

ROBUST HAND TRACKING IN LOW-RESOLUTION VIDEO SEQUENCES

Nyan Bo Bo¹

Matthew N. Dailey²

Bunyarit Uyyanonvara¹

¹Sirindhorn International Institute of Technology

Thammasat University

Klong Luang, Pathumthani, Thailand

email: {nyanbobo, bunyarit}@siit.tu.ac.th

²Computer Science and Information Management

Asian Institute of Technology

Klong Luang, Pathumthani, Thailand

email: mdailey@ait.ac.th

ABSTRACT

Automatic detection and tracking of human hands in video imagery has many applications. While some success has been achieved in human-computer interaction applications, hand tracking would also be extremely useful in security systems, where it could help the system to understand and predict human actions, intentions, and goals. We have designed and implemented a prototype hand tracking system, which is able to track the hands of moving humans in low resolution video sequences. Our system uses grayscale appearance and skin color information to classify image sub-windows as either containing or not containing a human hand. The prototype's performance is quite promising, detecting nearly all visible hands in a test sequence with a relatively low error rate. In future work we plan to improve the prototype and explore methods for interpreting human gestures in surveillance video.

KEY WORDS

Computational Intelligence, Hand Tracking, Computer Vision, Image Processing.

1 Introduction

Just as monitoring a person's face is key to effective communication with that person, monitoring a person's hands is key to understanding what that person is *doing*. Towards endowing intelligent systems with the ability to monitor and understand human behavior, in this paper, we propose and evaluate an object detection system capable of finding hands in video imagery without requiring any close-up, high-resolution analysis.

Human hand detection in video sequences would enable several useful applications. We are primarily interested in security applications, where human figures in the scene could be at a fairly large distance from the camera and therefore would appear at a fairly low resolution. In security applications, it would be useful to track people in the scene and perform automated analysis of their actions, e.g., by determining if they are walking, running, punching someone, and so on. The key to many of these behaviors is the ability to track the hand.

Over the last 15 years, the problem of hand tracking

has become an attractive area for research [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]. Many early hand tracking systems relied on uncluttered static backgrounds, high resolution images and manual initialization. Though state of the art systems are becoming more robust, most have been targeted towards sign language and gesture recognition, where the assumption of high-resolution imagery holds. In these cases, motion information between two consecutive frames makes hand tracking easier since most of the body parts will be stationary except the hands. In the general case, motion information is less useful for tracking hands, since all of the body parts may be moving from frame to frame.

Some systems like Pfinder [5] imitate the way humans look for hand in images — instead of directly detecting hands in the image, Pfinder tries to find human bodies first and then finds body parts such as hands. However, finding the human in an image is itself a difficult problem, so in our work we attempt to bypass this problem entirely by finding hands without any attempt to model the human context.

Some hand tracking systems use skin color information to segment hands from the background and then track segmented hands over a frame sequence. The face and hand tracking system for sign language recognition by Soontranon and Chalidabhongse [7] first segments the image into skin and non-skin regions using a Gaussian model for skin pixels in CbCr space. Then they obtain the connected components of the skin pixel image and track the connected components from frame to frame assuming translation only. They use face detection to prevent false positives due to skin pixels on the face. A similar approach is used by [10, 12] to detect and track hands for human-robot interaction and human-computer interaction.

In recent years, face detection has been one of the great success stories in computer vision; new techniques are achieving excellent performance in real time [13, 14, 15, 16, 17]. Hand tracking is more difficult, however, since the face contains structural information such as eyes, mouth and so on, even in low resolution images. Much less structural information, except the gaps between fingers, is present in hand image, however, so hands look more or less like elliptical blobs of skin in low resolution images.

Barreto and colleagues [9] applied the Viola and Jones face detector [15] (later improved by Lienhart and Maydt [18]) to upright hand detection. Ong and colleagues

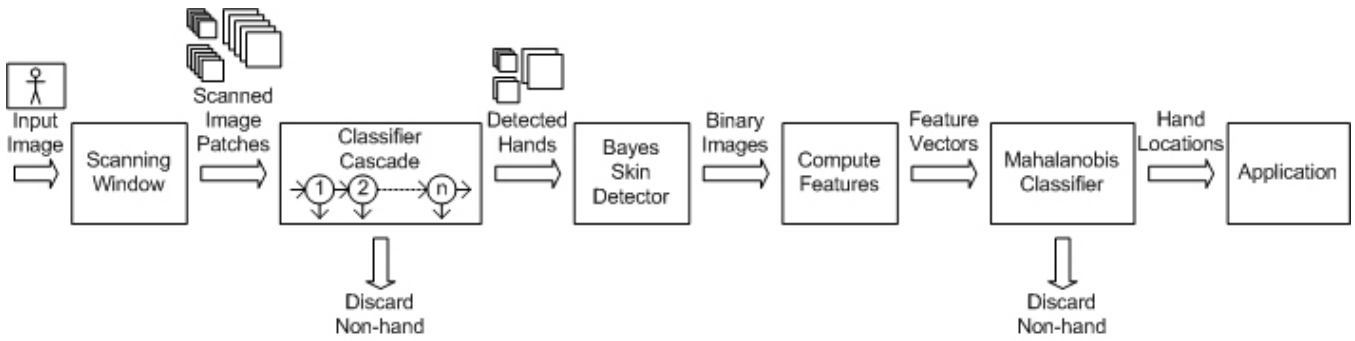


Figure 1. Hand detection system architecture.

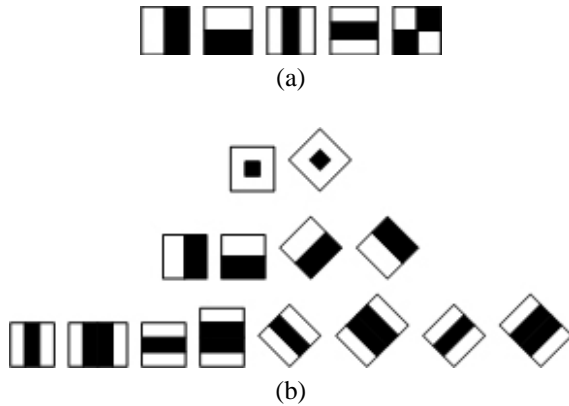


Figure 2. Haar-like features used to construct weak classifiers in the boosted classifier cascade. (a) Viola and Jones features. (b) Lienhart and Maydt features.

[8] use a similar approach, but instead of a linear cascade to detect a single hand posture, they construct a tree-structured classifier to not only detect hands but also to classify hand posture.

Thus far, hand detection systems using general object recognition techniques like those from the face detection literature are tailored to detecting distinct sets of specific gestures. Our goal, however, is to detect and track multiple hands in arbitrary postures in relatively low-resolution video sequences. Our approach uses grayscale appearance information to reject most non-hand image regions very quickly, then uses skin color information to reject the remaining non-hand image patches. We have conducted preliminary experiments with the proposed system, and the results are encouraging. In the rest of the paper, we describe the system and its evaluation in detail.

2 Hand Detection

Figure 1 depicts our hand detection system’s architecture schematically. A scan window sweeps over the input image at multiple scales. Each resulting image patch is classified as either hand or non-hand by a boosted classifier cascade

Algorithm TRAIN-CASCADE

Given: \mathcal{P}_0 , a set of k positive examples
 \mathcal{N}_0 , an initial set of k negative examples
 α_p , the desired true positive rate per stage
 α_f , the desired false positive rate per stage
Returns: C , a cascade of ensemble classifiers
 $C \leftarrow H_0 \leftarrow \text{ADABOOST}(\mathcal{P}_0, \mathcal{N}_0, \alpha_p, \alpha_f)$
 $i \leftarrow 0$
Repeat:
 Test C on new, known-to-be-negative examples
 $\mathcal{N}_i \leftarrow$ The top k false positives from test
 $\mathcal{P}_i \leftarrow (\mathcal{P}_{i-1} - \text{positives dropped by } H_{i-1})$
 $H_i \leftarrow \text{ADABOOST}(\mathcal{P}_i, \mathcal{N}_i, \alpha_p, \alpha_f)$
 $C \leftarrow C$ with H_i appended to the cascade
 $i \leftarrow i + 1$
Until performance is “good enough”
Return C

Figure 3. Cascade training algorithm. The algorithm utilizes the ADABOOST routine, which, given a training set $\langle \mathcal{P}, \mathcal{N} \rangle$, finds a combination of weak threshold classifiers obtaining a true positive rate of at least α_p and a false positive rate at most α_f on that training set.

[15, 18]. To further reduce false positive detections using a priori knowledge of hand color and geometry, each positive detection from the classifier cascade is further processed by a skin detection module, a feature extractor, and a simple classifier based on Mahalanobis distance to the “average” hand. We describe each of the modules in turn.

2.1 Boosted classifier cascade

The core of our object detection system is the cascade of boosted classifiers originally proposed by Viola and Jones [19, 15] and later modified by Lienhart and Maydt [18, 20]. The cascade reduces processing time while preserving classifier accuracy through the use of a sequence of classifiers tuned for reasonably low false positive rates but extremely high detection rates. The cascade quickly rejects most de-

tection windows unlikely to contain the object of interest and spends more compute time on the detection windows most likely to contain the object of interest.

Each stage in the cascade uses the “boosting” ensemble learning method [21] to induce a strong nonlinear classification rule that is a linear combination of the “votes” of multiple weak threshold classifiers, each considering the output of a single Haar wavelet-like filter at a fixed location in the detection window. Viola and Jones’ original method [19] uses Freund and Shapire’s “discrete” Adaboost algorithm [21] with a set of five types of Haar-like features for the weak threshold classifiers (Figure 2(a)). Lienhart and colleagues’ method [18, 20] uses the “gentle” Adaboost algorithm and additional Haar-like features (Figure 2(b)). Here we refer to both types of classifier as a *V&J cascade*.

The first step in constructing a V&J cascade for object detection is to obtain a large training set of images containing or not containing the object of interest. We then extract an initial set \mathcal{P}_0 of positive detection windows and an initial set \mathcal{N}_0 of negative detection windows and execute the procedure TRAIN-CASCADE, detailed in Figure 3.

2.2 Bayesian skin detector

Our skin detector is a Bayesian maximum likelihood classifier based on color histograms [22, 23]. The classifier estimates the class $S \in \{skin, nonskin\}$ of a single pixel based only on its observed color x measured in some color space. This simple method is extremely efficient and surprisingly effective. We let

$$\hat{s} = \arg \max_s P(X = x | S = s),$$

where the likelihood $P(X | S)$ is modeled by a color histogram estimated from training data we obtained in a pilot study. Our color histograms have two dimensions, namely the hue and saturation axes of the HSV color space, which we quantize into $16^2 = 256$ bins.

2.3 Feature extraction

The output of the Bayesian skin detector is a binary image in which one value represents skin pixels and the other value represents non-skin pixels. We have found that a few simple features extracted from this binary image allow surprisingly accurate classification. The particular features we extract are:

1. The *area* (in pixels) of the largest connected component of skin pixels.
2. The length of the *perimeter* of the largest connected component of skin pixels.
3. The *eccentricity* of the largest connected component of skin pixels.
4. The number of pixels in the largest skin component intersecting the detection window *boundary*.

Clearly, when the area feature is especially large or especially small, the given image patch is unlikely to contain a hand. The perimeter length and eccentricity features provide additional information about the shape of the detected skin “blob.” Finally, since a properly detected hand will only intersect the boundary of the detection window at the wrist, the boundary feature provides information about how wrist-like the boundary is.

When combined with a reasonable classifier, these four features are sufficient to correct most of the V&J cascade’s mistakes. We next describe our classifier.

2.4 Mahalanobis classifier

Given a feature vector \mathbf{x} consisting of the area, perimeter, eccentricity, and boundary features, we must determine whether \mathbf{x} represents a true hand or a false positive. We tackle this problem by applying a threshold θ to the dissimilarity of the given feature vector \mathbf{x} from the mean feature vector $\boldsymbol{\mu}$. Our dissimilarity measure is the Mahalanobis distance

$$d(\mathbf{x}) = (\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}).$$

Here the mean hand feature vector $\boldsymbol{\mu}$, covariance matrix Σ , and distance threshold θ are estimated from a training set.

Once we obtain a final classification for each possible detection window, the positively detected hands could then be forwarded to another component in an integrated application, for example a gesture recognition module. In the current paper we simply evaluate the efficacy of the proposed algorithm on a series of video segments.

3 Experimental Methods

Here we describe an experiment in which we captured video sequences of humans walking in an indoor environment then evaluated the hand detection system on those video sequences.

3.1 Data acquisition

For purposes of training and testing the hand detection system, we captured four video sequences of four different people walking in and out of a moderately cluttered laboratory environment. The video sequences were captured at 15 frames per second with an IEEE 1394 Web camera, and each sequence lasted approximately three minutes.

After video acquisition, we manually located all visible hands in every image of all four sequences, for a total of 2246 hands. We designated the first 2000 as training examples and reserved the remaining 246 for testing.

Our criteria for positioning the detection window on the hand was that that the hand should be roughly at the center of the window while taking up about 50% of the pixel area of selection window (see Figure 4 for examples).

As already described, the cascade learning algorithm, at step i , requires a set \mathcal{N}_i of negative examples that do

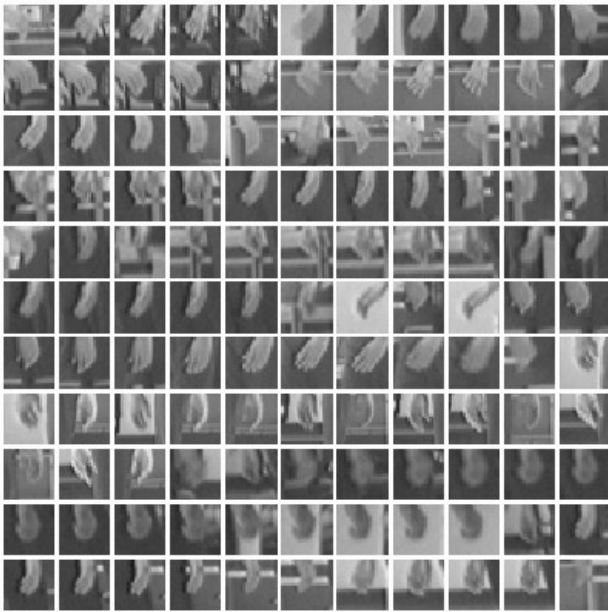


Figure 4. Example training images scaled to 24×24 .

not contain hands for training. For this purpose, we randomly selected eight frames from the training data that did not contain and hands. The OpenCV implementation of the V&J cascade (see below) scans these eight images to produce new negative examples for training at each stage.

3.2 Boosted classifier training

To train the classifier cascade, we used Linehart and Maydt's approach [18] as implemented in OpenCV [24]. With 2000 positive hand examples and 8 background images for negative examples, we trained 30 stages using GentleBoost, $\alpha_p = 0.995$, and $\alpha_f = 0.5$. This took two weeks on a 3 GHz Pentium 4 PC with 1 GB of RAM.

We found that the classifier's training set performance peaked at 26 stages, so we used only the first 26 stages, comprising 763 weak classifiers, in subsequent analysis.

3.3 Skin detector training

To train the skin detector, we selected 10 images containing one or more humans from a set of independent video sequences captured under various lighting conditions at several different locations. We manually marked the skin pixels in each image, extracted the hue (H) and saturation (S) of the resulting 70,475 skin and 1,203,094 non-skin pixels, quantized the H-S values into 16 bins, and constructed two 2D histograms: one for skin pixels, and one for non-skin pixels.



Figure 5. ROC curve between true positive rate and false positive rate

3.4 Gathering data to train the post-processor

We ran OpenCV's performance testing utility, which had been modified to produce true positive and false positive image patches, on all labeled images from the training set with a detection window scale factor of 1.1. The resulting image patches were scaled to the standard size of 24×24 . Then, we randomly selected 1000 true positive and 1000 false positive image patches for the training process of post-processor system.

3.5 Parameter estimation for the post-processor

To estimate the parameters of the Mahalanobis distance-based post processor, we applied skin detection to the 2000 training patches, eliminated all connected skin components smaller than 36 pixels, and filled in holes with a morphological opening operator. We then extracted features (area, eccentricity, perimeter, and boundary pixel count) for the largest connected skin blob in each of the 2000 patches. We randomly split the true positives into two groups, using the first 500 for mean and covariance calculation and using the remaining 500 to determine the best Mahalanobis distance threshold. Using the ROC curve in Figure 5 to explore the tradeoff between true positives and false positives, we selected the point where the false positive rate was as low as possible (18%) while maintaining a 100% true positive rate.

4 Results

To analyse the performance of our system, we selected 4 frames from a video sequence which had never been used in the training process, and fed it to our system. We manually classified the detections at each stage of the system as a false positive or true positive. We found that the classifier cascade, by itself, performed relatively poorly, but that the post processing system was extremely effective in eliminating false positives produced by the classifier cascade.

Image	Number of True Positives	Number of False Positives
1	1	22
2	2	35
3	1	19
4	1	18

Table 1. Test results without post-processing

Image	Number of True Positives	Number of False Positives
1	1	2
2	2	5
3	1	6
4	1	2

Table 2. Test results with post-processing

Table 1 shows that without the post-processing system, the false positive rate is too high for practical application. Table 2, on the other hand, shows that when we add post processing to our system, the false positive rate decreases rapidly.

Image 1 and 2 in Figure 6 illustrate example detection results from our complete system. In both images, we observe that regions on the person’s head, especially in the chin and neck areas, are detected as hands by the system. The most probable reason for this kind of false positive is that we did not include such image patches as negative examples during training of the boosted classifier cascade. We only used background images to generate negative examples. Unfortunately, not even the post-processing system can reject this kind of false positive because the shape of the skin blobs in those regions are very similar to the shape of skin blobs in patches actually containing hands. We believe that including human body parts other than hands as negative training examples will eliminate these types of false positives.

5 Conclusion

From the literature, we know that hand trackers incorporating Adaboost and Haar-like features perform quite well in applications like sign language recognition, in which the video is relatively high resolution and the hand gestures are rather constrained. We have found, on the other hand, that the approach suffers from high false positive rates when applied to less constrained scenes. However, as we have shown in this paper, these limitations can be reduced by incorporating domain-specific knowledge in the form of a post processing system.

One important limitation of the work described here is that both the training and testing data were captured in the same simple lab environment. The reported performance is therefore optimistic. In future experiments, we plan to test

the approach more rigorously by training on a wider variety of scenes and testing on a completely novel scene.

In any case, the system is a good starting point for a practically applicable low resolution, real-time hand tracker. The current results are encouraging, and with some improvement we will be able to integrate it with a complete gesture recognition or activity recognition system.

Acknowledgments

We thank the members of the Image and Vision Computing Laboratory at the Sirindhorn International Institute of Technology for participating in our data collection efforts. This research was partially supported by Thailand Research Fund grant MRG4780209 to MND.

References

- [1] J. M. Rehg and T. Kanade. Visual tracking of high DOF articulated structures: An application to human hand tracking. In *Third European Conference on Computer Vision*, pages 35–46, 1994.
- [2] J. Segen and S. Kumar. Human-computer interaction using gesture recognition and 3D hand tracking. In *Proceedings of the IEEE International Conference on Image Processing*, pages 188–192, 1998.
- [3] S. Ahmad. A usable real-time 3D hand tracker. In *Proceedings of the 28th IEEE Asilomar Conference on Signals, Systems and Computers*, pages 1257–1261, 1995.
- [4] J. Triesch and C. von der Malsburg. A system for person-independent hand posture recognition against complex backgrounds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(12), 2001.
- [5] C. R. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfunder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.
- [6] M. Kölsch and M. Turk. Robust hand detection. In *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition*, 2004.
- [7] N. Soontranon, S. Aramvith, and T. H. Chalidabhongse. Face and hands localization and tracking for sign language recognition. In *International Symposium on Communications and Information Technologies*, pages 1246–1251, 2004.
- [8] Eng-Jon Ong and R. Bowden. A boosted classifier tree for hand shape detection. In *Proceedings of the Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 889–894, 2004.

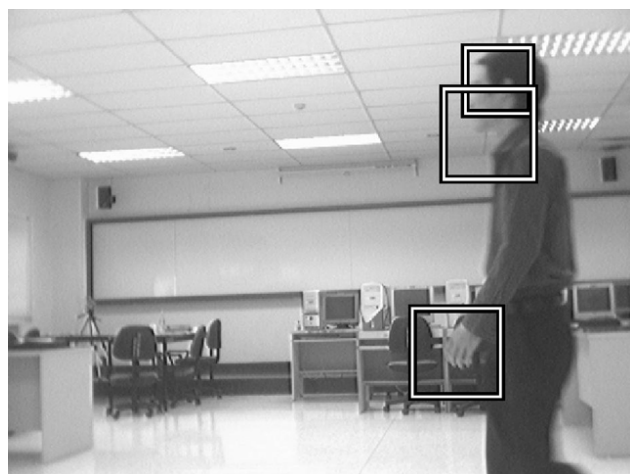


Image 1

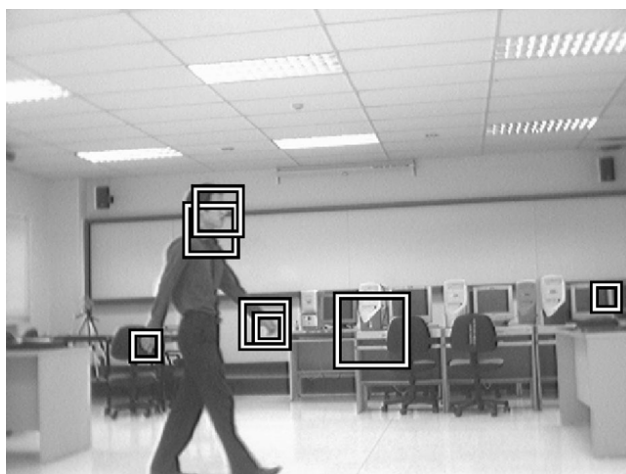


Image 2

Figure 6. Preliminary results on test data from the hand detector.

- [9] J. Barreto, P. Menezes, and J. Dias. Human-robot interaction based on haar-like features and eigenfaces. In *Proceedings of the 2004 IEEE Conference on Robotics and Automation*, pages 1888–1893, 2004.
- [10] J. Wachs, H. Stern, Y. Edan, M. Gillam, C. Feied, M. Smith, and J. Handler. A real-time hand gesture system based on evolutionary search. In *Genetic and Evolutionary Computation Conference*, 2005.
- [11] S. Wagner, B. Alefs, and C. Picus. Framework for a portable gesture interface. In *Proceeding of the 7th International Conference on Automatic Face and Gesture Recognition*, pages 58–63, 2006.
- [12] Kye Kyung Kim, Keun Chang Kwak, and Su Young Chi. Gesture analysis for human-robot interaction. In *The 8th International Conference on Advanced Communication Technology*, pages 1824–1827, 2006.
- [13] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(20):23–28, 1998.
- [14] D. Roth, M. Yang, and N. Ahuja. A SNoW-based face detector. In *Advances in Neural Information Processing Systems (NIPS)*, pages 855–861, 2000.
- [15] P. A. Viola and M. J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.
- [16] I. Fasel, B. Fortenberry, and J. Movellan. A generative framework for real time object detection and classification. *Computer Vision and Image Understanding*, 98:182–210, 2005.
- [17] E. Hjelmås and B. K. Low. Face detection: A survey. *Computer Vision and Image Understanding*, pages 236–274, 2001.
- [18] R. Lienhart and J. Maydt. An extended set of Haar-like features for rapid object detection. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, volume 1, pages 900–903, 2002.
- [19] P. A. Viola and M. J. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 511–518, 2001.
- [20] R. Lienhart, A. Kuranov, and V. Pisarevsky. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. Technical report, Microprocessor Research Lab, Intel Labs, 2002.
- [21] Y. Freund and R. E. Shapire. A decision-theoretic generalization of online learning and an application to boosting. *Journal of Computer and System Sciences*, 5(1):119–139, 1997.
- [22] B. D. Zarit, B. J. Super, and F. K. H. Quek. Comparison of five color models in skin pixel classification. In *International Workshop on Recognition, Analysis and Tracking of Faces and Gestures in Real-Time Systems*, pages 58–63, 1999.
- [23] M. J. Jones and J. M. Rehg. Statistical color models with application to skin detection. *International Journal of Computer Vision*, 46(1):81–96, 2002.
- [24] Intel Corporation. OpenCV Computer Vision Library (software). Open source software available at <http://sourceforge.net/projects/opencv/>.