Toward an Efficient Implementation of a Rotation Invariant Detector Using Haar-like Features

A. L. C. Barczak[1]

[1]Institute of Information and Mathematical Sciences, Massey University, Auckland
Email: a.l.barczak@massey.ac.nz

Abstract

This paper proposes a new approach to detect rotated objects at distinct angles using the Viola-Jones detector. The approach uses different types of Haar-like features, including *twisted* features which compute areas at 45^o of rotation. A conversion algorithm takes an original feature from the classifier and computes two features to find an equivalent value for any angle. This conversion is only an approximation, but the errors are constrained and they have limited impact on the final accuracy of the classifier. We discuss the sources of errors in the computation of the Haar-like features and show that in natural images the errors are often negligible. We also show the results for a generic angle classifier that was originally trained only for one angle. The preliminary results are promising and useful in practise, but we point out future improvements to be made.

Keywords: pattern recognition, real-time object detection, Haar-like features, Viola-Jones detector.

1 Introduction

The Viola-Jones detector [1][2] has received considerable attention since its publication. It has been used mainly for face detection [3], face recognition [4] and hands detection [5]. Other uses include robot-soccer ball detection[6] and ecological applications such as wild life surveillance[7].

Due to the non-invariant nature of the Haar-like features classifiers trained with this method are often incapable of finding rotated objects. It is possible to use rotated positive examples during training, but such a monolithic approach might result in an inaccurate classifier [8]. The other possibility is to train several classifiers which specialise in certain angle intervals. Although this might achieve the accuracy required, it makes the training computationally more expensive.

We present an alternative for rotation that keeps the training time constrained to that of a single classifier and has only moderate impact on the detection time. This insight came after analysing the Viola and Jones experiments on rotational invariant classifiers for faces [8] as well as working with the extended Haar-like features proposed by Lienhart and Maydt [9]. A classifier trained on normal images can be converted to detect objects at a certain angle interval. Several classifiers can then run in parallel to detect the same object at a generic angle.

The paper is organised as follows: a brief description of the Viola-Jones methods and algorithms are presented (training and detection), including some important extensions added by other authors. Next the proposed method for detection with rotation is explained in detail. The following section presents experiments regarding face detection. Finally the conclusions point out the limitations and possible improvements on a generic rotation invariant detector using Haar-like features.

2 Haar-like Features

The Viola and Jones detector has three main characteristics that differentiates it from other methods: it uses an over-complete set of Haar-like features, an integral image to compute areas (sum of pixels) in a very efficient way and it uses Adaboost for training.

Haar-like features can be constructed in many shapes and computed in different ways. Figure 1 shows some of the most common. Viola and Jones [1] only used types 0,1,2,3 and 7 while Lienhart and Maydt [9] used all types but types 7 and 17. In this implementation we used all the 16 types. Viola and Jones used integral images to compute the related areas and the value for a given feature. Each point of the integral image can be computed once for an image. The point is the sum of all pixels that have a coordinate (x,y) smaller then the point being computed. Once this is finished, it suffices to look-up 4 values to find any rectangular sum of pixels at any position or size over the image (details in [1]). Viola and

Jones used a customised version of Adaboost, which was first created by Freund and Schapire [10] to solve machine learning problems. One of the changes made to the algorithm was the creation of many layers (called *cascades*), each one being trained by several rounds of Adaboost. To improve training speed as well as detection performance they introduced some heuristics.

Since the time they first published the algorithm many variations have been proposed, of which we cite three. The first is the empirical analysis carried out by Lienhart et. al. [11], where they compared three Adaboost algorithms called Discrete, Real and Gentle. Their experiments, which were limited to face detection, pointed to the Gentle Adaboost as the more accurate method. The second is the introduction of the Floatboost algorithm by Li et. al. [12]. Floatboost allowed them to create classifiers with a smaller error margin with fewer features per cascades. The third is the introduction of a fast heuristic to find a sub-optimal feature by McCane and Novins [13]. They reported an improvement in the training time with just minor effects on the detection phase.

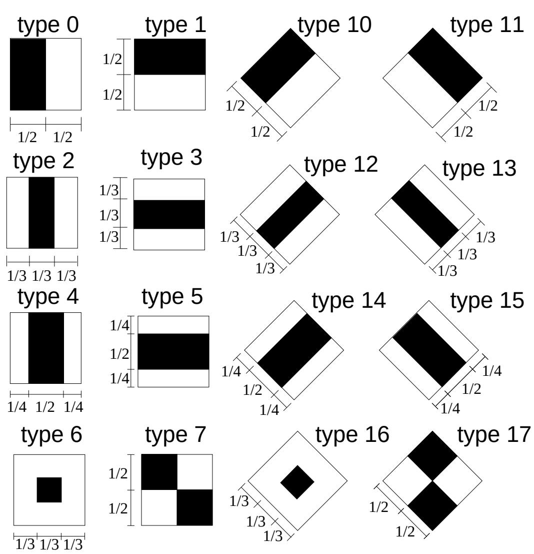

Figure 1: Haar-like Features used in this implementation.

2.1 Error sources in the computation of Haar-like features

In practise when computing the Haar-like features in digital images one cannot expect an exact value. Although the features are simple comparison between areas, ultimately a sum of pixels, there are three main sources of errors:

- the rounding up of the feature sizes due to scale changes
- the rounding up of the position of the feature in relation to a fixed point in the image
- the approximation due to some pre-processing of the image (rotation, scaling etc).

The first error source can be partially compensated by a correction factor that ensures that the areas are proportional to the theoretical definition of that particular feature. Lienhart et al. [11] suggested a correction to this problem. A correction factor is computed so that the weights of the different rectangles of a feature keep the original area ratio between them. One experiment that should be done when testing any implementation is to find the value of the features in a grayish image with equal pixel values all over the image. All features at any size and scale should yield zero.

The second error source cannot be compensated without a more complicated approach such as computing sub-pixel values for the areas. This is not usually a good approach because it loses the advantage of computing areas very rapidly with the assistance of the integral area.

The third error occurs when comparing the same image with its counterpart after a processing operation such as scaling and rotation. Anti-aliasing techniques applied when scaling images can cause variations even if the features can fit the position and the sizes perfectly. Although this error is not directly related to the features' definition or implementation itself, it is an important source of errors during the detection phase of testing. The classifier should be able to detect an object even if it is presented at different scales.

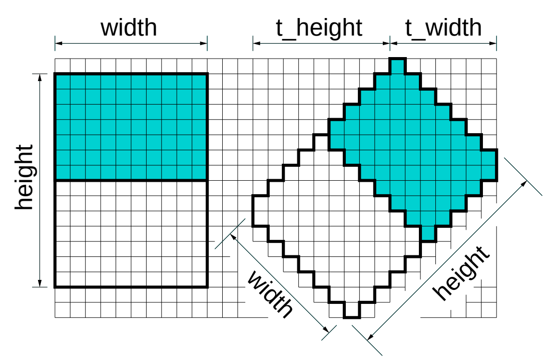

Figure 2: An approximated equivalence between a normal and a twisted Haar-like feature.

2.2 Converting features to different angles

In [8] Viola and Jones produced a classifier for multi-view faces. They had to use separate training for the various angles, except for the angles where they could find an exact equivalence between features. Using normal features one can convert an existing classifier to any angle multiple of 90^o.

Twisted (or "tilted') Haar-like features were proposed by Lienhart et al. [9] to test the hypotheses that a stronger classifier can be built if 45^o features were included in the set. The normal features and the twisted features are not mathematically equivalent in digital processing due to the fact that the twisted integral image needs slightly distorted rectangles to correctly compute an area. Figure 2 shows an example were two features, one normal and the other twisted, would compute similar areas. Notice the double pixel on two vertexes of the twisted feature. This is necessary in order to get the correct alignment so the integral image can supply the correct sum of pixels [9]. Yet another approximation has to be made when computing the width and the height of the twisted feature, as this calculation often yields a sub-pixel value.

We propose using a function of the values of two features, one normal and one twisted, to approximate the value of a given feature at any angle. This would be achieved using a weighted sum of the two equivalent features and a conversion of feature positions, features sizes and features types. Figure 3 shows how this conversion would be achieved. For an angle α the new features would have to be positioned on a new kernel, larger in size to accommodate the rotation of the twisted feature. We call this approach *pair of equivalent features* (PEF). To implement a simple code that allowed us to test this hypothesis we used a weighted average of the normal and twisted features to find the PEF value. For example, for angles between 0^o and 45^o formula 1 applies. For angles larger than 45^o there is also a feature type change to be made. For example, the PEF of a type 0 feature for angles between 0^o and 45^o will be a normal feature of type 0 and a twisted feature of type 10. If the angle is between 45^o and 90^o, the PEF will be a normal feature of type 1 and a twisted feature of type 10 (figure 4). For other angles the PEF may be computed from other pairs, in some cases involving a change of sign.

$$V = V_{normal}.(45^o - \alpha)/45^o + V_{twisted}.\alpha/45^o \quad (1)$$

Where: V_{normal} is the Value for the normal feature, $V_{twisted}$ is the value for the twisted feature and V is the weighted average that depends on the angle α (between 0^o and 45^o).

2.3 Measuring errors on PEFs

A Haar-like feature value will depend on the distribution of the pixels in the area where the kernel is applied. The maximum value for many types of

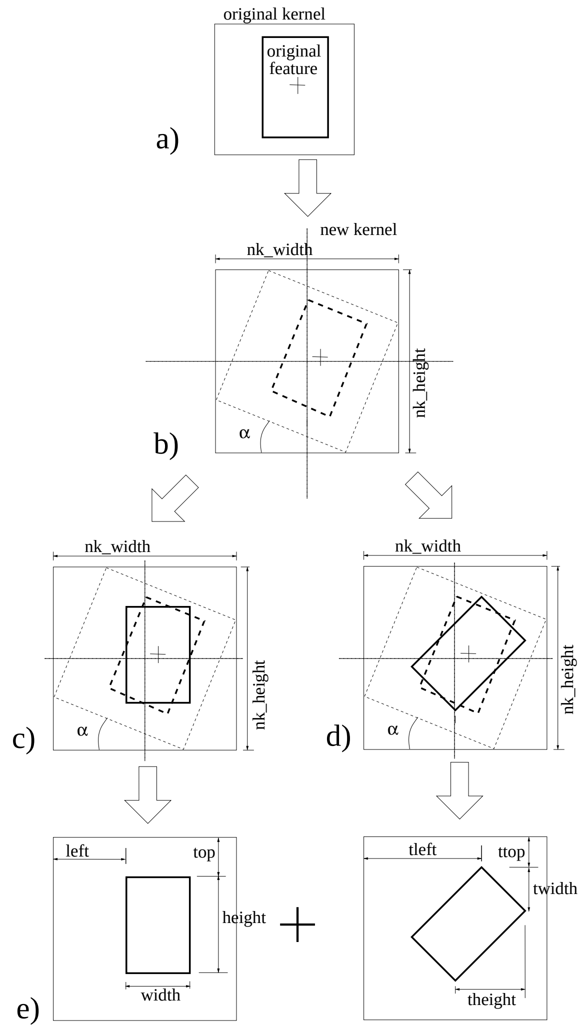

Figure 3: a) The original feature positioned in the original kernel. b) The new kernel size and the rotation angle are shown. c) The position for the new normal feature is computed. d) The size and the position is computed for the twisted feature. e) The resulting Pair of Equivalent Features (PEF).

Haar-like features occurs when an edge is located in the middle of the feature. In order to maximise the absolute value of the feature it is also necessary that all the values of the pixels on one side of the edge are zeros and all the other pixel values are maximum (for example, 255 in a grey scale image). From this point lets call the maximum value of a feature at a generic scale to be MAX, so that any feature value can vary from -MAX to MAX. Lets suppose that we could compute a *type 0* Haar-like feature at an angle of 22.5^o over an edge presented by the image. In this special case we assume that the ratio width/height is approximately 2:5. The theoretical value for the figure 5-a is MAX. The PEF value is abs(MAX/2) (figure 5-b), as both the normal feature and the twisted feature will yield the same value MAX/2. For figure 6-a the value of the feature would be zero. For the normal

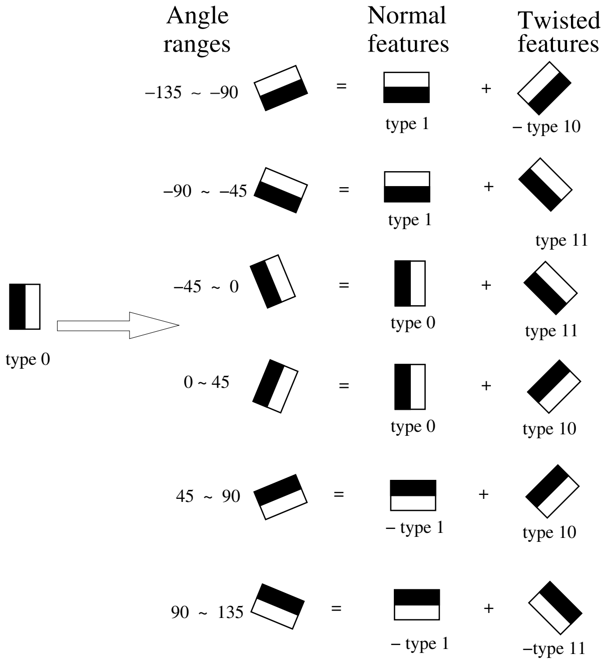

Figure 4: An example of PEFs for a type 0 feature considering a number of angle ranges.

features is MAX/8 and for the twisted features it is -MAX/8 (figure 6-b). Therefore the PEF value is zero, which is the required result. In the extreme cases an error of about 50% in relation to MAX is expected. Slight variations on these errors might be expected if the features are either more square or longer in shape.

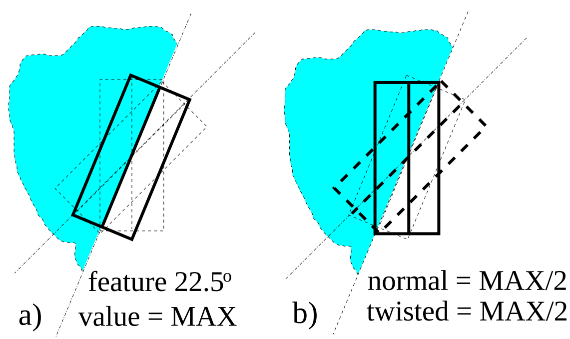

Figure 5: Case where the value of the 22.5^o feature should be MAX.

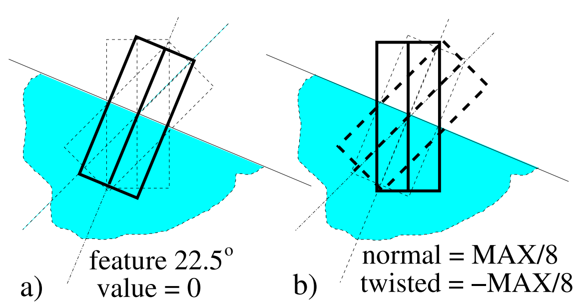

Figure 6: Case where the value of the 22.5^o feature should be zero.

In order to assess the impact of the approximation, an experiment was carried out using several natural images as well as binary images where the edges would call for the maximum value of the features. The images were rotated to angle α. Features at all possible scales and positions where computed in the original image. The PEF for each of those features were also computed. The error was calculated as a percentage of the maximum possible value (MAX) for that feature type at that scale (F is the original feature value and V is the PEF value for that angle):

$$error = \frac{\mid F - V \mid}{MAX} \tag{2}$$

We present here the results for two images, the first frame of the Akiyo sequence and a Chessboard (figure 7). The feature size was 20x20 pixels on a kernel of the same size. The kernel size for the PEF was 34x34 pixels. Figures 8 and 9 show the maximum errors for both images and as expected the largest errors were closer to the region of 22.5^o. When assessing features at 45^o the errors decrease almost to the same levels of those at 1^o. This indicates that the twisted features are successfully representing a normal feature despite the rotation.

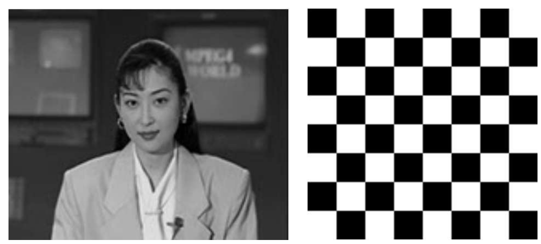

Figure 7: First frame of Akiyo sequence and the chessboard images.

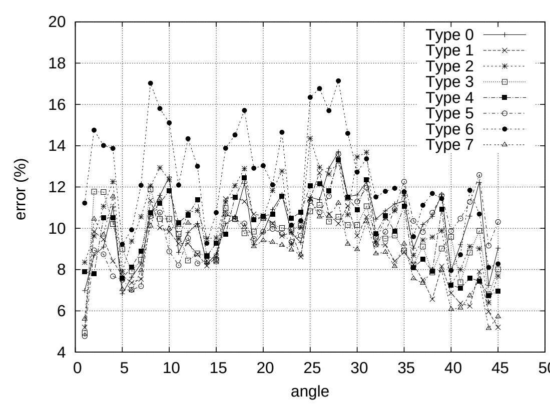

Figure 8: Maximum error vs angle (Akiyo).

The maximum error for Akiyo was 17% for type 6 feature. For the chessboard image the maximum error was 47% for type 7 features. The maximum error in this case almost reached the theoretical value of 50%. The question now is if these errors would cause significant problems during the classification process. The next section describes

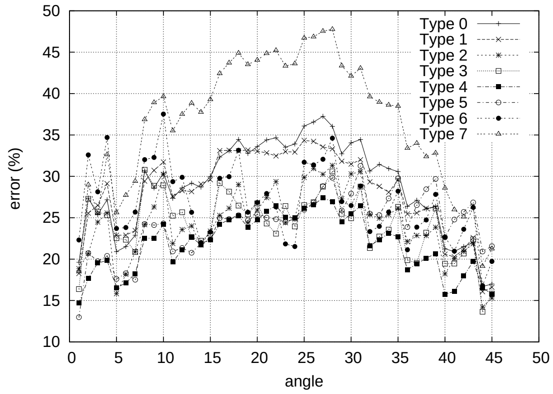

Figure 9: Maximum error vs angle (Chessboard).

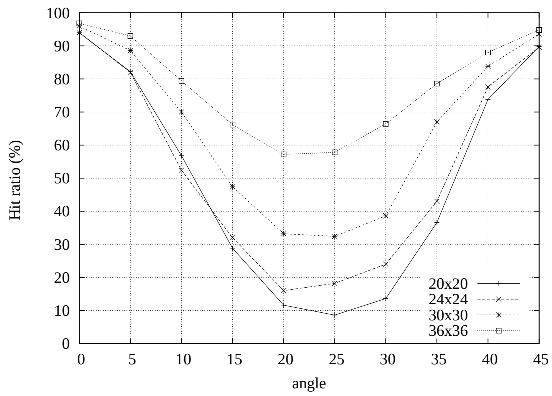

Figure 10: Hit ratio vs angle using 16 cascades.

an experiment and the results that answer this question.

3 Using PEFs in a cascade

Now that it is possible to convert any classifier into any angle by finding its PEF, we decided to test real classifiers and compare the results at different angles. We trained several classifiers using 1000 frontal faces from FERET and reserving other 500 frontal faces for the test set. The first set of classifiers were trained using kernels of different sizes (20x20, 24x24, 30x30 and 36x36), each with 16 cascades. Initial tests were done using 12 to 16 cascades to measure the effects of the PEF at various angles. Figure 10 shows the hit ratio for the classifiers using 16 cascades. Clearly the larger kernel presented the best result, while the smaller kernel hardly worked at 20 or 25^o. Notice however how the twisted features converted the classifier for 45^o perfectly for any kernel size. This result shows that for small kernels the errors of the PEFs are too large to be effective.

We show more details on the 36x36 kernel, which presented results good enough to be used in practise. Figure 11 and figure 12 show the variation of the hit ratio and false detection rates for this kernel size. A hit is considered when it is within a 10%

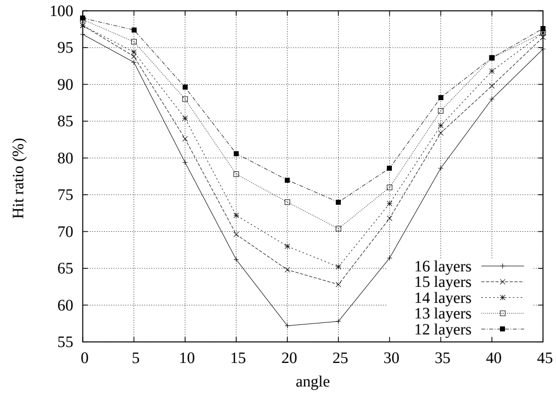

Figure 11: Hit ratio vs angle, kernel 36x36.

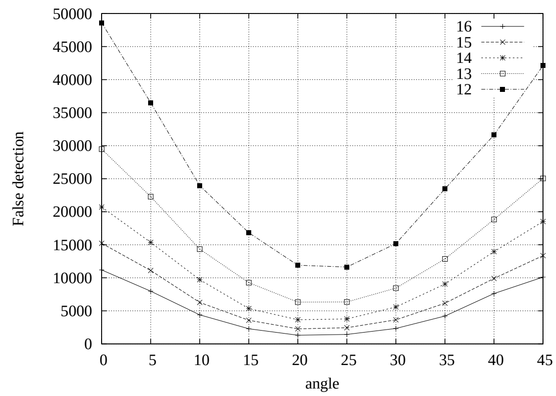

Figure 12: False detection rate vs angle, kernel 36x36.

tolerance area around the face. The false detection was computed conservatively as the total number of hits outside the tolerance area. On the worst part of the curve (at 22.5^o) the 16 cascades yielded 55% and the 12 cascades 75% hit ratio. The false detection was much higher for angles close to 0^o or 45^o than it was for 22.5^o region. This indicates that a correction factor could be applied so the classifier would effectively work at the same equivalent point of a ROC curve for all angles.

A correction factor could be applied in two ways. One option would be to apply a correction factor directly to the computation of each PEF, because the value of each PEF will typically be lower than the actual value for the rotated feature. This however would imply in searching for different correction factors as each feature has different shapes. Instead we decided to apply a correction factor for the final threshold of each cascade. This correction factor has to be a function of the angle, so the false detection levels can be adjusted properly. Based on the shape of the hit ratio curve, we found empirically that the following formula is suitable:

$$factor(\alpha) = \frac{(1-fc)cos(8\alpha) + (1+fc)}{2} \qquad (3)$$

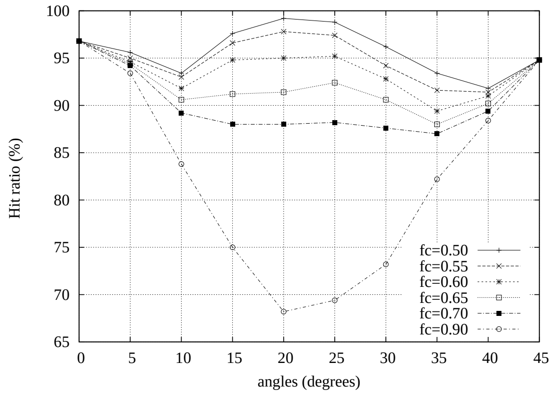

Figure 13: The classifier 36x36 with various correction factors.

Where: *fc* is a constant between 0.3 and 0.9. Notice that at 0^o or 45^o the correction factor is 1 for any *fc*.
The results of the application of the correction factor can be seen in figure 13. Notice that an *fc* of 0.5 lowered the threshold by too much, compromising the false detection rate. To achieve a balance between the false detections and hit ratios an *fc* of 0.65 was the best value. An alternative to a correction factor would be to use a different number of cascades for each angle.

4 Conclusions and Future Work

We proposed and successfully demonstrated a new approach for a rotational invariant Viola-Jones detector. The method allows the conversion of a trained classifier for any angle, so rotated objects can be detected. The impact on the speed of the original Viola-Jones detector is minimum. An empirical correction factor was introduced as a mean for balancing the classifier accuracy for any angle. We can improve on the proposed method by implementing a third integral image that computes features around 22.5^o, so the errors of PEFs for longer features would be minimised. Further evaluation has to be made with other objects such as hands to verify the viability for gesture recognition applications.

5 Acknowledgements

Portions of the research in this paper used the *Colour FERET* database of facial images collected under the *FERET* program. The author would like to thank Dr. Chris Messom and Dr. Martin Jonhson for their valuable suggestions and comments regarding this work.

References

[1] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *CVPR01*, pp. I:511–518, IEEE, December 2001.

[2] M. J. Jones and P. Viola, "Robust real-time object detection," Tech. Rep. CRL-2001-1, Hewlett Packard Laboratories, Feb. 25 2001.

[3] P. Viola and M. Jones, "Robust Real-Time face detection," in *Proceedings of the Eighth International Conference On Computer Vision (ICCV-01)*, (Los Alamitos, CA), pp. 747–747, IEEE Computer Society, July 9–12 2001.

[4] G. Guo and H. Zhang, "Boosting for fast face recognition," Tech. Rep. MSR-TR-2001-16, Microsoft Research (MSR), Feb. 2001.

[5] M. Kolsch and M. Turk, "Analysis of rotational robustness of hand detection with a viola-jones detector," in *ICPR04*, pp. III: 107–110, 2004.

[6] S. Mitri, K. Pervlz, H. Surmann, and A. Nchter, "Fast color-independent ball detection for mobile robots," *Mechatronics and Robotics*, pp. 900–905, September 2004.

[7] T. Burghardt, B. Thomas, P. J. Barham, and J. Calic, "Automated visual recognition of individual african penguins," in *Fifth International Penguin Conference*, (Ushuaia, Tierra del Fuego, Argentina), September 2004.

[8] M. J. Jones and P. Viola, "Fast multi-view face detection," Tech. Rep. TR2003-96, MERL, 2003.

[9] R. Lienhart and J. Maydt, "An extended set of haar-like features for rapid object detection," in *ICIP02*, pp. I: 900–903, September 2002.

[10] Y. Freund and R. E. Schapire, "A short introduction to boosting," *Journal of Jap. Society for Art. Intell.*, vol. 14, no. 5, pp. 771–780, 1999.

[11] R. Lienhart, A. Kuranov, and V. Pisarevsky, "Empirical analysis of detection cascades of boosted classifiers for rapid object detection," in *DAGM03*, (Madgeburg, Germany), pp. 297–304, September 2003.

[12] S. Z. Li and Z. Zhang, "Floatboost learning and statistical face detection," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 26, pp. 1112– 1123, September 2004.

[13] B. McCane and K. Novins, "On training cascade face detectors," in *Image and Vision Computing New Zealand*, (Palmerston North), pp. 239–244, 2003.