Rapid Object Detection using a Boosted Cascade of Simple Features

Paul Viola
viola@merl.com
Mitsubishi Electric Research Labs
201 Broadway, 8th FL
Cambridge, MA 02139

Michael Jones
michael.jones@compaq.com
Compaq Cambridge Research Lab
One Cambridge Center
Cambridge, MA 02142

Abstract

This paper describes a machine learning approach for visual object detection which is capable of processing images extremely rapidly and achieving high detection rates. This work is distinguished by three key contributions. The first is the introduction of a new image representation called the "Integral Image" which allows the features used by our detector to be computed very quickly. The second is a learning algorithm, based on AdaBoost, which selects a small number of critical visual features from a larger set and yields extremely efficient classifiers[5]. The third contribution is a method for combining increasingly more complex classifiers in a "cascade" which allows background regions of the image to be quickly discarded while spending more computation on promising object-like regions. The cascade can be viewed as an object specific focus-of-attention mechanism which unlike previous approaches provides statistical guarantees that discarded regions are unlikely to contain the object of interest. In the domain of face detection the system yields detection rates comparable to the best previous systems. Used in real-time applications, the detector runs at 15 frames per second without resorting to image differencing or skin color detection.

1. Introduction

This paper brings together new algorithms and insights to construct a framework for robust and extremely rapid object detection. This framework is demonstrated on, and in part motivated by, the task of face detection. Toward this end we have constructed a frontal face detection system which achieves detection and false positive rates which are equivalent to the best published results [14, 11, 13, 10, 1]. This face detection system is most clearly distinguished from previous approaches in its ability to detect faces extremely rapidly. Operating on 384 by 288 pixel images, faces are detected at 15 frames per second on a conventional 700 MHz Intel Pentium III. In other face detection systems, auxiliary information, such as image differences in video sequences, or pixel color in color images, have been used to achieve high frame rates. Our system achieves high frame rates working only with the information present in a single grey scale image. These alternative sources of information can also be integrated with our system to achieve even higher frame rates.

There are three main contributions of our object detection framework. We will introduce each of these ideas briefly below and then describe them in detail in subsequent sections.

The first contribution of this paper is a new image representation called an *integral image* that allows for very fast feature evaluation. Motivated in part by the work of Papageorgiou et al. our detection system does not work directly with image intensities [9]. Like these authors we use a set of features which are reminiscent of Haar Basis functions (though we will also use related filters which are more complex than Haar filters). In order to compute these features very rapidly at many scales we introduce the integral image representation for images. The integral image can be computed from an image using a few operations per pixel. Once computed, any one of these Harr-like features can be computed at any scale or location in *constant* time.

The second contribution of this paper is a method for constructing a classifier by selecting a small number of important features using AdaBoost [5]. Within any image subwindow the total number of Harr-like features is very large, far larger than the number of pixels. In order to ensure fast classification, the learning process must exclude a large majority of the available features, and focus on a small set of critical features. Motivated by the work of Tieu and Viola, feature selection is achieved through a simple modification of the AdaBoost procedure: the weak learner is constrained so that each weak classifier returned can depend on only a single feature [15]. As a result each stage of the boosting process, which selects a new weak classifier, can be viewed as a feature selection process. AdaBoost provides an effective learning algorithm and strong bounds on generalization performance [12, 8, 9].

The third major contribution of this paper is a method for combining successively more complex classifiers in a cascade structure which dramatically increases the speed of

0-7695-1272-0/01 $10.00 © 2001 IEEE

the detector by focusing attention on promising regions of the image. The notion behind focus of attention approaches is that it is often possible to rapidly determine where in an image an object might occur [16, 7, 1]. More complex processing is reserved only for these promising regions. The key measure of such an approach is the "false negative" rate of the attentional process. It must be the case that all, or almost all, object instances are selected by the attentional filter.

We will describe a process for training an extremely simple and efficient classifier which can be used as a "supervised" focus of attention operator. The term supervised refers to the fact that the attentional operator is trained to detect examples of a particular class. In the domain of face detection it is possible to achieve fewer than 1% false negatives and 40% false positives using a classifier constructed from two Harr-like features. The effect of this filter is to reduce by over one half the number of locations where the final detector must be evaluated.

Those sub-windows which are not rejected by the initial classifier are processed by a sequence of classifiers, each slightly more complex than the last. If any classifier rejects the sub-window, no further processing is performed. The structure of the cascaded detection process is essentially that of a degenerate decision tree, and as such is related to the work of Geman and colleagues [1, 3].

An extremely fast face detector will have broad practical applications. These include user interfaces, image databases, and teleconferencing. In applications where rapid frame-rates are not necessary, our system will allow for significant additional post-processing and analysis. In addition our system can be implemented on a wide range of small low power devices, including hand-helds and embedded processors. In our lab we have implemented this face detector on the Compaq iPaq handheld and have achieved detection at two frames per second (this device has a low power 200 MIPS *Strong Arm* processor which lacks floating point hardware).

The remainder of the paper describes our contributions and a number of experimental results, including a detailed description of our experimental methodology. Discussion of closely related work takes place at the end of each section.

2. Features

Our object detection procedure classifies images based on the value of simple features. There are many motivations for using features rather than the pixels directly. The most common reason is that features can act to encode ad-hoc domain knowledge that is difficult to learn using a finite quantity of training data. For this system there is also a second critical motivation for features: the feature based system operates much faster than a pixel-based system.

The simple features used are reminiscent of Haar basis functions which have been used by Papageorgiou et al. [9].

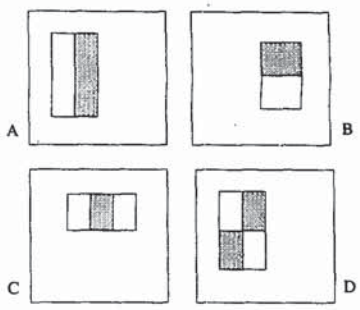

Figure 1: Example rectangle features shown relative to the enclosing detection window. The sum of the pixels which lie within the white rectangles are subtracted from the sum of pixels in the grey rectangles. Two-rectangle features are shown in (A) and (B). Figure (C) shows a three-rectangle feature, and (D) a four-rectangle feature.

More specifically, we use three kinds of features. The value of a *two-rectangle feature* is the difference between the sum of the pixels within two rectangular regions. The regions have the same size and shape and are horizontally or vertically adjacent (see Figure 1). A *three-rectangle feature* computes the sum within two outside rectangles subtracted from the sum in a center rectangle. Finally a *four-rectangle feature* computes the difference between diagonal pairs of rectangles.

Given that the base resolution of the detector is 24x24, the exhaustive set of rectangle features is quite large, over 180,000 . Note that unlike the Haar basis, the set of rectangle features is overcomplete[1].

2.1. Integral Image

Rectangle features can be computed very rapidly using an intermediate representation for the image which we call the integral image.[2] The integral image at location x, y contains the sum of the pixels above and to the left of x, y, inclusive:

$$ii(x,y) = \sum_{x' \leq x, y' \leq y} i(x', y'),$$

where $ii(x,y)$ is the integral image and $i(x,y)$ is the original image. Using the following pair of recurrences:

$$s(x,y) = s(x,y-1) + i(x,y) \quad (1)$$
$$ii(x,y) = ii(x-1,y) + s(x,y) \quad (2)$$

(where $s(x,y)$ is the cumulative row sum, $s(x,-1) = 0$, and $ii(-1,y) = 0$) the integral image can be computed in one pass over the original image.

[1] A complete basis has no linear dependence between basis elements and has the same number of elements as the image space, in this case 576. The full set of 180,000 thousand features is many times over-complete.

[2] There is a close relation to "summed area tables" as used in graphics [2]. We choose a different name here in order to emphasize its use for the analysis of images, rather than for texture mapping.

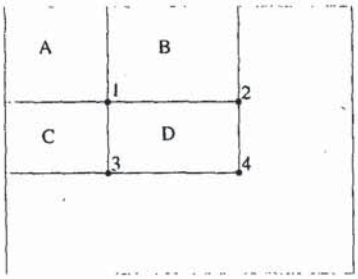

Figure 2: The sum of the pixels within rectangle D can be computed with four array references. The value of the integral image at location 1 is the sum of the pixels in rectangle A. The value at location 2 is $A+B$, at location 3 is $A+C$, and at location 4 is $A+B+C+D$. The sum within D can be computed as $4+1-(2+3)$.

Using the integral image any rectangular sum can be computed in four array references (see Figure 2). Clearly the difference between two rectangular sums can be computed in eight references. Since the two-rectangle features defined above involve adjacent rectangular sums they can be computed in six array references, eight in the case of the three-rectangle features, and nine for four-rectangle features.

2.2. Feature Discussion

Rectangle features are somewhat primitive when compared with alternatives such as steerable filters [4, 6]. Steerable filters, and their relatives, are excellent for the detailed analysis of boundaries, image compression, and texture analysis. In contrast rectangle features, while sensitive to the presence of edges, bars, and other simple image structure, are quite coarse. Unlike steerable filters the only orientations available are vertical, horizontal, and diagonal. The set of rectangle features do however provide a rich image representation which supports effective learning. In conjunction with the integral image , the efficiency of the rectangle feature set provides ample compensation for their limited flexibility.

3. Learning Classification Functions

Given a feature set and a training set of positive and negative images, any number of machine learning approaches could be used to learn a classification function. In our system a variant of AdaBoost is used *both* to select a small set of features *and* train the classifier [5]. In its original form, the AdaBoost learning algorithm is used to boost the classification performance of a simple (sometimes called weak) learning algorithm. There are a number of formal guarantees provided by the AdaBoost learning procedure. Freund and Schapire proved that the training error of the strong classifier approaches zero exponentially in the number of rounds. More importantly a number of results were later proved about generalization performance [12]. The key insight is that generalization performance is related to the margin of the examples, and that AdaBoost achieves large margins rapidly.

Recall that there are over 180,000 rectangle features associated with each image sub-window, a number far larger than the number of pixels. Even though each feature can be computed very efficiently, computing the complete set is prohibitively expensive. Our hypothesis, which is borne out by experiment, is that a very small number of these features can be combined to form an effective classifier. The main challenge is to find these features.

In support of this goal, the weak learning algorithm is designed to select the single rectangle feature which best separates the positive and negative examples (this is similar to the approach of [15] in the domain of image database retrieval). For each feature, the weak learner determines the optimal threshold classification function, such that the minimum number of examples are misclassified. A weak classifier $h_j(x)$ thus consists of a feature f_j, a threshold θ_j and a polarity p_j indicating the direction of the inequality sign:

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases}$$

Here x is a 24x24 pixel sub-window of an image. See Figure 3 for a summary of the boosting process.

In practice no single feature can perform the classification task with low error. Features which are selected in early rounds of the boosting process had error rates between 0.1 and 0.3. Features selected in later rounds, as the task becomes more difficult, yield error rates between 0.4 and 0.5.

3.1. Learning Discussion

Many general feature selection procedures have been proposed (see chapter 8 of [17] for a review). Our final application demanded a very aggressive approach which would discard the vast majority of features. For a similar recognition problem Papageorgiou et al. proposed a scheme for feature selection based on feature variance [9]. They demonstrated good results selecting 37 features out of a total 1734 features.

Roth et al. propose a feature selection process based on the Winnow exponential perceptron learning rule [10]. The Winnow learning process converges to a solution where many of these weights are zero. Nevertheless a very large number of features are retained (perhaps a few hundred or thousand).

3.2. Learning Results

While details on the training and performance of the final system are presented in Section 5, several simple results merit discussion. Initial experiments demonstrated that a

- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For $t = 1, \ldots, T$:
  1. Normalize the weights,
  $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
  so that w_t is a probability distribution.
  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i |h_j(x_i) - y_i|$.
  3. Choose the classifier, h_t, with the lowest error ϵ_t.
  4. Update the weights:
  $$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$
  where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- The final strong classifier is:
$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$
where $\alpha_t = \log \frac{1}{\beta_t}$

Figure 3: The AdaBoost algorithm for classifier learning. Each round of boosting selects one feature from the 180,000 potential features.

frontal face classifier constructed from 200 features yields a detection rate of 95% with a false positive rate of 1 in 14084. These results are compelling, but not sufficient for many real-world tasks. In terms of computation, this classifier is probably faster than any other published system, requiring 0.7 seconds to scan an 384 by 288 pixel image. Unfortunately, the most straightforward technique for improving detection performance, adding features to the classifier, directly increases computation time.

For the task of face detection, the initial rectangle features selected by AdaBoost are meaningful and easily interpreted. The first feature selected seems to focus on the property that the region of the eyes is often darker than the region of the nose and cheeks (see Figure 4). This feature is relatively large in comparison with the detection sub-window, and should be somewhat insensitive to size and location of the face. The second feature selected relies on the property that the eyes are darker than the bridge of the nose.

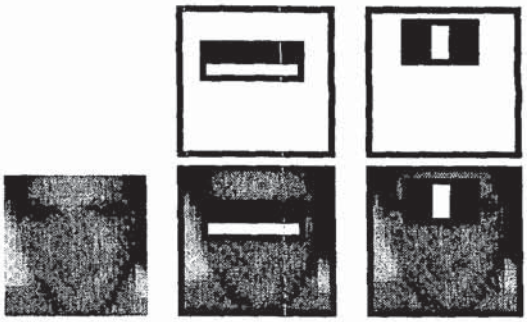

Figure 4: The first and second features selected by AdaBoost. The two features are shown in the top row and then overlayed on a typical training face in the bottom row. The first feature measures the difference in intensity between the region of the eyes and a region across the upper cheeks. The feature capitalizes on the observation that the eye region is often darker than the cheeks. The second feature compares the intensities in the eye regions to the intensity across the bridge of the nose.

4. The Attentional Cascade

This section describes an algorithm for constructing a cascade of classifiers which achieves increased detection performance while radically reducing computation time. The key insight is that smaller, and therefore more efficient, boosted classifiers can be constructed which reject many of the negative sub-windows while detecting almost all positive instances (i.e. the threshold of a boosted classifier can be adjusted so that the false negative rate is close to zero). Simpler classifiers are used to reject the majority of sub-windows before more complex classifiers are called upon to achieve low false positive rates.

The overall form of the detection process is that of a degenerate decision tree, what we call a "cascade" (see Figure 5). A positive result from the first classifier triggers the evaluation of a second classifier which has also been adjusted to achieve very high detection rates. A positive result from the second classifier triggers a third classifier, and so on. A negative outcome at any point leads to the immediate rejection of the sub-window.

Stages in the cascade are constructed by training classifiers using AdaBoost and then adjusting the threshold to minimize false negatives. Note that the default AdaBoost threshold is designed to yield a low error rate on the training data. In general a lower threshold yields higher detection rates and higher false positive rates.

For example an excellent first stage classifier can be constructed from a two-feature strong classifier by reducing the threshold to minimize false negatives. Measured against a validation training set, the threshold can be adjusted to detect 100% of the faces with a false positive rate of 40%. See

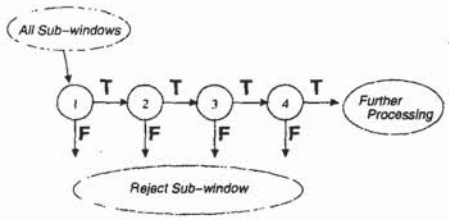

Figure 5: Schematic depiction of a the detection cascade. A series of classifiers are applied to every sub-window. The initial classifier eliminates a large number of negative examples with very little processing. Subsequent layers eliminate additional negatives but require additional computation. After several stages of processing the number of sub-windows have been reduced radically. Further processing can take any form such as additional stages of the cascade (as in our detection system) or an alternative detection system.

Figure 4 for a description of the two features used in this classifier.

Computation of the two feature classifier amounts to about 60 microprocessor instructions. It seems hard to imagine that any simpler filter could achieve higher rejection rates. By comparison, scanning a simple image template, or a single layer perceptron, would require at least 20 times as many operations per sub-window.

The structure of the cascade reflects the fact that within any single image an overwhelming majority of sub-windows are negative. As such, the cascade attempts to reject as many negatives as possible at the earliest stage possible. While a positive instance will trigger the evaluation of every classifier in the cascade, this is an exceedingly rare event.

Much like a decision tree, subsequent classifiers are trained using those examples which pass through all the previous stages. As a result, the second classifier faces a more difficult task than the first. The examples which make it through the first stage are "harder" than typical examples. The more difficult examples faced by deeper classifiers push the entire receiver operating characteristic (ROC) curve downward. At a given detection rate, deeper classifiers have correspondingly higher false positive rates.

4.1. Training a Cascade of Classifiers

The cascade training process involves two types of tradeoffs. In most cases classifiers with more features will achieve higher detection rates and lower false positive rates. At the same time classifiers with more features require more time to compute. In principle one could define an optimization framework in which: i) the number of classifier stages, ii) the number of features in each stage, and iii) the threshold of each stage, are traded off in order to minimize the expected number of evaluated features. Unfortunately finding this optimum is a tremendously difficult problem.

In practice a very simple framework is used to produce an effective classifier which is highly efficient. Each stage in the cascade reduces the false positive rate and decreases the detection rate. A target is selected for the minimum reduction in false positives and the maximum decrease in detection. Each stage is trained by adding features until the target detection and false positives rates are met (these rates are determined by testing the detector on a validation set). Stages are added until the overall target for false positive and detection rate is met.

4.2. Detector Cascade Discussion

The complete face detection cascade has 38 stages with over 6000 features. Nevertheless the cascade structure results in fast average detection times. On a difficult dataset, containing 507 faces and 75 million sub-windows, faces are detected using an average of 10 feature evaluations per sub-window. In comparison, this system is about 15 times faster than an implementation of the detection system constructed by Rowley et al.[3] [11]

A notion similar to the cascade appears in the face detection system described by Rowley et al. in which two detection networks are used [11]. Rowley et al. used a faster yet less accurate network to prescreen the image in order to find candidate regions for a slower more accurate network. Though it is difficult to determine exactly, it appears that Rowley et al.'s two network face system is the fastest existing face detector.[4]

The structure of the cascaded detection process is essentially that of a degenerate decision tree, and as such is related to the work of Amit and Geman [1]. Unlike techniques which use a fixed detector, Amit and Geman propose an alternative point of view where unusual co-occurrences of simple image features are used to trigger the evaluation of a more complex detection process. In this way the full detection process need not be evaluated at many of the potential image locations and scales. While this basic insight is very valuable, in their implementation it is necessary to first evaluate some feature detector at every location. These features are then grouped to find unusual co-occurrences. In practice, since the form of our detector and the features that it uses are extremely efficient, the amortized cost of evaluating our detector at *every scale and location* is much faster than finding and grouping edges throughout the image.

In recent work Fleuret and Geman have presented a face detection technique which relies on a "chain" of tests in order to signify the presence of a face at a particular scale and

[3]Henry Rowley very graciously supplied us with implementations of his detection system for direct comparison. Reported results are against his fastest system. It is difficult to determine from the published literature, but the Rowley-Baluja-Kanade detector is widely considered the fastest detection system and has been heavily tested on real-world problems.

[4]Other published detectors have either neglected to discuss performance in detail, or have never published detection and false positive rates on a large and difficult training set.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.