Machine Vision and Applications (2006) 17:163–172
DOI 10.1007/s00138-006-0022-6

ORIGINAL PAPER

Real time machine learning based car detection in images with fast training

Milos Stojmenovic

Received: 23 November 2005 / Accepted: 28 March 2006 / Published online: 30 May 2006

Abstract Our primary interest is to build fast and reliable object recognizers in images based on small training sets. This is important in cases where the training set needs to be built mostly manually, as in the case that we studied, the recognition of the Honda Accord 2004 from rear views. We describe a novel variant of the AdaBoost based learning algorithm, which builds a strong classifier by incremental addition of weak classifiers (WCs) that minimize the combined error of the already selected WCs. Each WC is trained only once, and examples do not change their weights. We describe a set of appropriate feature types for the considered recognition problem, including a redness measure and dominant edge orientations. The existing edge orientation bin division was improved by shifting so that all horizontal (vertical, respectively) edges belong to the same bin. We propose to pre-eliminate features whose best threshold value is near the trivial position at the minimum or maximum of all threshold values. This is a novel method that has reduced the training set WC quantity to less than 10% of its original number, greatly speeding up training time, and showing no negative impact on the quality of the final classifier. We also modified the AdaBoost based learning machine. Our experiments indicate that the set of features used by Viola and Jones and others for face recognition was inefficient for our problem, recognizing cars accurately and in real time with fast training. Our training method has resulted in finding a very accurate classifier containing merely 30 WCs after about 1 h of training. Compared to existing literature, we have overall achieved the design of a real time object detection machine with the least number of examples, the least number of WCs, the fastest training time, and with competitive detection and false positive rates.

M. Stojmenovic
SITE, University of Ottawa,
Ottawa, ON, Canada
e-mail: Milos22@gmail.com

1 Introduction

The objective of this article is to analyze the capability of current machine learning techniques of solving other similar image retrieval problems. By ‘capability’, we mean real time performance, a high detection rate, low false positive rate, fast training and learning with a small training set. We are particularly interested in cases where the training set is not easily available, and most of it needs to be created manually.

We will apply machine learning to the detection of rears of cars in images. Specifically, the system should be able to recognize cars of a certain type such as a Honda Accord, 2004. Therefore, input should be an arbitrary image, and the result should be that same image with a rectangle around any occurrence of the car we are searching for. In addition to precision of detection, the second major aim is real time performance. The program should quickly find all the cars of the given type and position in an image, in the same way that Viola and Jones [13] finds all the heads. The definition of ‘real time’ depends on the application, but generally speaking we would like to receive an answer for testing an image within a second or so. The response time depends on the size of the tested image, thus what appears to be real time for smaller images may not be so for larger ones.

Finally, our aim is also to design the object detection system based on a small number of training examples. We envision applications in cases where training examples are not easily available. For instance, in the case that we studied, we had to take photos of back views of a few hundred Honda Accords and other cars to create training sets, since virtually no positive images were found on the Internet. In such cases, it is difficult to expect that one can have tens of thousands of images readily available, which was the case for the face detection problem. The additional benefit of a small training set is that the training time is reduced. This enabled us to perform a number of training attempts, adjust the set of examples, adjust the set of features, test various sets of weak classifiers (WCs), and otherwise analyze the process by observing the behavior of the generated classifiers. Since the success of training with a small number of examples was unclear, we also had to set a fast training time as a goal from the very beginning so that we can perform a number of adjustments and improvements to the system. Two main contributions of this article, pre-elimination of features and shifted edge orientation bins, were made to satisfy all the goals.

We will apply machine learning methods in an attempt to solve the problem of detecting rears of a particular car type since they appear to be appropriate given the setting of the problem. Machine learning in similar image retrieval has proven to be reliable in situations where the target object does not change orientation. A classic application has become the detection of upright forward facing heads as proposed by Viola and Jones [13]. Cars are typically found in the same orientation with respect to the road. They can be photographed from various angles (front, side . . .) but they are rarely found up-side-down. The situation we are interested in is the rear view of cars. This situation is typically used in monitoring traffic since license plates are universally found at the rears of vehicles. It is also the target of choice of police traffic monitoring equipment, since their cameras are usually positioned to film the license plates for the purposes of vehicle recognition. Therefore, hardware is already in place for various software applications in vehicle detection.

The positive images were taken such that all of the Hondas have the same general orthogonal orientation with respect to the camera. Some deviation occurred in the pitch, yaw and roll of these images, which might be why the resulting detector has such a wide range of effectiveness. The machine that was built is effective for the following deviations in angles: pitch −15°, yaw −30° to 30°, and roll −15° to 15° . This means that pictures of Hondas taken from angles that are off by the stated amounts are still detected by the program.

Section 2 presents the related work. The features used in our car detection system were described and discussed in Sect. 3. Section 4 describes the AdaBoost based learning algorithm used for our object detection problem. Experimental results are given in Sect. 4. Conclusions and references complete this article. This paper is the full version of conference article [10] which only discussed the feature set used in our car detection system.

2 Related work

We are not aware of any existing solution that recognizes back view of any particular type of cars. We therefore reviewed solutions to more general problems. Existing vehicle detection systems, such as those that try to drive cars automatically along a highway do not actually detect cars on the road. They simply assume that anything that is moving on the highway is a vehicle. In scientific literature, some car recognition solutions also exist that are based on shape detectors [12]. An existing shape matching based approach [12] is reported to have a 60–85% detection rate, which is below our stated goals. The approach based on nearest neighbor matching [9] is too sensitive to viewpoint change, while the approach based on PCA (principal component analysis) [9] is not a real time system.

The most popular example of object detection is the detection of faces. The fundamental application that gave credibility to Adaboost (proposed originally in [5]) was Viola and Jones's real time face finding system [13]. Adaboost is the concrete machine learning method that was used by Viola and Jones to implement their system. In this approach, positive and negative training sets are separated by a cascade of classifiers, each constructed by Adaboost. Real time performance is achieved by selecting features that can be computed in constant time (after a pre-processing step). The training time of the face detector appears to be slow, even taking months according to some reports [14].

Viola and Jones's face finding system has been modified in literature in a number of articles. The modifications include addition of new features. Of particular interest to us were features based on gradient histograms [7], and those based on the color of certain parts of an image [8]. The Adaboost machine itself was modified in literature in several ways. We have considered all of the modifications proposed in literature, and adopted ideas that were considered helpful for achieving our goals.

We again stress that most of the successful applications of Adaboost used a large training set. In Viola and Jones's original face detector [13], 10,000 images were used in the training set. The smallest known training database for face detection was by Levi and Weiss [7]. They started to achieve detection rates in the 90% category when the number of positive examples reached

250. The number of negative examples was not specified at this level, but the authors say that they randomly downloaded 10,000 images from the Internet containing over 100,000,000 sub-windows. They only moderately increased their detection rates as the size of the positive set grew drastically.

We apply a similar general design as in [13]. The object search is based on a machine that takes in a square region of size equal to or greater than 24 × 24 pixels (for the face search) [13] (we had a limit of 100 × 150 for the car search) as input and declares whether or not this region contains the searched object. We use such a machine to analyze the entire image. We pass every sub-window of every scale through this machine to find all sub-windows that contain faces. A sliding window technique is therefore used. The window in [13] is shifted 1 pixel after every analysis of a sub-window (we used a 2 pixel shift to speed things up, without notable negative impact). Both dimensions of the sub-window grow in both length and width 10% every time all of the sub-windows of the previous size were exhaustively searched.

One of the key contributions in [13] was the introduction of a new image representation called the "*Integral Image*", which allows the features used by their detector to be computed very quickly. In the pre-processing step, Viola and Jones [13] find the sums $ii(x, y)$ of pixel intensities (or other measurements) $i(x', y')$ for all pixels (x', y') such that $x' \leqslant x, y' \leqslant y$. This can be done in one pass over the original image using the following recurrences: $s(x, y) = s(x, y-1) + i\ (x, y), ii\ (x-1, y) + s(x, y)$ (where $s(x, y)$ is the cumulative row sum, $s(x, -1) = 0$, and $ii\ (-1, y) = 0$). The feature value in the rectangle with corners (x_1, y_1) (bottommost), (x_2, y_2), (x_3, y_3) and (x_4, y_4) (uppermost) is then $ii(x_1, y_1) + ii(x_4, y_4) - ii(x_2, y_2) - ii(x_3, y_3)$ [13].

3 Features used in recognition

A feature is a function that maps an image into a real number. We will give more details on the features used in the training procedure here. Two types of basic features were used. They were redness features and dominant edge orientation features. The dominant edge orientation and redness features proved themselves to be much better than Viola and Jones's [13] original set, for the problem we studied, the detection of the Honda Accord 2004 from behind (see details in Sect. 5).

3.1 Redness features

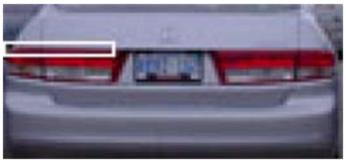

Fig. 1 Redness feature

The redness features we refer to are taken from the work of [8]. They concentrated on finding circular red regions in images. Their goal was to find and fix red eyes in pictures taken of people. Most of their work focused on the shape of the red regions found, rather than the techniques involved in finding the color red. We borrow their idea of finding predominantly red regions. Our work differs in the fact that we look for red regions that signify the stop lights of the Honda accord, as opposed to human eyes. Therefore, our red regions are rectangular, and much larger than theirs. Figure 1 shows an example of a redness feature determined by the training process.

A special 'redness' color space was formed during pre-processing to assist in the detection of red areas. This color space was taken from [8], and is a one dimensional linear combination of the RGB color space. All of the positive and negative inputs in the training set are RGB color images which means that each of their pixels is represented by three 8-bit numbers that represent the quantity of red, green and blue in a pixel, respectively. Each pixel in the redness colour space is defined as follows: $Redness = 4R - 3G + B$ [8]. The integral image computation (the technique is proposed in [13]) was applied to the redness image to produce one of the inputs to the training procedure. The areas of the redness training features were determined in constant time using the integral image of the redness image.

3.2 Edge orientation features

Several dominant edge orientation features were used in the training algorithm. A well known greyscale Sobel gradient mask (three pixels by three pixels) [4] is used in determining the location of edges in an image. The mask is applied in both coordinate directions, and the combined intensities (called Laplacian intensities, based on Euclidean distance) are taken. One more detail of our implementation is the threshold that was placed on the intensities of the Laplacian values. We used a threshold of 80 to eliminate the faint edges that are not useful. A similar threshold was employed in [7]. The orientations of each pixel are calculated from its intensity in both directions. The orientations are divided into six bins so that similar orientations can be grouped together. The whole circle is divided into six bins. It is important

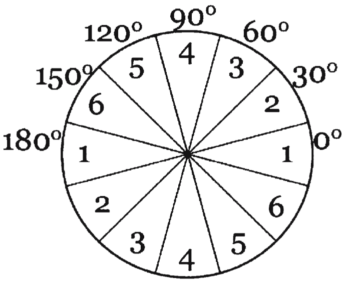

Fig. 2 Six orientation bins

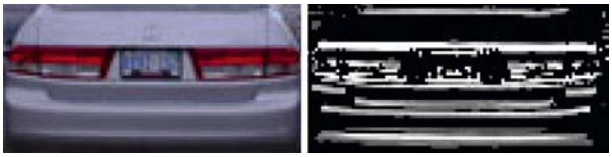

Fig. 3 Honda and corresponding horizontal edges

to note that the orientations of the 0°, 90°, and 180° bins are critical in identifying Hondas. They are important since Hondas mainly have horizontal and vertical edges. The division of the bins which places 0° or 90° at the border of two bins poses problems since all vertical and horizontal edges can fall into two bins. We handle this problem by shifting all of the bins by 15°. Note that [7] did not mention any bin shifting, so we believe that they used non-shifted bins. They did however, vary the number of bins from 4 to 8. Effectively the number of bins does not impact the performance much, but these boundaries are avoided. To fit all of the orientations into the 0–180° range, we add 180° if the angle is smaller than 0°, and subtract 180 if the angle is greater than 180°. The effects of these transformations can be seen in Fig. 2. In Fig. 3, we see a Honda accord and its corresponding edge orientation image for the first bin $\left[-15^\circ,+15^\circ\right]\cup\left[+165^\circ,+195^\circ\right]$. Bin shifting significantly contributed to the accuracy of the system. The detection rate improved from 74.3 to 89.1%, while the number of false positives decreased from 168 to 26.

Since we use six bins, we create six images where each image represents an edge orientation bin. Each value $B(i,j)$ of each orientation bin image is the corresponding Laplacian intensity if the orientation falls into this bin, and 0 otherwise. As we can see from Figs. 3 and 4, in a given region in an image, there is typically one orientation that is dominant. We exploited this fact in our use of dominant edge orientations. This idea was first developed by [7]. Dominant edge orientations are calculated as the total edge intensities of a given orientation divided by the sum of all edge intensities of all orientations in the same region. Dominant edge orientation features were used for training. We see some less distinguishing, yet nonetheless noticeable edges in some of the other bins in Fig. 4. One of the most successful edge orientations was the horizontal one depicted in Fig. 3. All of the possible dominant edge bins were offered to the training procedure except for bin 2 since it visually had nothing distinguishing about it.

Integral images (for constant time calculation of feature values [13]) were created from these orientation bin images and were used in the training procedure to find sums of edge intensities. Seven features were used in the training procedure. They were: dominant edge orientations (1, 3, 4, 5 and 6) and the redness feature. One of the problems we anticipated was that edges might not be so clearly defined in larger examples of Hondas. By larger we mean Hondas that are larger than 100 × 50 pixels. In the examples in our training set, edges determined by the Sobel masks are very distinctive since the images are very small. We feared that in larger examples of Hondas, edges would be represented by thicker lines, and result in different edge orientation maps. This did not happen since Hondas have crisp, well defined lines. Even in larger images, edges are very similarly defined compared to those of smaller images. Furthermore, it is the dominant edge orientation we are interested in when measuring feature values. Most of the WCs in the strong classifier were in the areas of the image in which their orientation was often the only one present. Therefore, any quantity of edges in such an area would be enough to help identify them positively.

The redness features are not normalized in any way, and the scaling of these features must be handled differently. Since the redness quantity in a rectangle directly depends on the area of the rectangle, threshold Θ of redness features is multiplied by the square of the scaling factor before the scaled redness feature is compared against it. This operation normalizes the scaling effect.

3.3 Reducing the training time by selecting a subset of features

A WC is a function of the form $h(x,f,s,\theta)$ where x is the tested sub-image, f is the feature used, s is the sign ($+$ or $-$) and θ is the threshold. The sign s defines on what side of the threshold the positive examples are located. Threshold θ is used to establish whether a given image passes a classifier test in the following fashion: when feature f is applied to image x, the resulting number is compared to threshold θ to determine how this image is categorized by the given feature. The equation is given as $sf(x) < s\theta$. If the equation evaluates true, the image is classified as positive. The function $h(x,t,s,\theta)$ is then defined as follows: $h(x,f,s,\theta) = 1$ if $sf(x) < s\theta$ and -1

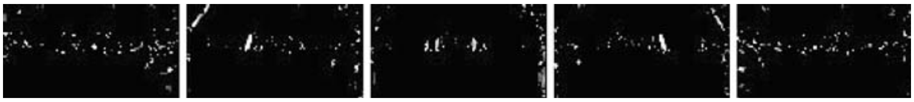

Fig. 4 Edge orientation bins [2..6] for Honda in Fig. 2

otherwise. This is expected to correspond to positive and negative examples, respectively.

We proposed to pre-eliminate features whose corresponding best threshold value θ is near the trivial position at the maximum or minimum of feature values. This is a novel method that has reduced the set of available WCs to less than one-tenth of its original size (for the case we studied), greatly speeding up training time, and showing no negative impact on the quality of the final classifier. In the initial training of the WCs, each WC is evaluated based on its cumulative error of classification (*ce*). The cumulative error of a classifier is $ce = (false\_positives + missed\_examples)/q$.

An alternative definition (available also in OpenCV) that we tested is $nce = (false\_positives/n + missed\_examples/p)/2$, where n and p denote the numbers of negative and positive examples, and $q = n + p$. We refer to this as the normalized cumulative error (*nce*). This error function tries to more equally treat both types of training examples. It is theoretically better to have such a normalized error function when there is a large discrepancy in the sizes of the positive and negative training sets. In our case, the set of positives numbered 154 examples, and the set of negatives contained 760 examples.

Weak classifiers that had a *ce* that was greater than a pre-determined threshold were automatically (in our program) eliminated from further consideration. In our implementation, no WC could have a worse cumulative error than $\min(n,p)/q$. This is due to the fact that the threshold Θ is initially inserted before the beginning or after the end of the sorted list of training records in each WC. This means that it initially classifies all records to be either all negative, or all positive, respectively. If there is no better place for Θ within the list of training records, it remains where it was first placed, with the ce that it was originally awarded. The pre-determined threshold mentioned above was set as $\min(n,p)/q - 0.01 \times \min(n,p)/q$. This means a minimum of 1% improvement over the trivial initial error was needed for a WC to be accepted into the next round of selection. Luckily, the distribution of the WCs is heavily biased passed the $\min(n,p)/q$ boundary. This means that most WCs are very poor, and have the maximum *ce* (which is equal to $\min(n,p)/q$) which means that they were eliminated early from the training procedure. This distribution of WC efficiency is illustrated by the training results of the best strong classifier generated by our program. We deal with roughly 530,000 WCs. For example, let us assume that there are 150 positive and 750 negative examples in a training set. Therefore, the greatest cumulative error any one of the WCs could be assigned is approximately 16.67%. We adjust our threshold to accept all weak classifiers that have a cumulative error equal or better than $16.67 - 0.01 \times 16.67 = 16.5\%$. According to our experiments, it turns out that there are only ≈15,000 WCs that satisfy this requirement out of a total of ≈530,000. That means that over 97% of all initial WCs are eliminated from further training. This drastically reduces training time while having little impact on the quality of the pool of available WCs to choose from. The final results of the strong classifier do not suffer from this reduction of unnecessary WCs as is evident from their high detection rates and low false positive rates.

Since the cumulative error function was altered when considering *nce*, the triviality threshold also had to be adjusted. The new triviality threshold was determined to be $2 \times \min(n,p)/q - 0.01 \times \min(n,p))/q$. This threshold leaves roughly the same number of WCs after the first round of training. It reduces training time in the same way as the original triviality threshold. Significant changes in n or p may require adjustments to the threshold.

4 AdaBoost based learning algorithm used for training

We described a novel variant of the AdaBoost based learning algorithm, which builds a strong classifier by incremental addition of WCs that minimize the combined error of the already selected WCs. Each WC is trained only once, and examples do not change their weights. While all the individual components of this approach exist in literature, it was not yet used as a combined whole algorithm the way we propose here.

Our variant of the AdaBoost learning algorithm is similar in flavor to the alternative voting AdaBoost variant described by Wu, Rehg and Mullin [15]. Both algorithms train WCs only once, at the beginning. There are several important differences in the two methods. In [15], each feature is trained so that it has minimal false positive rate. In our variant, each feature is trained to minimize a single combined error, which includes both false positives and missed positives. In [15], a new feature is added to either minimize the false positive or maximize the detection rates, depending on the current detection rate. In our variant, a new feature is always one that minimizes the combined error of the classifier. Next, the decision of a classifier in [15] is made by majority voting (where each WC has equal weight). In our approach, the weights of each WC are decided at the beginning of the training process, and the decision of each classifier is made by weighted voting. Next, [15] considered different weights for positive and negative examples. We considered equal weights $1/q$ for the case where $ce = (\mathit{false\_positives} + \mathit{missed\_examples})/q$, and weights $1/n$ and $1/p$ for the case where $nce = \mathit{false\_positives}/n + \mathit{missed\_examples}/p$. Finally, our variant is a single strong classifier while [15] described a cascaded design.

A strong classifier is obtained by running the Adaboost machine. It is a linear combination of WCs. A WC is constructed from a feature and a threshold. We assume that there are T WCs in a strong classifier, labelled $h_1, h_2, \ldots, h_T$, and each of these comes with its own weight labelled $\alpha_1, \alpha_2, \ldots, \alpha_T$. The tested image x is passed through the succession of WCs $h_1(x), h_2(x), \ldots, h_T(x)$, and each (WC) assesses if the image passed its test. The recommendations are either -1 or 1, multiplied by their corresponding weight. Note that $h_i(x) = h_i(x, f_i, s_i, \theta_i)$ is abbreviated here for convenience. The decision that classifies an image as being positive or negative is made by the following test: $\alpha_1 h_1(x) + \alpha_2 h_2(x) + \cdots + \alpha_T h_T(x) > 0$.

4.1 Training optimal WCs

In the original approach [5,13], examples are weighted, and weights change in the process. WCs are re-trained after selecting any of them for the strong classifier. In our algorithm, all WCs are trained only once, at the beginning of the training process. They do not change in the process afterwards, therefore the needed values can be memorized. The input consists of feature f and all positive and negative examples. The algorithm scans through the sorted list of feature values, looking for threshold θ and direction s that minimizes the classification error, which is the total number of misclassified examples. The output is specified below.

Algorithm: Training optimal weak classifiers

Input: Feature f, n negative examples, p positive examples,

Output: Threshold Θ, sign s, *false_pos, missed,* weight α.

Calculate records $(f(x_i), y_i)$, where $y_i = 1$ for a positive example, and =−1 for a negative example (using integral images where appropriate). Sort these records by the $f(x_i)$ field by any sorting algorithm, e.g. mergesort, in increasing order. Let the obtained array of the $f(x_i)$ field be: $g_1, g_2, \ldots, g_q$. The corresponding records are $(g_j, \text{status}(j)) = (f(x_i), y_i)$, where $g_j = f(x_i)$. That is, if the j-th element g_j is equal to i-th element from the original array $f(x_i)$ then $\mathit{status}(j) = y_i$.

```
$s = 1; sp = 0; sn = 0;$ (*number of positives/negatives below a considered threshold *)
If $n<p$ then {$misclassified = n; \theta = g_q+1$} (*all declared positive*)
else {$misclassified = p; \theta = g_1 - 1$}; (*all declared negative *)
For $j=1$ to $q-1$ do {
  If $status(j)=+1$ then $sp= sp + 1$ else $sn = sn + 1;$
  If $sp + n - sn < misclassified$
    then {$misclassified = sp + n - sn; s = -1; \theta = (g_j + g_{j+1})/2$
      $false\_pos = n - sn; missed = sp$};
  If $sn + p - sp < misclassified$
    then {$misclassified = sn + p - sp; s = 1; \theta = (g_j + g_{j+1})/2;$
      $false\_pos = sn; missed = p - sp$}
};
```

The output is a WC $h(x) = h_I(x, f, s, \theta)$. The detection rates and false positive rates of WCs can also be considered output at this stage, as $(p - \mathit{missed})$/p and *false_pos/n*, respectively. Variables *missed* and *false_pos* denote the number of misclassified positive and negative examples, respectively.

The relative error of the constructed WC is $e = \mathit{misclassified}/q = (\mathit{false\_pos} + \mathit{missed})/(p + n)$, and is used to decide the weight of the constructed WC as follows:

$\beta = e/(1 - e)$, and $\alpha = -\log(\beta)$. The assigned weight is α.

End of Algorithm.

Adaboost therefore assigns large weights with each good WC and small weights with each poor WC. Note that the algorithm corresponds to the variant with combined error $ce = (\mathit{false\_positives} + \mathit{missed\_examples})/q$. If the alternate formula $nce = (\mathit{false\_positives}/n + \mathit{missed\_examples}/p)/2$ is used, some minor changes to the algorithm are needed to reflect the weights of the positive and negative misclassifications being different (proportional to $1/p$ and $1/n$, respectively).

Optionally, the value of α could be limited. In the best performing variant of the protocol for the case we studied, if $\alpha > 1$ then $\alpha = 1$ is executed. This was directly applied only to the choice of the first WC in our scenario, but (indirectly) impacted the selection of the others, including their number and overall performance.

4.2 Training the best classifier

First, all WCs are trained, as described, and the training process returns classification errors *missed* and *false_pos*. It also returns the weight α for each WC $h(x)$. The combined error can be defined in one of several ways, such as $ce = (missed + false\_pos)/q, ce = missed/p + false\_pos/n$, ce $= \lambda *$ missed$/p + (1 - \lambda) false_pos/n$, where λ is a weighting parameter. In our implementation, we use the trivial $ce = (missed + false\_pos)/q$. For each feature, find the optimal WC as described above. Then the construction of a classifier proceeds as follows.

Algorithm: Training the best classifier

Input: set of weak classifiers $h_i(x)$, weights α_i, n negative examples, p positive examples

Output: series of selected weak classifiers $h_1, h_2 \ldots h_T$, and their weights $\alpha_1, \alpha_2 \ldots \alpha_T$.

Select the first WC $h_1(x)$ (and its weight α_1) as the one that has minimal combined error ce;

Set $T = 1$; (* the number of WCs in the classifier *)

```
Repeat
   For each WC h(x) calculate the combined error of the classifier
      $\alpha_1 h_1(x) + \alpha_2 h_2(x) + \ldots + \alpha_T h_T(x) + \alpha h(x)$
      and select h(x) that minimizes the error; find its weight $\alpha$;
   $T = T + 1, \alpha_T = \alpha, h_T(x) = h(x)$;
Until (detection rate $(p - missed)/p \geq d$ and false positive rate $false\_pos/n \leq fp$) or $T \geq Tmax$.
```

End of Algorithm.

Note that values $\alpha_1 h_1(x) + \alpha_2 h_2(x) + \cdots + \alpha_T h_T(x)$ can be memorized so that testing candidates is faster. In the test, *false_pos* and *missed* are the numbers of incorrectly classified negative and positive examples, respectively, by the tested classifier. This section of code is executed for every feature, and for every example in the training sets, up to $T_{\max}$ times. The method takes $O(Fq \log q)$ time to train all of the classifiers in the initial stage, where F is the number of WCs, and q is the number of examples. We are left with f WCs, where $f << F$, after the elimination of poor WCs that do not improve the cumulative error more than 1% from the trivial position. Testing each new WC while assuming that the summary votes of all classifiers are previously stored would take $O(q)$ time. It would then take $O(fq)$ time to select the best WC. Therefore, it takes $O(Tqf)$ time to chose T WCs. We deduce that it would take $O(Fq \log q) + O(Tqf)$ time to complete the training using our method (the same time complexity applies to the variant described by [15]). Since $f << F$, the dominant term in the time complexity is $O(Fq \log q)$. Had F and f been roughly equal, the dominant term would have been $O(Tqf)$.

5 Experimental results

Our experimental setup generally follows the ones used in relevant references for face detection. We started by implementing and training with the original set of features [13]. Pre-elimination was added early in the design. Because of the experienced inefficiencies with respect to the set goal, new features were added to the initial set. In parallel, after providing some initial negative examples, false positives were added to the negative example pool from an unrelated image set. This method is known as bootstrapping [11]. We confirmed that this method has its limits, since the continued application of it (overfitting) starts to 'attack' the best WCs and consequently starts to reduce the accuracy of the classifier. Our experiments indicated that the set of features used by Viola and Jones [13] was inefficient for our goals. Not only were their cumulative errors in the training process inferior to other features, but their very presence in the feature set greatly, yet fruitlessly, increased training time. Haar wavelets [13] were eliminated from the training procedure completely early in the implementation and testing phase, at the point where none of them were selected for the strong classifier. At the end, we have built a fast and reliable object recognizer based on small training set, consisting of 155 positive and 760 negative images, with the feature set described here. It detects back views of Honda Accords with a 98.7% detection rate and 0.4% false positive rate on the training set. We designed a strong classifier with a record low number of WCs (30). Compared to existing literature, we have achieved the overall design of a real time object detection machine with the least number of examples, the least number of WCs, and with competitive detection and false positive rates.

It was a common occurrence in the testing phase that sub-windows around an actual positive example would also be considered positive. This problem was solved by storing the positive examples in a vector during the detection phase, and comparing new positives to the ones stored in the vector. If an already stored example was found that had a weaker decision than the new positive and was too close to the positive, it was replaced. Two positive rectangles are judged to be too close if their top left corners are within 60 pixels of each other.

In order to verify the impact of the feature set on the quality of the final classifier, we ran, at the end, the original feature set [13] on the same training set, with 30 and 80 WCs, respectively. The one with 30 WCs was slightly faster while the one with 80 WCs was 2.5 times slower than our selected classifier. The detection rates on the training set were 94 and 95%, respectively.

Since there exist no standardized test sets for the detection of any cars, let alone one specific car, our machine was tested on a set that was created the same way the training set was created. Pictures were taken of cars around town. Our test set boasts 106 images that contain 101 positive examples of the Honda accord

2004. The positives in the set are in various scales and positions within the images. They are also taken from a variety of angles that are detectable by our program. The test set images themselves also come in a variety of sizes. The smallest images are basically the same size as those used by Viola and Jones [13] (320 × 240 pixels). The largest image size in the test set is 640 × 480 pixels.

Our object recognizer performed with 89.1% detection rate and 26 false detections on a test set containing 106 images of different sizes. These numbers are very good when compared to other systems such as those put forward by Viola and Jones [13] and Levi & Weiss [7]. Viola and Jones's face detection system was tested on a set that contained 130 images with 507 positives. Keep in mind that gathering such a test set is much easier when positives are faces. Our test has a similar number of images, yet a much smaller number of positives. Nevertheless, it is a sufficient comparison base to use as a basis for discussion. Viola and Jones [13] gave statistics for the number of false positives their system produced at various detection rates. At a detection rate of roughly 89% (such as our system), their system produced roughly 35 false positives. They, however, used a much greater number of classifiers to achieve this result. Levi and Weiss [7] used the same test set as Viola and Jones to evaluate their system and they achieve an 89% detection rate at the cost of roughly 45 false positives. They used a 2,500 item training database to achieve these results.

We also studied the impact of various decisions we made in the process. The detection rates for Honda's using only WCs from [13] with 30 and 80 classifiers were 57.4 and 56.4% (surprisingly lower for a higher number of WCs), and with 253 and 194 false positives, respectively. This might provide a basis for a cascaded design but is clearly inferior to our selected set which solved the problem satisfactorily well without resorting to this step (even though it was originally anticipated).

One of the major contributions in this work is the shifting of the edge bins. It might seem like a trivial matter which would not impact the results of the algorithm much, but in fact it greatly contributes to the performance of the system. Two variants of the algorithm were trained and tested to illustrate this point: one with the bin shifting approach, and one without. Both systems were trained and subsequently tested on the same training and testing sets, each composed of 30 WCs. The detection rates were 89.1 and 74.3%, while the number of false positives were 26 and 168, respectively. From this data, we see that a simple shift of the bin orientations significantly impacts the results of the system. Nearly seven times as many false positive detections are reported by the non-bin shifting method. The detection rates also noticeably differ.

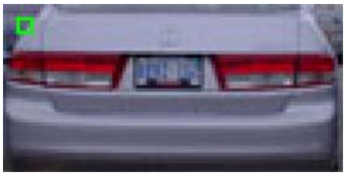

Fig. 5 WC1

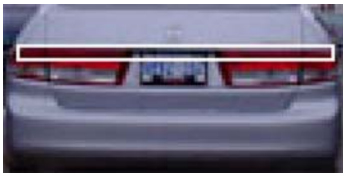

Fig. 6 WC2

The described AdaBoost learning machine, without limiting α, had a perfect performance on our training set (described below): 100% detection rate and zero false positives. However, it had 88% detection rate and 38 false positives on the testing set. We then made a minor change in the AdaBoost machine. If the weight α of a selected WC was $\alpha > 1$ then we reduced it to $\alpha = 1$. This was effectively applied only on the first classifier (in our experiments) but had an impact on selecting the others (including a small increase in the number of WCs needed to reach satisfactory results on the training set). However, the results on the testing set were improved in both detection rate and false positives.

The results of the normalized error function (*nce*) are somewhat ambiguous. The detection rate of the obtained machine is 92.1%, yet the false positive count is 117! The false positive rate is roughly four times higher when using the normalized error function. It is interesting to note that the detection rate during the training phase is 100%, with a 0.008% false positive rate. These results carried over to the testing phase as is evident by the relatively high detection rate.

The training procedure produced interesting results when it came to the selection of WCs. Figures 5 and 6 show the best and second best WCs as chosen by the training algorithm.

The best WC as determined by the system was an edge detector applied to the upper-left hand corner of the Honda. The small green box in Fig. 4 defines this WC. It detects the 45° edge that is dominant in this region. After reflecting back on our test set, it became evident that most of the positive examples share this attribute. It is not a WC that we would have chosen by hand had we tried static selection of features. The second best WC was a redness feature that detected the rear stop lights of the Honda (Fig. 5). The selection of this feature validated our assumption that a redness feature used for detecting the rear stop lights would be very important. This just further emphasizes our belief that the basic features selected for a given detection problem should be custom selected before training starts to produce better results down the road. The other WCs that were chosen identify many areas of horizontal edges that are common, redness features that define each stop light separately, and areas that do not have a specific orientation of edge. An example of such an occurrence is the area just below the license plates. This area is mostly void of any vertical edges.

The speed at which images were processed while testing is shown here. The computer configuration is an AMD athlon 2800 processor. Images as small as 150 × 120 pixels are processed in 0.05 seconds. Images of size 170 × 227 pixels are processed in 0.18 s. Standard size images similar to those used in Viola and Jones [13] and other papers of size 320 × 240 are processed in 0.49 s. We consider this to be real time. It takes more and more time to process larger pictures. For example, it takes 1.93 s to process a picture of size 500 × 253. The size of the picture directly impacts the time it takes to process it. This is logical since larger images contain more sub-windows that must be searched. In fact, the running time is proportional to the number of features contained in a window of a given size. In our implementation, widths and heights of sub-windows grow by 10%. The time complexity is therefore $O(AT\log(b/c))$, where b is the image width, c is the minimal example width, T is the number of WCs in the strong classifier and A is the area of the searched window, since there are $O(A)$ features of a given size, and $O(\log(b/c))$ incremental steps. When T is fixed, the complexity may also be expressed as $O(b^2\log(b))$, where $b^2 \approx A$. We see that the complexity grows faster than quadratic time with respect to image width.

6 Conclusions

We began our research with the following question. Is there any 'magic' software package that can find any type of object in an image, reasonable accurately, and in real time, merely by replacing the positive and negative sets? If the answer was yes, there would not be so much research in this area. However, the AdaBoost software framework appears to be widely adopted for real time object detection. For example, Le and Satoh [6] recently claimed that cascaded AdaBoost is about 1000 times faster than a support vector machine approach.

The training program can be considered as being composed of three components: an AdaBoost classifier, a Feature set, and a Training set. The AdaBoost classifier is a general framework, which can be safely claimed to be applicable to the recognition of any type of object efficiently, provided the object roughly appears with the same orientation and angle (e.g. straight upfront faces, or backs of horizontal cars such as our positive).

The feature sets are not as general. Viola and Jones's set of features [13] was successfully used for recognizing similar types of objects such as lion faces [2]. However, we show that the feature set for recognizing faces is completely different (practically disjoint) from the feature set for finding cars. Some existing articles added new features to help in recognizing objects which are different from faces, but we did not see anyone actually making the two sets disjoint. On the other hand, the same set of features could be used to recognize different objects, by simply replacing one training set with the other. For instance, we believe that one could equally well recognize the back of another car such as the Toyota Camry 2004 by collecting the corresponding pictures for the training set and using the version of AdaBoost described here. We also believe that our system can be extended to recognize, in real time, any one of a number of car types; following the approach of sharing visual features for multiclass and multiview object. Common general features for several car types would be selected first, followed by a selection of individual features to differentiate between the cars. Further extensions may consider the presence of partial occlusions and shadows (studied by Barczak [1] for the case of face detection).

In addition to defining a good feature set for Honda's, we have proposed a general elimination step in the program, by introducing a threshold for the quality of a feature, to reduce training time. Pre-elimination is applicable to all object detection problems for speeding up the training process by experimentally finding the proper threshold value, but no level of improvement is guaranteed in advance for other problems. It is an open area to further elaborate on the applicability of common feature sets, fully or partially, in recognition of some objects. One can always merge two sets into one, threshold them for triviality on a given training set, and then claim that the same feature set is applicable to recognition of two totally distinct objects. For instance, one can add the set of dominant edge orientations to Viola and Jones's set for face detectors, and use them to train either faces or cars. When recognizing faces from appropriate training sets, redness features are eliminated, while many dominant edge orientation features may remain. When the same set is applied to recognize Honda's, all of Viola and Jones's features are basically removed first before real training, with the idea that we proposed (to the best of our knowledge, our system is the first real time AdaBoost based system that recognizes an object without Viola and Jones's features). But one cannot claim that this process can be continued to eventually include any type of objects, given the desired performance metrics. A new type of object may always exist that has its specific feature that works ideally for it, and needs to be added. An example is a circular object where a hypothetical roundness measure may be used to help identify it. For instance, Blaschko et al. [3] considered a variety of features separated into five groups: simple shape, moments, contour representations, along with both differential and texture features, for automatic in situ identification of plankton. We believe that one can further develop the idea presented here of introducing an automatic feature triviality test and link it to this discussion. Simply speaking, features from a large set are put through this test, and only some of them pass. Given two objects to recognize, one can define a measure of their similarity by looking at the number of common and different remaining features.

There exists no 'magic' answer that can easily explain the effort required to apply the techniques discussed here to the recognition of other objects such as faces. More research needs to be done to be able to quantitatively answer this point. The simple answer is that other types of cars can be recognized just by replacing the training sets in this work. Our program is not restricted to only recognizing cars. We believe that dominant edge orientation features are powerful ones, and are applicable in many scenarios. For instance, fire extinguishers mainly have horizontal and vertical edges. Also, the redness feature is applicable in searching for fire extinguishers, given that fire extinguishers are mostly red. Therefore, we are confident that fire extinguishers (which are vertical in position and visually of the same shape in robotic vision applications) can be recognized with our software, because they appear quite simple to recognize, much simpler than the backs of cars. Our system might also recognize fire extinguishers even if dominant edge orientation features are removed (that is, based solely on the redness feature), because of their clear rectangular shape, and typical red color.

The training and testing programs, the positive and negative examples used in the training, along with the testing sets are all available at http://www.site.uottawa.ca/∼mstoj075/.

Acknowledgements We thank the anonymous referees for their constructive comments that have greatly improved the clarity of the article. This work is supported by OGS.

References

1. Barczak, A.L.C.: Evaluation of a boosted cascade of Haar-like features in the presence of partial occlusions and shadows for real time face detection, LNAI 3157. In: Proceedings of the PRICAI 2004, 969–970 (2004)
2. Burghardt, T., Calic, J.: Analysing Animal Behaviour in Wildlife Videos Using Face Detection and Tracking, submitted to IEE Proceedings Vision, Image & Signal Processing, Special issue on the Integration of Knowledge, Semantics and Digital Media Technology, March (2005)
3. Blaschko, M., Holness, G., Mattar, M., Lisin, D., Utgoff, P., Hanson, A., Schultz, H., Riseman, E., Sieracki, M., Balch, W., Tupper, B.: Automatic In Situ Identification of Plankton. In: Proceedings of IEEE Workshop on Applications of Computer Vision, Breckenridge, Colorado, Jan. 5–7 (2005)
4. Efford, N.: Digital Image Processing: a Practical Introduction Using Java. Addison Wesley, Reading (2000)
5. Freund, Y., Schapire, R.E.: A decision-theoretic generalization on on-line learning and an application to boosting. J. Comput. Syst. Sci. **55**(1), 119–139 (1997)
6. Le, D., Satoh, S.: Fusion of Local and Global Features for Efficient Object Detection, IS & T/SPIE Symposium on Electronic Imaging (2005)
7. Levi, K., Weiss, Y.: Learning Object Detection from a Small Number of Examples: the Importance of Good Features, International Conference on Computer Vision and Patern Recognition *(CVPR)*, (2004)
8. Luo, H., Yen, J., Tretter, D.: An Efficient Automatic Redeye Detection and Correction Algorithm, 17th IEEE International Conference on Pattern Recognition, (ICPR'04) V. 2, Aug. 23–26, 2004, Cambridge
9. Petrovic, V.S., Cootes, T.F.: Analysis of Features for Rigid Structure Vehicle Type Recognition, Proceedings of British Machine Vision Conf. 2004, vol. 2, pp. 587–596
10. Stojmenovic, M.: Real time car detection in images based on an Adaboost machine learning approach and a small training set. In: Proceedings of IEEE International Workshop on Systems, Signals & Image Processing IWSSIP, Chalkida, Greece, Sept 22–24 (2005)
11. Sung, K., Poggio, T.: Example based learning for view-based human face detection, IEEE Trans. Pattern Anal. Mach. Intell. **20**, 39–51 (1998)
12. Thureson, J., Carlsson, S.: Finding Object Categories in Cluttered Images Using Minimal Shape Prototypes. In: Proceedings of 13th Scandinavian Conference on Image Analysis SCIA, Goteborg, Sweden (2003)
13. Viola, P., Jones, M.: Robust real-time face detection, Int. J. Comput. Visi. **57**(2), 137–154 (2004)
14. Verschae, R., Ruiz-del-Solar, J.: A Hybrid Face Detector based on an Asymmetrical Adaboost Cascade Detector and a Wavelet-Bayesian-Detector, Lecture Notes in Computer Science 2686, Springer, Berlin Heidelberg NewYork, 742–749 (2003)
15. Wu, J., Regh, J., Mullin, M.: Learning a Rare Event Detection Cascade by Direct Feature Selection. Proceedings Advances in Neural Information Processing Systems 16 (NIPS*2003), MIT Press, Cambridge (2004)