Pre-Eliminating Features for Fast Training in Real Time Object Detection in Images with a Novel Variant of AdaBoost

Milos Stojmenović
SITE, University of Ottawa,
Ottawa, Ontario, Canada K1N 6N5
mstoj075@site.uottawa.ca

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 described 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 proposed to pre-eliminate features whose cumulative error of corresponding best WCs exceeds a predetermined threshold value. We tested two straightforward definitions of cumulative error. In both cases, we showed that, when over 97% of the initial features are eliminated at the very beginning from further training, training time is drastically reduced while having little impact on the quality of the pool of available WCs. This is a novel method that has reduced the training set WC quantity to less than 3% of its original number, greatly speeding up training time, and showing no negative impact on the quality of the final classifier. Our experiments indicated that the set of features used by Viola and Jones and others for face recognition was inefficient for our problem; therefore, each object requires its own custom-made set of features for real time and accurate recognition. Our training method, combined with the selection of appropriate features, has resulted in finding a very accurate classifier containing merely 30 weak classifiers. 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 weak classifiers, the fastest training time, and with competitive detection and false positive rates.

Keywords: AdaBoost, real time, object detection.

I. INTRODUCTION

The goal of this article is to analyze the capability of current machine learning techniques of detecting objects in images different from faces**.** By 'capability' we mean real time performance, a high detection rate, low false positive rate, 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 have applied 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 goal 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 [VJ] find 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 goal is to also 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, and otherwise analyze the process by observing the behaviour of the generated classifiers.

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 [VJ]. 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 -15^0, yaw -30^0 to 30^0, and roll -15^0 to 15^0. 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 in section 3 (details are given in a companion article [S]). Our main contribution, pre-elimination of features, is elaborated on in section 4. Section 5 describes the AdaBoost based learning algorithm used for our object detection problem. Experimental results are given in section 6. Conclusions and references complete this article.

II. Literature Review

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 [TC]. An existing shape matching based approach [TC] is reported to have a 60%-85% detection rate, which is below our stated goals. The approach based on nearest neighbour matching [PC] is too sensitive to viewpoint change, while the approach based on PCA (Principal Component Analysis) [PC] 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 [FS]) was Viola and Jones' real time face finding system [VJ]. 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. One of the key contributions in [VJ] 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.

Viola and Jones' 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 [LW], and those based on the color of certain parts of an image [LYT]. 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' original face detector [VJ], 10,000 images were used in the training set. The smallest known training database for face detection was by Levi and Weiss [LW]. 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 [VJ]. The object search is based on a machine that takes in a square region of size equal to or greater than 24x24 pixels (for the face search) [VJ] (we had a limit of 100x150 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 [VJ] is shifted 1 pixel after every analysis of a sub window (we used a 2 pixels 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.

III. Features used in Recognition

We described a set of appropriate feature types for the considered recognition problem, including a redness measure and dominant edge orientations, in companion article [S], where details and formal definitions are given. The existing edge orientation bin division was improved by shifting, so that all horizontal (vertical, respectively) edges belong to the same bin.

A feature is a function that maps an image into a real number. Our experiments indicated that the set of features used by Viola and Jones [VJ] was inefficient for our problem. Viola and Jones' Haar wavelets were eliminated from the training procedure completely early in the implementation and testing phase since they failed to produce any usable results. Not only were their cumulative errors in the training process too great to be useful, but their very presence in the feature set greatly, yet fruitlessly, increased training time. Therefore, each problem requires its own custom-made set of features. Two types of basic features were used. They were redness features (see also [LYT], and dominant edge orientation features (see also [LW]). Figures 1 and 2 show the best and second best weak classifiers as chosen by our training algorithm.

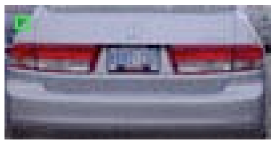
Figure 1. WC1

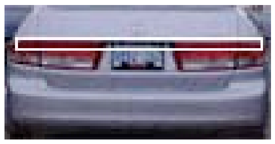
Figure 2. WC2

The best weak classifier as determined by the system was an edge detector applied to the upper-left hand corner of the Honda. The small green box in Figure 4 defines this weak classifier. It detects the 45 degree edge that is dominant in this region. The second best weak classifier was a redness feature that detected the rear stop lights of the Honda (Figure 2). The selection of this feature validated our assumption that a redness feature used for detecting the rear stop lights would be very important. The other weak classifiers 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.

IV. Reducing Training time by selecting a subset of features

A weak classifier (WC) is a function of the form *h(x, f, s, θ)* 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 can 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, θ)* is then defined as follows: $h(x, f, s, \theta)=1$ if $sf(x)<s\theta$ and *-1* otherwise. This is expected to correspond to positive and negative examples respectively.

Our main contribution in this article is a novel method for reducing training time, by selecting a subset of original features, and eliminating others from further training. We propose to pre-eliminate features whose cumulative error of corresponding best WCs exceeds a predetermined threshold value. We will discuss this method in detail for two different (both straightforward) definitions of cumulative error.

In the initial training of the WCs, each WC is evaluated based on its cumulative error of classification (*ce*). We first tested the cumulative error of a classifier defined as *ce= (false_positives + missed_examples)/q,* where *q* is the total number of examples. 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 weak classifiers to less than 3% 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.

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 weak classifier. 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* 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 WC's is heavily biased passed the *min(n,p)/q* boundary. This means that most weak classifiers 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 weak classifier efficiency is illustrated by the training results of the best strong classifier generated by our program. We deal with roughly 530,000 weak classifiers. 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 weak classifiers 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*16.67=16.5%. According to our experiments, it turns out that there are only ≈15000 weak classifiers that satisfy this requirement out of a total of ≈530,000. That means that over 97% of all initial weak classifiers are eliminated from further training. This drastically reduces training time while having little impact on the quality of the pool of available weak classifiers to choose from. The final results of the strong classifier do not suffer from this reduction of unnecessary weak classifiers as is evident from their high detection rates and low false positive rates.

An alternative definition 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 new 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. 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*min(n,p)/q - 0.01*min(n,p))/q*. This threshold leaves roughly the same number of weak classifiers after the first round of training. It reduces training time in the same way as the original triviality threshold, and produces similar quality of final classifier. Significant changes in *n* or *p* may require adjustments to the threshold.

V. AdaBoost based learning algorithm used for training

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. 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 flavour to the alternative voting AdaBoost variant described by

Wu, Rehg and Mullin in [WRM]. Both algorithms train WCs only once, at the beginning. There are several important differences in the two methods. In [WRM], 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 [WRM], 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 [WRM] 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, [WRM] considered different weights for positive and negative examples. We considered equal weights $1/q$ for the case where *ce= (false_positives + missed_examples)/q,* and weights $1/n$ and $1/p$ for the case where *nce= false_positives/n + missed_examples/p.* Finally, our variant is a single strong classifier while [WRM] described a cascaded design.

A strong classifier is obtained by running the Adaboost machine. It is a linear combination of weak classifiers. A weak classifier is constructed from a feature and a threshold. We assume that there are T weak classifiers in a strong classifier, labelled $h_1, h_2, ..., h_T$, and each of these comes with its own weight labelled $\alpha_1, \alpha_2, ... , \alpha_T$. The tested image x is passed through the succession of weak classifiers $h_1(x), h_2(x), ..., h_T(x)$, and each weak classifier 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: $\alpha_1 h_1(x)+\alpha_2 h_2(x)+ ...+\alpha_T h_T(x)>0$.

A. Training optimal weak classifiers

In the original approach [FS, VJ], examples are weighted, and weights change in the process. Weak classifiers are re-trained after selecting any of them for the strong classifier. In our algorithm, all weak classifiers 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: $g_1, g_2, ..., g_q$. The corresponding records are $(g_j, status(j))= (f(x_i), y_i)$, 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 $status(j)=y_i$.

```
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 weak classifier $h(x)=h_i(x, f, s,\theta)$. The detection rates and false positive rates of weak classifiers can also be considered output at this stage, as *(p- 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=misclassified/q=(false_pos+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 weak classifier and small weights with each poor weak classifier. Note that the algorithm corresponds to the variant with combined error *ce= (false_positives + missed_examples)/q.* If the alternate formula *nce= (false_positives/n + 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.

B. Training the best classifier

First, all weak classifiers (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 weak classifier 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 $\alpha_i$, $n$ 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 $\alpha_1$) 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 $\alpha_1 h_1(x)+\alpha_2 h_2(x)+\ldots+\alpha_T h_T(x)$ 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 weak classifiers that do not improve the cumulative error more than 1% from the trivial position. Testing each new weak classifier 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 weak classifier. Therefore it takes $O(Tqf)$ time to chose T weak classifiers. 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 [WRM]). 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 $O(Tqf)$.

VI. Experimental results

We demonstrated the significant impact of negative examples on the training process. We employ a semi-testing set of examples. After providing some initial negative examples, false positives from the semi-testing set are added to the negative example pool. This method is known as bootstrapping in some papers. It was introduced in [SP]. We have shown that this method has its limits, since the continued application of it (over fitting) starts to 'attack' the best weak classifiers and consequently starts to reduce the accuracy of the classifier.

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, as follows.

We designed a strong classifier with a record low number of weak classifiers (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 weak classifiers, and with competitive detection and false positive rates.

We have built a fast and reliable object recognizer based on small training set, consisting of 155 positive and 760 negative images. It detects back views of Honda Accords with a 98.7% detection rate and 0.4% false positive rate on the training set. 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 (320 x 240 pixels). The largest image size in the test set is 640 x 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 [VJ] and Levi & Weiss [LW]. 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 gave statistics for the number of false positives his system produced at various detection rates. At a detection rate of roughly 89% (such as our system), his system produced roughly 35 false positives. He however used a much greater number of classifiers to achieve this result. Levi and Weiss 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 2500 item training database to achieve these results.

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 4 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. This form of training was the only one capable of completely correctly identifying the entire positive set during the training phase. These results carried over to the testing phase as is evident by the relatively high detection rate.

The speed at which images are processed is measured. Images as small as 150 x 120 pixels are processed in 0.05

seconds. Images of size 170 x 227 pixels are processed in 0.18 seconds. Standard size images similar to those used in Viola and Jones [VJ] and other papers of size 320 x 240 are processed in 0.49 seconds. We consider this to be real time. It takes more and more time to process larger pictures. For example, it takes 1.93 seconds to process a picture of size 500 x 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.

VII. Conclusions

The training program can be considered as being composed of three components: an AdaBoost classifier, a Feature set, and a Training set. The AdaBoost software framework appears to be widely adopted for real time object detection. For example, Le and Satoh [LS] recently claimed that cascaded AdaBoost is about 1000 times faster than a support vector machine approach. The feature sets are not as general. Viola and Jones' set of features [VJ] was successfully used for recognizing similar types of objects such as lion faces [BC]. However, we show in [S] that the feature set for recognizing faces is completely different (practically disjoint) from the feature set for finding cars. 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 have proposed a general elimination step in the program, by introducing a threshold for the quality of a feature, to reduce training time. 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' 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' 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' 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 [BHM] 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.

VIII. References

[BHM] M. Blaschko, G. Holness, M. Mattar, D. Lisin, P. Utgoff, A. Hanson, H. Schultz, E. Riseman, M. Sieracki, W. Balch, and B. Tupper, Automatic In Situ Identification of Plankton, Proceedings of IEEE Workshop on Applications of Computer Vision, Breckenridge, Colorado, Jan. 5-7, 2005.

[FS] Y. Freund, R.E. Schapire, A decision-theoretic generalization on on-line learning and an application to boosting, *Journal of Computer and System Sciences*, 55(1):119–139, August 1997.

[LS] D. Le and S. Satoh, Fusion of Local and Global Features for Efficient Object Detection, IS & T/SPIE Symposium on Electronic Imaging, 2005.

[LW] K. Levi, Y. Weiss, Learning Object Detection from a Small Number of Examples: the Importance of Good Features, *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[LYT] H. Luo, J. Yen, D. Tretter, An Efficient Automatic Redeye Detection and Correction Algorithm, *17th IEEE International Conference on Pattern Recognition, (ICPR'04) V. 2,* Aug. 23 - 26, 2004, Cambridge UK.

[PC] V.S. Petrovic and T.F. Cootes, Analysis of Features for Rigid Structure Vehicle Type Recognition, Proc. *British Machine Vision Conf. 2004*, Vol.2, pp.587-596.

[S] M. Stojmenovic, Real time car detection in images based on an AdaBoost machine learning approach and a small training set, Proc. 12th International IEEE Workshop on Systems, Signals & Image Processing IWSSIP*,* 119-124 Chalkida, Greece, Sept. 22-24, 2005.

[SP] K. Sung, T. Poggio, Example based learning for view-based human face detection, *IEEE Trans. Pattern Anal. Mach. Intelligence, 20, 39-51, 1998.*

[TC] J. Thureson, S. Carlsson, Finding Object Categories in Cluttered Images Using Minimal Shape Prototypes, *13th Scandinavian Conference on Image Analysis SCIA,* Goteborg, Sweden, 2003.

[VJ] P. Viola, M. Jones, Robust real-time face detection, *Int. J. Computer Vision,* 57, 2, 137-154, 2004.

[WRM] J. Wu, J. Regh, and M. Mullin, Learning a Rare Event Detection Cascade by Direct Feature Selection, Proc. *Advances in Neural Information Processing Systems 16 (NIPS*2003)*, MIT Press, 2004.