant functions that produce the same output no matter the pose and appearance of the object necessarily have less discriminative power.

There are two common types of operations leading to invariant functions. The first involves computing local features that are invariant to certain image transformations. The second operation involves computing geometric quantities that are invariant to some or all three-dimensional pose variation. For example the cross-ratio among distinguished points is a projective invariant that has been used to recognize rigid objects.

An example of a local feature, invariant to certain photometric variations and changes in illumination, is the *direction* of the image gradient, from which a variety of *edge* features can be computed. More complex features capture the appearance of small image patches and are often computed from edge features. An example would be the histogram of gradient (HOG) features [9]. These features are usually computed at a *dense* grid of locations in the image, leading to a dense feature map.

Local pooling of features is commonly used to introduce some degree of invariance to small geometric variations. A typical example is the MAX operation [11,1]. In this case a quantity that is to be computed at a pixel is replaced by the maximum of the quantity in a neighborhood of the pixel. When the maximum is extended over the entire window the result is a *bag of features* model [12], which reduces to counting the number of binary features of different types that occur

within a window. In this case all spatial information is lost, leading to models that are invariant to fairly large geometric transformations.

For computational reasons it is often useful to sparsify the feature map by applying local decisions to find a small set of *interest points*. The assumption is that only certain features are useful (or necessary) for object detection. The approach yields *sparse* feature maps that can be processed much more efficiently. Examples of commonly used sparse features are SIFT descriptors [13], corner detectors and edge conjunctions [1]. One drawback of sparse features is that hard decisions are being made on their presence, and if some are missed an algorithm may fail to detect an instance of the object.

Note that it is possible to predefine a very large family of features that is never fully computed, rather, in training an informative subset is selected that can produce the required classification for a particular object class. One example are the Haar features that compute differences of intensity averages in adjacent rectangles of varying sizes and locations [7]. Another example are geometric edge arrangments of increasing complexity.

Latent variables An explicit parameterization of the variability can be defined via *latent* variables that are not directly observable from the image data. These are not necessarily needed for the final report on the object detections, but their values simplify the solution of the detection problem. For example to detect faces at a range of orientations, at each candidate region one could decide, for *each* possible orientation, whether or not the region contains a face at that orientation. In general a set Θ defines latent parameters that could capture global illumination parameters, a linear or non-linear map from a model domain into the image domain, or specify the locations of a finite set of object parts. The last case is common in part-based models where latent part placements are used to decide if the object might be present at a particular location in the image [10]. The set of possible latent values, Θ, can be quite large or infinite. This leads to computational challenges that have been addressed by a variety of methods including coarse-to-fine computation, dynamic programming and geometric alignment.

Detection via Classification

Both generative and discriminative models start with an initial choice of image features and with a choice of the latent pose parameters that will be explicitly modeled. The primary differences between generative and discriminative models are in the methods of training and computation. One important distinction is that generative models do not need data from background to train the object model whereas discriminative methods need data from both classes to learn the decision boundaries.

The most common approach to object detection reduces the problem to one of binary classification. Consider the problem of detecting objects of fixed size but varying positions in the image. Let W denote a reference window size that an instance of the object would occupy. Let L denote a grid of locations in

the image. Let X_{s+W} denote the image features in a window (sub-image) with top-left corner at $s \in L$. One can reduce the detection problem to a binary classification problem as follows. For each location $s \in L$ classify X_{s+W} into two possible classes corresponding to windows that contain an object and windows that do not contain an object. The sliding-window approach to object detection involves explicitly considering and classifying every possible window. Note that the same approach can be used to detect objects of different sizes by considering different window sizes or alternatively windows of fixed size at different levels of resolutions in an image pyramid.

Generative Models

A general framework for object detection using generative models involves modeling two distributions. A distribution $p(\theta; \eta_p)$ is defined on the possible latent pose parameters $\theta \in \Theta$. This distribution captures assumptions on which poses are more or less likely. An appearance model is defined describing the distribution of the image features in a window *conditional* on the pose, $p(X_{s+W}|object, \theta; \eta_a)$. Here η_p and η_a are the model parameters. For example, η_a might define a *template* specifying the probability of observing certain features at each location in the detection window under a canonical choice for the object pose, while θ specifies a transformation of the template. Warping the template according to θ leads to probabilities for observing certain features at each location in X_{s+W} conditioned on this particular choice of pose parameters [1,2,4].

Training data with images of the object are used to estimate the parameters η_p and η_a. Note that the images do not normally come with information about the latent pose variables θ, unless annotation is provided. Estimation thus requires inference methods that handle unobserved variables, for example the different variants of the expectation maximization algorithm [4,3]. In some cases a probability model for background images is estimated as well using large numbers of training examples of images not containing the object.

The basic detection algorithm then scans each candidate window in the image, computes the most likely pose under the object model and obtains the 'posterior odds', i.e. the ratio between the conditional probability of the window under the object hypothesis at the optimal pose, and the conditional probability of the window under the background hypothesis. This ratio is then compared to a threshold τ to decide if the window contains an instance of the object

$$\frac{p(X_{s+W}|object, \theta; \eta_a)p(\theta; \eta_p)}{p(X_{s+W}|background)} > \tau.$$

When no background model has been trained offline, a simple *adaptive* background model can be estimated online for each window being tested. In this case no background training data is needed [4]. Alternative background models involve sub-collections of parts of the object model [14].

Discriminative Models

If no explicit latent pose variables are used the underlying assumption is that the training data is sufficiently rich to provide a sample of the entire variation of object appearance. The discriminative approach trains a standard two class classifier using large amounts of data from the object and background classes. Many classifier types have been used, including neural networks, SVMs, boosted decision trees and radial basis functions.

Cascades Because of the large size of the background population and its complexity discriminative methods are often organized in *cascades* [7]. An initial classifier is trained to distinguish between the object and a manageable amount of background data. The classifier is designed to have no false negatives at the price of a larger number of false positives. Then a large number of background examples are evaluated and the misclassified ones are collected to form a new background data set. Once a sufficient number of such false positives is accumulated a new classifier is trained to discriminate between the original object data and the new 'harder' background data. Again this classifier is designed to have no false negatives. This process can be continued several times.

At detection time the classifiers in the cascade are applied sequentially. Once a window is classified as background the testing terminates with the background label. If the object label is chosen, the next classifier in the cascade is applied. Only windows that are classified as object by all classifiers in the cascade are labelled as object by the cascade. Note that having no false negatives during training training provides no guarantee that in testing instances of the objects won't be missed.

Pose variables Certain discriminative models can also be implemented with latent pose parameters [10]. Assume a generic classifier defined in terms of a space of classifier functions $f(x;u)$ parameterized by u. Usual training of a discriminative model consists of solving an equation of the form

$$\min_u \sum_{i=1}^{n} D(y_i, f(x_i;u)) + C(u),$$

for some regularization term $C(u)$ which prevents overfitting and a loss function D measuring the distance between the classifier output $f(x_i;u)$ and the ground truth label $y_i = 1$ for object and $y_i = 0$ for background.

The minimization above can be replaced by

$$\min_u \sum_{y_i=1} \min_{\theta\in\Theta} D(1, f(\theta(x_i);u)) + \sum_{y_i=0} \max_{\theta\in\Theta} D(0, f(\theta(x_i);u)) + C(u).$$

Here $\theta(x)$ defines a transformation of the example x. Intuitively for a positive example one would like there to be some transformation under which x_i is classified as object, while for a negative example one would like it to be the case that there is no transformation under which x_i is classified as object.

Computational methods

The basic detection process consist of scanning the image lattice and at each location s testing whether X_{s+W} is classified as object or background. This is typically done at multiple resoltions of the image pyramid to detect objects at multiple scales, and is clearly a very intesive computation. There are a number of methods to make it more efficient.

Sparse features When sparse features are used it is possible to focus the computation only in regions around features. The two main approaches that take advantage of this sparsity are *alignment* [15] and the *generalized Hough transform*. Alignment uses information regarding the relative locations of the features on the object. In this case the locations of some features determine the possible locations of the other features. Various search methods enable a quick decision on whether a sufficient number of features was found to declare object, or not. The Hough transform typically uses information on the location of each feature type relative to some reference point in the object. Each detected feature votes with some weight for a set of candidate locations of the reference point. Locations with a sufficiently large sum of weighted votes determine detections. This idea can also be generalized to include identification of scale as well. The voting weights can be obtained either through discriminative training or through generative training [1].

Cascades As mentioned above the cascade method trains a sequence of classifiers with successively more difficult background data. Each such classifier is designed to be very computationally efficient. When the data in the window X_{s+W} is declared background by any classifier of the cascade the decision is final and the computation proceeds to the next window. Since most background windows are rejected early in the cascade most of the windows in the image are processed very quickly.

Coarse to fine The cascade method can be viewed as a coarse to fine decomposition of background that gradually makes finer and finer discriminations between object and background images that have significant resemblance to the object. An alternative is to create a coarse to fine decomposition of object poses [8]. In this case it is possible totrain classifiers that can rule out a large subset of the pose space in a single step. A general setting involves a rooted tree where the leaves correspond to individual detections and internal nodes store classifiers that quickly rule out all detections below a particular node. The idea is closely related to branch-and-bound methods [12] that use admissible lower-bounds to search a space of transformations or hypotheses.

Dynamic programming There are a number of object detection algorithms that represent objects by a collection of parts arranged in deformable configurations or as hierarchies of such arrangements of parts of increasing complexity. When the hierarchies and the arrangements have the appropriate structure dynamic

programming methods can be used to efficiently search over the spaces of arrangements [1], [2].

Application

Object detection methods have a wide range of applications in a variety of areas including robotics, medical image analysis, surveillance and human computer interaction. Current methods work reasonably well in constrained domains but are quite sensitive to clutter and occlusion.

A popular benchmark for object detection is the PASCAL VOC object detection challenge. The goal of the challenge is to detect objects from common categories such as people, cars, horses and tables in photographs. The challenge has attracted significant attention in the computer vision community over the last few years and the performance of the best systems have been steadly increasing by a significant amount on a yearly basis.

Face detection is a typical application of object detection algorithms. There has been significant success in deploying face detection methods in practical situations. For example current digital cameras use face detection to decide where to focus and even detect smiles to decide when to take the picture. Figure 2 shows a typical output of a face detection algorithm.

Recommended Readings

[1] Amit, Y. (2002). 2d Object Detection and Recognition: Models, Algorithms and Networks. MIT Press, Cambridge, MA.

[2] Felzenszwalb, P., Huttenlocher, D. (2005). Pictorial structures for object recognition. International Journal of Computer Vision **61**(1) 55–79

[3] Fergus, R., Perona, P., Zisserman, A. (2003). Object class recognition by unsupervised scale-invariant learning. IEEE CVPR 2003

[4] Amit, Y., Trouvé, A. (2007). POP: Patchwork of parts models for object recognition. International Journal of Computer Vision **75**(2) 267–282

[5] Jin, Y., Geman, S. (2006). Context and hierarchy in a probabilistic image model. IEEE CVPR 2006

[6] Rowley, H.A., Baluja, S., Kanade, T. (1998). Neural network-based face detection. IEEE Transactions on Pattern Analysis and Machine Intelligence **20**(1) 23–38

[7] Viola, P., Jones, M.J. (2004). Robust real time face detection. International Journal of Computer Vision **57**(2) 137–154

[8] Fleuret, F., Geman, D. (2001). Coarse-to-fine face detection. International Journal of Computer Vision **41**(1-2) 85–107

[9] Dalal, N., Triggs, B. (2005). Histograms of oriented gradients for human detection. IEEE CVPR 2005

[10] Felzenszwalb, P., Girshick, R., McAllester, D., Ramanan, D. (2010). Object detection with discriminatively trained part based models. IEEE Transactions on Pattern Analysis and Machine Intelligence **32**(9) 1627–1645

[11] Riesenhuber, M., Poggio, T. (2000). Models of object recognition. Nature Neuroscience **3** 1199–1204 Supplement.

[12] Lampert, C., Blaschko, M., Hofmann, T. (2009). Efficient subwindow search: A branch and bound framework for object localization. IEEE Transactions on Pattern Analysis and Machine Intelligence **31**(12) 2129–2142
[13] Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision **60**(2) 91–110
[14] Chang, L.B., Jin, Y., Zhang, W., Borenstein, E., Geman, S. (2011). Context computation, and optimal roc performance in hierarchical models. International Journal of Computer Vision
[15] Ullman, S. (1996). High-Level Vision. MIT. Press, Cambridge, MA.