Real-Time Face Detection And Motorized Tracking Using ScicosLab and SMCube On SoC's

Wing Jack Lee*, Kok Yew Ng*†, Chin Luh Tan‡ and Chee Pin Tan*
*School of Engineering, Monash University Malaysia, 47500 Bandar Sunway, Malaysia
Emails: wjlee24@student.monash.edu,nkymark@ieee.org,tan.chee.pin@monash.edu
†Vehicular Systems, Department of Electrical Engineering, Linköping University, SE-581 83, Sweden
‡Trity Technologies, Malaysia. Email: chinluh@tritytech.com

***Abstract*—This paper presents a method for real-time detection and tracking of the human face. This is achieved using the Raspberry Pi microcomputer and the Easylab microcontroller as the main hardware with a camera mounted on servomotors for continuous image feed-in. Real-time face detection is performed using Haar-feature classifiers and ScicosLab in the Raspberry Pi. Then, the Easylab is responsible for face tracking, keeping the face in the middle of the frame through a pair of servomotors that control the horizontal and vertical movements of the camera. The servomotors are in turn controlled based on the state-diagrams designed using SMCube in the EasyLab. The methodology is verified via practical experimentation.**

I. Introduction

In recent times, many applications that use image processing for face detection and tracking have been developed for various purposes. Generally, it involves the recognition of the human face from other objects in the image frame and then to continuously reposition the location of the face, preferably at the centre within the frame. Basically, the face detection algorithm can be implemented using different methods with and/or without colour segmentation or even unique object identifiers. The general operation is that the algorithm will filter every frame of the image and scan pixel by pixel for a match, and depending on the algorithm used, the processing may be quite time and memory consuming.

Colour segmentation for face localisation exploits the differences of the human skin colour from the external environment. By doing so, pixels that match the designated skin colour thresholds are considered to be a match to that of a human face [1]. However, since the detection is solely based on specific coloured pixels, the detection time is increased for different facial positions or poses. To overcome the issue of the wide range of colour samples of various individuals, the Gaussian Mixture Model (GMM) and the machine learning algorithm Adaboost can be used to improve reliability and accuracy of detection [2]. By plotting a probability distribution function of all coloured pixels within the image frame and then using a predetermined threshold for the skin colour pixel, it can be determined statistically if the skin colour can be classified and declared as a match. Conversely, pixels that fall outside the threshold are ignored. To reinforce this distribution more precisely, a GMM can be used to construct a skin colour model. Commonly, the GMM utilises the Expectation-Maximization (EM) algorithm to provide a powerful estimation on every established Gaussian component [3]. However, in order to improve the efficiency of the GMM, a huge variety of samples must be provided to provide better training data.

There is an alternative approach that aims to avoid skin colour segmentation due to its sensitivity to various lighting conditions. Instead, it detects the unique features of a face using the half-face template identification method [4]. Assuming that all subjects' faces are symmetrical, effective face detection can be performed by analysing half, either left or right hand side of the frontal face, to speed up the processing time and to ease the computational loads required to process a full face.

Other interesting recent research contributions in this field include the work by Zhou *et al.*. In [5], the use of perceptual features such as Gabor filtering in combination with diffusion distance was reported, of which the researchers claimed to be able to significantly improve the performance of facial recognition compared to some classical methods. Other work involves the integration of the extraction of semantic features from faces with Tensor Subspace Analysis (TSA) to perform recognition based on the eyes and mouth of the subject [6].

Another alternative for an efficient face detection is the Viola-Jones method of using Boosted Cascade Classifiers that made up of Haar-like features [7], [8]. This method suggests that with the aid of the machine learning algorithm Adaboost to generate a specific weight and size for each Haar-feature on large samples of training images [9], a powerful classifier can be created by cascading weak feature classifiers together. The product of these weak classifiers in cascade is a single Haar-feature classifier that consists of a combination of multiple Haar-like features that correspond to specific parts of the human face. An Integral Image is used to compute the Haar-like features at any scale or location.

The Haar-like features, also known as Haar-wavelets, are digital image features that contain a mixture of black and white regions. These features are used in the learning algorithm as they provide a better object classification rather than processing raw input [10]. The matching process works by applying the Haar-like features across an image and computing its Integral Image. In order to train the Cascade Classifiers, a huge number of training samples are required. These training samples are passed through the Adaboost training algorithm to develop the classifier in stages.

This widely used Viola-Jones algorithm yields around 90%

detection rate with negligible false detections [11]. False detection occurs when the algorithm wrongly registers or identifies a random part or object within an image to be a human face. At present, many algorithms still have trouble with such shortcoming.

This paper aims to perform an efficient and effective real-time detection and motorised tracking of the human face using simple systems-on-chip (SoC's). The required hardware to achieve this purpose are the Raspberry Pi microcomputer and the EasyLab microcontroller. It is proposed that the tasks of detection and tracking to be assigned to these 2 hardware individually, albeit with a communication for transmission of data between them. The face detection is performed based on the Haar-feature classifiers using ScicosLab on the Raspberry Pi while the tracking is executed with state-diagrams using SMCube on the EasyLab to control the servomotors that will in turn control the movements of the camera. Thus, by doing this, the remaining unused GPIOs of the Raspberry Pi can be used to power and to operate other hardware for other tasks besides face detection and tracking, especially that the Raspberry Pi can only supply a total of 51mA from all of its GPIOs.

This paper is organized as follows: Sections II and III provide the hardware and software overviews respectively. Section IV explains the face detection algorithm and Section V covers the algorithm for face tracking. The communication and transmission of data between the Raspberry Pi and the EasyLab is presented in Section VI. Section VII discusses the experimental setup and results and finally, Section VIII provides some conclusions.

II. Hardware Overview

The Raspberry Pi is a credit card-sized mini computer with the capability of a full-sized desktop machine. For this research work, the Pi is used to perform the face detection algorithm.

Meanwhile, the EasyLab is a simple and relatively cheap prototyping board with a dsPIC class microchip. Developed by Evidence, this board offers different approaches of which it can be programmed, including the possibility of using tools such as block diagrams-based ScicosLab with automatic code generation. This allows for realistic practical applications as block-diagrams can be used to model actual physical systems to great extent of accuracy [12]. Both of these devices are shown in Figure 1.

The camera used for image feed-in, as shown in Figure 2, is the Microsoft LifeCam HD-5000 which is 720p (1280×720 pixels) HD capable with Auto Focus built in, and is fully compatible with the Pi where the driver can be easily located and installed. The camera is mounted on 2 servomotors, where one controls the horizontal motion of the camera, while the other controls the vertical motion. As such, the camera can be panned and tilted on both the x-axis and y-axis.

III. Software Overview

The main software used in this system is ScicosLab, an open-source software developed by Evidence that utilizes

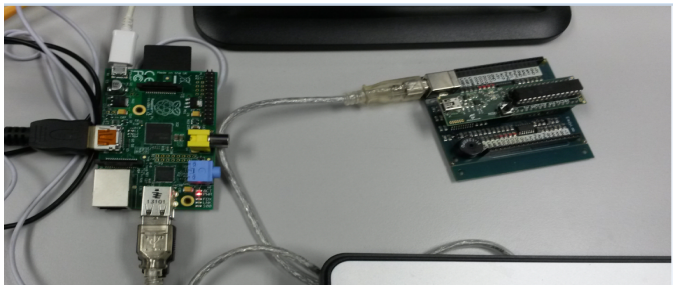

Fig. 1. The Raspberry Pi on the left and the EasyLab on the right connected via USB-serial connection.

Fig. 2. The Microsoft LifeCam HD-5000 Camera mounted on 2 servomotors.

block diagrams for modelling and simulation of dynamic systems, similar to MATLAB/Simulink. ScicosLab has a built-in automatic code generation function that can be used to translate the block diagrams into C-codes, which makes deployment of codes onto the EasyLab microcontroller feasible. For example, Figure 3 shows the block diagrams modelled in ScicosLab to represent the overall of the system proposed in this paper.

The SMCube, also developed by Evidence, is used for code generation of discrete-time state machines. It allows for the use of state diagrams to determine the current state of the system based on the data obtained from sensors such as the camera, and as such is able to determine the next operational state of the system. Figure 6 shows the 12 states of the servomotors operation modes for continuous tracking of the human face based on the position of the detected face in the image frame.

Lastly, MPLAB is used to upload the generated C-codes into the EasyLab.

IV. The Face Detection Algorithm

As mentioned in Section I, Viola-Jones developed the Haar-feature method that uses unique Haar-like features for face detection [11]. It can be safely assumed that each of these unique features gives a positive recognition to a specific part and/or section of the human face. The face implementation

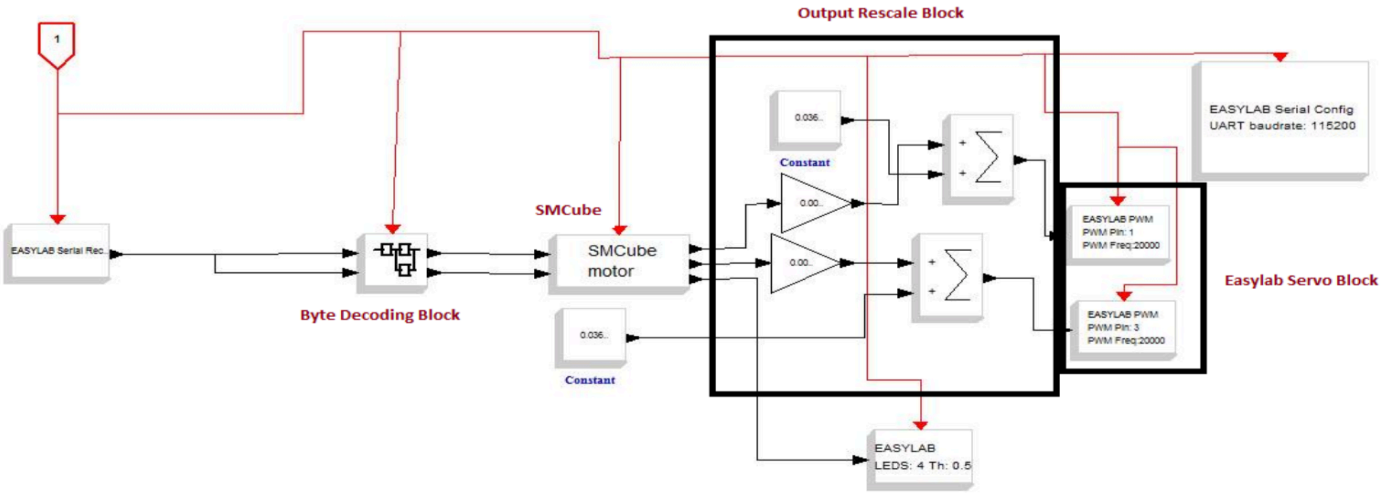

Fig. 3. The overall of the system as modelled in ScicosLab

Fig. 4. Samples of images with human faces representing positive images for training.

Fig. 5. Samples of images without human faces representing negative images for training.

of the Haar-feature Classifier can be described through the following phases:

A. Sampling Phase

The sampling phase collects and archives multiple source images representing a wide range of data on the different types and shapes of human faces. These samples are classified as positive images. On the other hand, non-human face samples are used as negative images to train the classifier to distinguish between human faces and otherwise. Altogether, 500 positive images and 1000 negative images are used as training data. Samples of these positive and negative images are shown in Figures 4 and 5.

B. Training Phase

The training phase matches and selects suitable Haar-features based on the different features and shapes of the human face from the training samples. The machine learning algorithm Adaboost is then used to match each Haar-feature to each sample image and assigns a weight based on every positive detection. Haar-features that provide no detection are eliminated. The greater the weight, the higher the priority that feature representing part of a human face will be. The end products are multiple Haar-feature classifiers that have been trained to correspond to the different parts of the human face.

C. Cascading Phase

Here in the cascading phase, each of the Haar-feature classifiers are cascaded in series to create a cascaded classifier and also to amplify the detection power. Each stage of classifier acts as a filter to continuously remove non-human faces until a powerful final cascaded classifier is obtained.

The final product upon the completion of the training process is a `.xml` file that holds information regarding the Haar-features for the face detection system. The classifier is then loaded into the detection process and each frame of the image is scanned, pixel by pixel, and compared with the trained data in the classifier file. The detection algorithm is programmed using the Python language on the Raspberry Pi. The algorithm can be summarized through the following steps:

1) Loads the classifier `.xml` file.
2) Obtains and stores the image from the webcam (attached to the Raspberry Pi via USB connection).
3) Downsizes the resolution of the image to enhance processing speed.
4) Feeds the image data into the classifier for face detection.
5) Highlights a region of pixels that provides a positive result from the face classifier.
6) The centre coordinate of the face is computed and sent to the face tracking algorithm.

V. Face Tracking Algorithm

The tracking algorithm is a series of functions called by the EasyLab based on the data containing the coordinates of the face sent by the Raspberry Pi. By decoding the serial bits back into the respective x- and y-coordinates representing the centre of the face in the image frame, the SMCube can then be

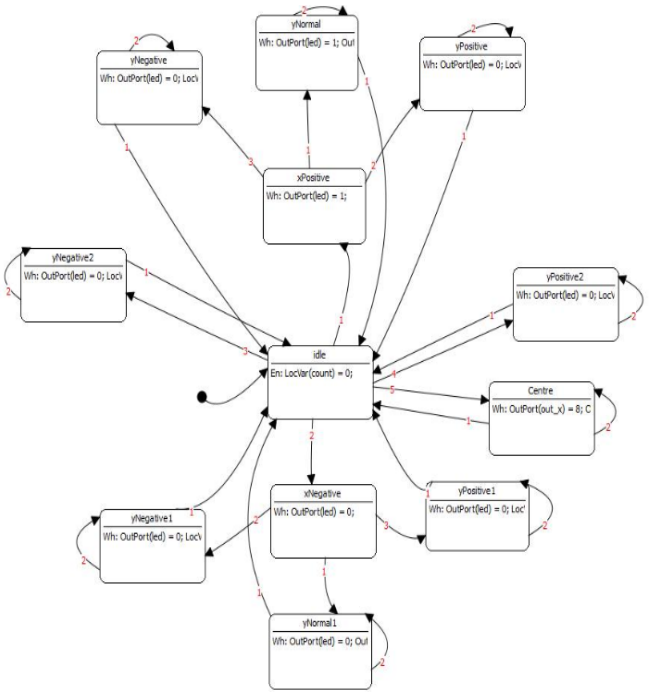

Fig. 6. The state diagram as modelled in SMCube

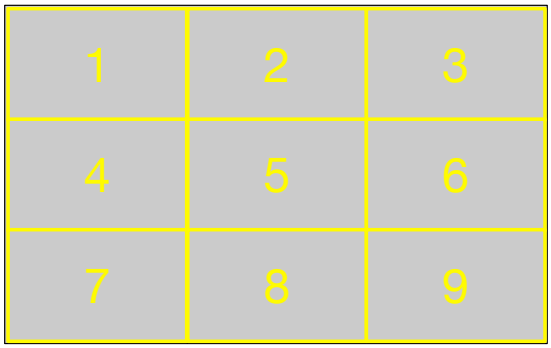

Fig. 7. The 9 regions of which the position of the face within the frame can be defined.

utilized to make decisions on the pan and tilt of the webcam using the two servomotors to keep the detected human face in the centre of the frame.

Figure 6 shows the 12 states based on the position of the face in the frame, i.e. the face is located below and on the right side to the centre of the frame etc. These positions are separated into 9 specific regions as shown in Figure 7. Table I explains in detail the 12 states in Figure 6, their relation to the regions specified in Figure 7, as well as the decisions that are to be made to the servomotors to pan and tilt the camera in order to track the human face.

TABLE I
THE 12 STATES IN SMCUBE FOR PAN AND TILT OF CAMERA BASED ON POSITION OF FACE IN FRAME

States	Position of Face (Region #)	Decision Made
`Idle`	Rest state	Camera at rest position and do nothing
`xPositive`	Right centre of frame (6)	Shift camera right
`xNegative`	Left centre of frame (4)	Shift camera left
`yNegative`	Bottom right of frame (9)	Shift camera lower right
`yNegative1`	Bottom left of frame (7)	Shift camera lower left
`yPositive`	Top right of frame (3)	Shift camera upper right
`yPositve1`	Top left of frame (1)	Shift camera upper left
`yNormal`	Right centre of frame (webcam height) (3,6,9)	Shift camera right
`yNormal1`	Left centre of frame (webcam height) (1,4,7)	Shift camera left
`Centre`	Centre of frame (5)	Do nothing
`yPositive2`	Top centre of frame (2)	Shift camera upward
`yNegative2`	Bottom centre of frame (8)	Shift camera downward

VI. COMMUNICATION AND TRANSMISSION OF DATA BETWEEN HARDWARE

A communication channel is set up so that the Raspberry Pi is able to transmit the coordinates of the detected face in the image frame to the EasyLab microcontroller. Since the EasyLab only utilizes the USB-serial connection via the MCP2200 microchip for communication purposes, hence the only viable method to establish a communication channel between the two devices is through a wired connection.

Using USB-serial communication, data transmits through the channel in the form of serial bits. This is now used to send the coordinates of the centre of the detected face from the Raspberry PI to the EasyLab, which will then be used to assist in the face tracking algorithm. Assuming that the image has been downsized to a resolution of 540×480 pixels in order to enhance processing time as well as to reduce the amount of graphical data to analyze.

The general idea is to encode the x- and y-coordinates in pixels into 8 serial bits. The left most 4 bits will carry information about the x-coordinate value while the right most 4 bits will store the y-coordinate data, as shown in Figure 9. Figure 8 meanwhile shows an example of converting a

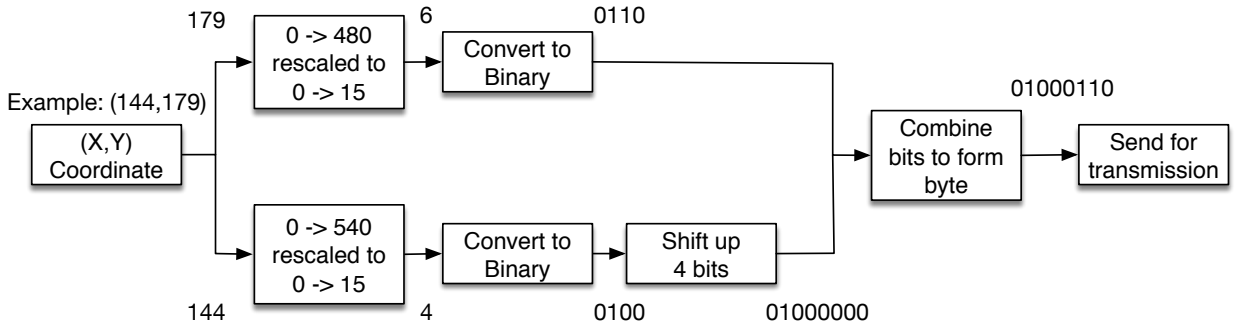

Fig. 8. Encoding the coordinates into 8 bits for transmission.

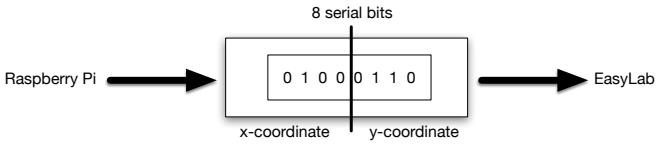

Fig. 9. Coordinates of image represented by 8 serial bits.

coordinate in pixels from the image into 1 byte using this method.

Assuming that the centre of the face is located at position in pixels (144,179) in the frame, both values are scaled to a maximum value of 15 in order to fit the data into 4 bits respectively. Shifting the x-coordinate binary data by 4 bits to the left, and then combining with the y-coordinate binary value, will result in `01000110`. This information will then be sent to the EasyLab microcontroller. The decoding process will then take place in the EasyLab to convert the binary data back into coordinate positions.

This method however, has one disadvantage. During the encoding and decoding process, the coordinates of the source data at the Raspberry Pi are never exactly matched to the decoded values at the EasyLab due to the scaling of the coordinate values into their respective 4-bit representations. As such, this causes a slight offset in the data with an average of about 30 pixels for both axes, which is still insignificant compared to the full resolution of the image. Therefore, this method is still acceptable if not used for high definition tracking applications.

VII. Experimental Results

The system is assembled as shown in Figure 10. Figure 11 shows the success of the face detection algorithm in recognizing subjects with different skin tones. Observe also that the faces are successfully detected for non-single coloured backgrounds as well as the presence of other objects that might act as noise. The system has an accuracy of 90% detection rate when the face is fully exposed with a maximum 15° of tilt angle.

With a data sampling rate of 0.01s, the face tracking system has a delay of approximately 0.5s. As the frame size of the

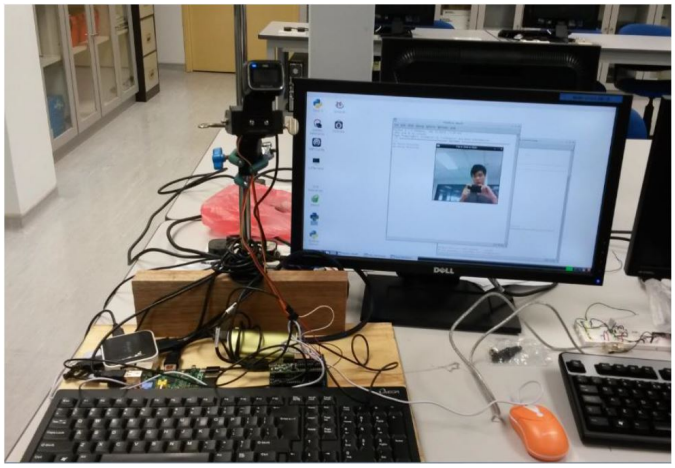

Fig. 10. Assembly of the experimental setup.

image has been scaled down to a resolution of 540×480 for faster response time as well as less graphical pixels to evaluate, the detection range for the human face is limited to about 1–3m away from the webcam. As shown in both Figures 12 and 13, the system is able to track a moving human face within the frame.

VIII. Conclusion

This paper has presented the real-time detection and motorised tracking of the human face using Raspberry PI and EasyLab. The detection algorithm is carried out via ScicosLab on the Raspberry PI and the tracking algorithm is performed on the EasyLab using SMCube. Given that the Raspberry PI is only able to supply limited amount of current through its GPIO pins, it might not be feasible to use it to power other hardware if it is used to control the servomotors as well for tracking. Furthermore, with the SMCube being able to provide precise decision based on the current state of operation, it a more preferred choice for the control of the servomotors to ensure effective tracking of the human face. An overall tracking of the human face could be achieved with a relatively short 0.5s delay.

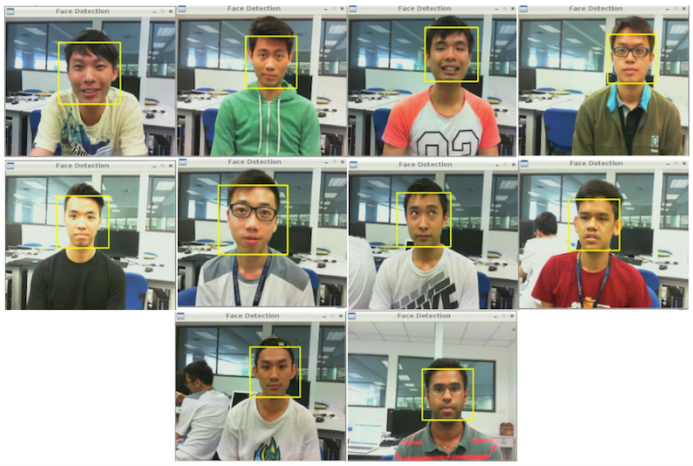

Fig. 11. Detection of the faces of multiple subjects.

Fig. 12. Tracking of a moving human face (1).

Fig. 13. Tracking of a moving human face (2).

REFERENCES

[1] M.-J. Seow and V. K. Asari, "Homomorphic processing system and ratio rule for color image enhancement," in *IEEE International Joint Conference on Neural Networks*, vol. 4, 2004, pp. 2507–2511.

[2] L. Zou and S.-i. Kamata, "Face detection in color images based on skin color models," in *IEEE Region 10 Conference TENCON 2010*, 2010, pp. 681–686.

[3] R. A. Redner and H. F. Walker, "Mixture densities, maximum likelihood and the EM algorithm," *SIAM Review*, vol. 26, no. 2, pp. 195–239, 1984.

[4] W. Chen, T. Sun, X. Yang, and L. Wang, "Face detection based on half face-template," in *9th International Conference on Electronic Measurement & Instruments, 2009 (ICEMI'09)*, 2009, pp. 4–54.

[5] H. Zhou and A. H. Sadka, "Combining perceptual features with diffusion distance for face recognition," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 41, no. 5, pp. 577–588, 2011.

[6] H. Zhou, Y. Yuan, and A. H. Sadka, "Application of semantic features in face recognition," *Pattern Recognition*, vol. 41, no. 10, pp. 3251–3256, 2008.

[7] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, 2001, pp. I–511.

[8] M. Jiang, A. H. Sadka, and H. Zhou, "Automatic human face detection for content-based image annotation," 2008.

[9] M. G. Krishna and A. Srinivasulu, "Face detection system on AdaBoost algorithm using Haar classifiers," *International Journal of Modern Engineering Research*, vol. 2, no. 5, pp. 3556–3560, 2012.

[10] R. Lienhart and J. Maydt, "An extended set of Haar-like features for rapid object detection," in *Proceedings of International Conference on Image Processing, 2002*, vol. 1, 2002, pp. I–900.

[11] P. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[12] W. J. Palm, *Modeling, analysis, and control of dynamic systems*. Wiley New York, 1983.