Real time car detection in images based on an AdaBoost machine learning approach and a small training set

Milos Stojmenovic

SITE, University of Ottawa, Ottawa
Milos22@gmail.com

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. Our experiments indicated that the set of features used by Viola 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. We described a set of appropriate feature types for the considered car 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. This feature set was a basis for building a fast and reliable car 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, and with 89.1% detection rate and a 1.48 x 10^{-6} false positive rate on a test set of 106 images containing roughly 17.5 million tested sub windows.

1 INTRODUCTION

The goal 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, 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 manually created.

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 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 [VJ] 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 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 instances, 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 [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.

This article discusses only the feature selection for our car detection system. The AdaBoost based framework and the remaining issues are discussed in the full version of this article [S].

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 [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's real time face finding system [VJ]. AdaBoost is the concrete machine learning method that was used by Viola to implement his 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.

Viola's face finding system has been modified in literature in a number of articles. The modifications include inclusion 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 modifications proposed in literature, and adopted ideas that were considered helpful for achieving our goals.

We stress again that most of the successful applications of AdaBoost used a large training set. In Viola's 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 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.

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. In the pre-processing step, Viola [VJ] finds the sums $ii(x,y)$ of pixel intensities (or other measurements) $i(x',y')$ for all pixels (x',y') such that $x' \leq x,\ y' \leq y$. This can be done in one pass over the original image using the following recurrences: $s(x, y) = s(x, y-1) + i(x, y)$, $ii(x, y) = ii(x-1, y) + s(x, y)$ (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 $ii(x_1,y_1)+ii(x_4,y_4)-ii(x_2,y_2)-ii(x_3,y_3)$ [VJ].

3 FEATURES USED IN THE CAR RECOGNITION

A feature is a function that maps an image into a real number. Our experiments indicated that the set of features used by Viola [VJ] was inefficient for our problem. Viola's Haar wavelets were eliminated from the training procedure completely early in the implementation and testing phase since they failed to produce any usable results. Much programming effort went into incorporating them into the training framework. 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. 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's original set. This only confirmed our view that specific types of features need to be used in specific applications. Not all features are equally useful.

3.1. Redness Features

The redness features we refer to are taken from the work of [LYT]. 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 colour 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. Fig 1 shows an example of a redness feature determined by the training process.

Figure 1 – *Redness feature*

A special 'redness' colour space was formed during pre-processing to assist in the detection of red areas. This colour space was taken from [LYT], and is a one dimensional linear combination of the RGB colour space. All of the positive and negative inputs in the training set are RGB colour 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* [LYT]. The integral image computation (the technique is proposed in [VJ]) 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 image Sobel gradient mask (three pixels by three pixels) [E] 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 [LW]. The orientations of each pixel are calculated from its intensity in both directions. The orientations are divided into 6 bins so that similar orientations can be grouped together. The whole circle is divided into 6 bins. It is important to note that the orientations of the 0^0, 90^0, and 180^0 bins are critical in identifying Hondas. They are important since Hondas mainly have horizontal and vertical edges. The division of the bins which places 0^0 or 90^0 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 degrees. Note that [LW] did not mention any bin shifting, so we believe that they used non-shifted bins. They did however vary the number of bins from 4-8. Effectively the number of bins does not impact the performance much, but these boundaries are avoided. To fit all of the orientations into the $0\text{-}180^0$ range, we add 180^0 if the angle is smaller than 0^0, and subtract 180 if the angle is greater than 180^0. The effects of these transformations can be seen in Figure 2. In Figure 3, we see a Honda accord and its corresponding edge orientation image for the first bin $[-15^0,+15^0]\cup[+165^0,+195^0]$. Bin shifting contributed significantly 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.

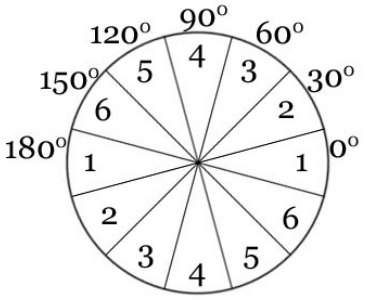

Figure 2 *Six orientation bins*

Figure 3 *Honda and corresponding horizontal edges*

Since we use 6 bins, we create 6 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 Figures 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 [LW]. 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 Figure 4. One of the most successful edge orientations was the horizontal one depicted in Figure 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.

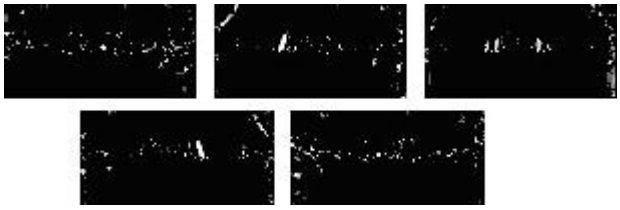

Figure 4 *Edge orientation bins [2..6] for Honda in Figure 3*

Integral images (for constant time calculation of feature values [VJ]) were created from these orientation bin images and were used in the training procedure to find sums of edge intensities. 7 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 100x50 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 weak classifiers in the strong classifier were in 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.

4 EXPERIMENTAL RESULTS

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 (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 [VJ] and Levi & Weiss [LW]. Viola'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 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 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 training procedure produced interesting results when it came to the selection of weak classifiers. Figures 5 and 6 show the best and second best weak classifiers as chosen by the training algorithm.

Figure 5 *WC1*

Figure 6 *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 5 defines this weak classifier. It detects the 45 degree 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 weak classifier that we would have chosen by hand had we tried static selection of features. The second best weak classifier was a redness feature that detected the rear stop lights of the Honda (Figure 6). 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 claim 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 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.

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 [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.

5 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.

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. 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 in [S]. In addition to defining a good feature set for Honda's, we have proposed in [S] 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 [S], 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'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's features are basically removed first before real training, with the idea that we proposed in [S] (to the best of our knowledge, our system is the first real time AdaBoost based system that recognizes an object without Viola'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. We believe that one can further develop the idea presented in [S] 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 and in [S] 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 be 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 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.

REFERENCES

[E] N. Efford, *Digital Image Processing: a practical introduction using Java* Addison Wesley, 2000.

[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.

[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 Patern 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 object detection in images based on an AdaBoost machine learning approach and a small training set, Master thesis, School of Computer Science, Carleton University, Ottawa, May 2005.

[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.

ACKNOWLEDGEMENT

This research was partially supported by the Ontario Graduate Scholarship (OGS) grant. We thank them for their support.