National Conference on Engineering Innovations and Solutions (NCEIS – 2018)
International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2018 IJSRCSEIT | Volume 4 | Issue 6 | ISSN : 2456-3307

Emulation Of Associative Learning in a Humanoid Robot using Artificial Neural Networks

Prof. Bhagirathi bai V*, Tanmay N Deshmukh, Kishore R
Department of Mechatronics Engineering, Acharya Institute of Technology, Bangalore, Karnataka, India

ABSTRACT

The objective of this paper is to document the development, design, and implementation of a humanoid interface to a robotic implementation of the right human forelimb, the purpose of which is to interact with its environment under programmed instructions by imitating the movements of an arm and engage users in conversations. The arm has 10 Degrees of Freedom in total and weighs 2 kilograms. The software driving this interface is put to action by a microcomputer primarily in the form of a chat-bot which aids in human-robot interaction orally real-time. The areas for future application of this include human-machine interaction in industry & medical, social, & hospitality settings. The necessary mechanisms for arm control and the rest are presented in this paper.

Keywords: Humanoid robot, machine vision, face recognition, chatbot

I. INTRODUCTION

The need for robotics and automation has moved beyond the factory floor to enter relevance in almost all fields of human interest Giving an automation system a humanoid shape and function not only make their actions more comprehensible to the user, but also socially acceptable. Life-size humanoid robotics is the next avenue these field must explore. It essentially replicates the human element to complete tasks assigned to it, which the user can easily understand and exploit to the required extent. The self-funded project described in this paper attempts to do just that. Along with the arm, the robot built also boasts of a voice-driven intelligent chat-bot feature which deals with information in the aural and textual form. The service offers a conversational context-based response to make it appear more human. All this coupled with the movable camera housing modelled around a life-size human head has given the robotic interface a familiar look and feel leading the writers christening it with a relatable human name 'Charlie'. This paper outlines the design of the torso and its integration with the chat feature.

II. LITERATURE SURVEY

One of the hallmark of a humanoid design was the Honda Asimo[1] developed by the Toyota Group, debuted in 2000. Several other humanoid robots were developed since. The HRP-4 Robot [2] is a humanoid consisting of 7DOF arms weighing about 39Kgs. It uses complex mechanisms for actuating all the joints in the robot. However, we need to overcome the complexities of such designs and need to consider costs when designing these systems. We aim to achieve a focused human-robot interaction, initiated by an encounter such as eye contact, or a statement made in a tone of voice. A study [3] found that people evaluate a physical embodied social agent

more positively than a disembodied social agent. Robots are being used in factories, they are being weaponized in military, and are also being sent to outer space. Yet, we do not have them in our homes. We need to make progress in the field of social robotics and our fascination towards humanoid robots and the affect it has to people forms our motivation for this work.

III. HARDWARE DESIGN

This is the design of the humanoid torso. Each of the robot's arms consists of a 3DOF shoulder joint, an elbow joint, and the wrist rotation, giving a total of 5DOF per arm, excluding the movements of the fingers. Actuation of the robot's arm joints required high torque motors delivering over 40kg-cm force. A Johnson DC side-shaft geared motor meeting the required specifications were used in every joint. Potentiometers are used for position feedback, as a DC motor does not possess absolute positional control. 3D design is developed in Catia V5, a 3D modelling software.

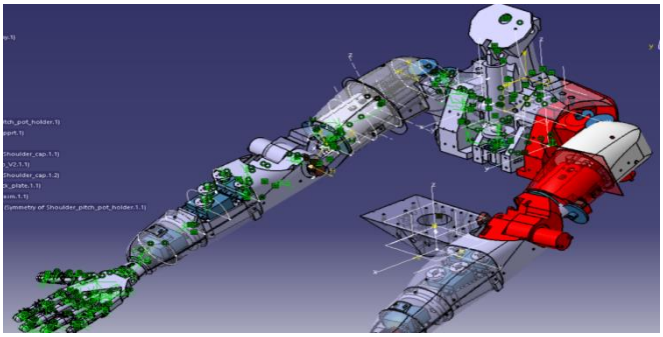

Figure 1. 3D Assembled CAD design.

A. Hand

Each finger is made of three hollow segments that act like a kinematic chain, mimicking a human finger. A tendon strand made of 3 beading wires runs over the pulleys and under the roofs of the segments. Pulling on this tendon causes the finger to fold. Another tendon runs below the pulleys and over the base of the segments. Pulling on this tendon causes the finger to undo the folding and bringing it to an open position. The tendons pass through the hollow palm, and then through the axis of the wrist gear. The wrist is actuated by a MG-995 servo motor, a common motor used in hobby robotics.

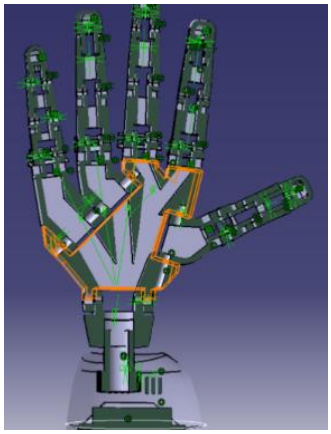

Figure 2. Channels(hollow, gray) for tendons

B. Forearm

The forearm houses 5 MG-995 servo motors that actuate the fingers independently. The tendons from the wrist gear go through individual holes at the bottom of the skeletal structure, and guides them out at the top, where they would lead to their respective servo motors. Rotation of the servo clockwise pulls the actuation tendon and relaxes the other, and the opposite is true for a counter-clockwise rotation. The elbow joint is actuated by a high torque Johnson motor, held by the bicep assembly and the shaft is held by the forearm. A potentiometer is coupled with the motor shaft, on the opposite side.

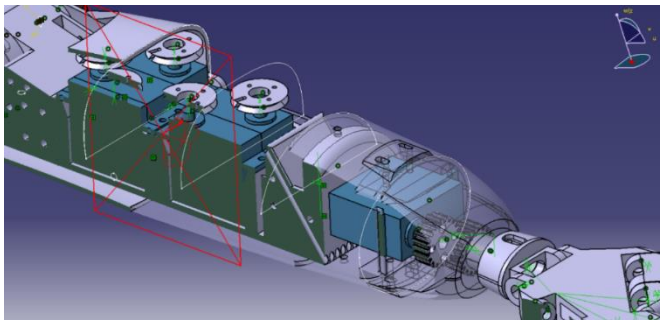

Figure 3. Cross-section of 3D model of the forearm

C. Bicep

The bicep houses the shoulder yaw-rotation DC motor. The feedback potentiometer is placed parallel with the axis of the motor shaft, coupled by gears having gear ratio 4:3. That is, a 180' rotation of the arm, causes a 270' rotation of the potentiometer shaft.

D. Shoulder

Two DC motors constitute pitch and roll motions. The roll motion and its potentiometer feedback is achieved in the same way as the elbow design, with the potentiometer placed on the opposite side, along the axis of rotation. The pitch motor is located at the robot's clavicle, with its potentiometer directly under the shaft, coupled by gears in the ratio 1:1.

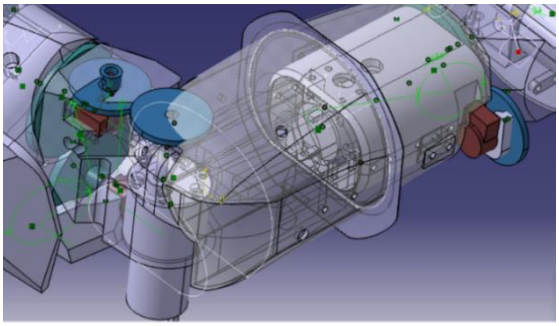

Figure 4. CAD model of the shoulder

E. Arm support and Sternum

These set of parts are printed with the highest density setting since it holds the entire arm's weight. This portion houses the shoulder pitch rotation DC motor. To prevent the motor from tilting because of the arm's weight, an additional "sleeve" is inserted after inserting the motor and is bolted in place. Screws from the back panel and from the bottom secures the assembly firmly.

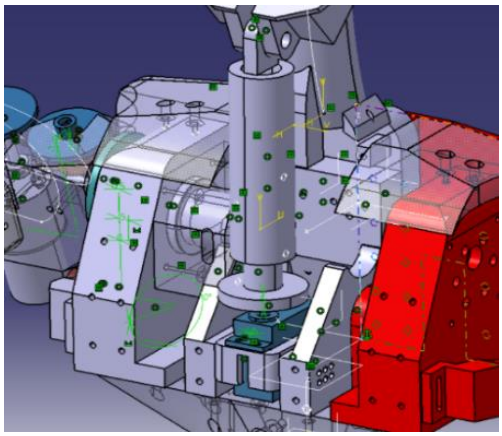

Figure 5. Sternum design

F. Face

InMoov is a robot developed for artistic purposes by French sculptor Gael Langevin in September 2011 (The first blueprint files were published in January 2012 on thingiverse.com) and its files are under Creative Commons license. The face has provisions for neck rotation servo and the jaw movements.

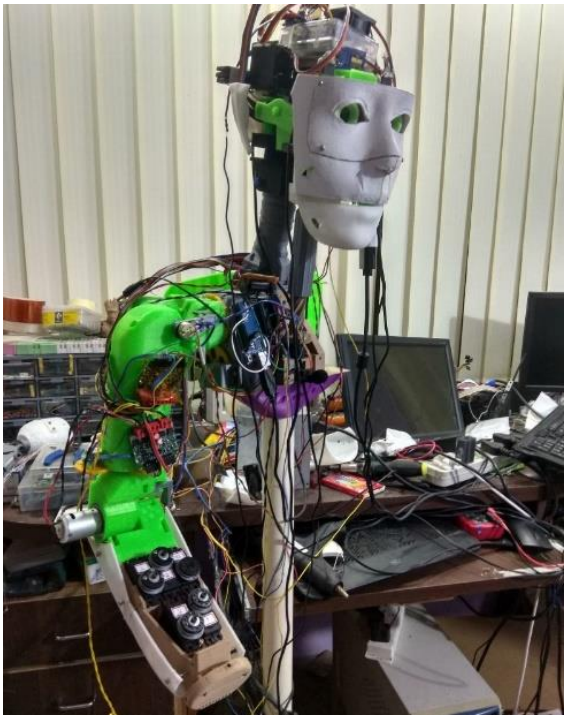

Figure 6. Assembled robot with right arm(till wrist) and Inmoov head.

G. Electronics

The two web-cams used are Microsoft HD-3000, for which the Inmoov eyes are designed. They provide superior quality images and have small form factor. The web-cams are plugged into the laptop computer where image processing takes place. The robot's on board Raspberry Pi and the laptop computer used to process images are connected to a LAN. Post image processing, necessary data like servo tilt angles for tracking, are transmitted to the Raspberry Pi through a TCP port. The RPi receives position values from the potentiometers through an Arduino Mega, an Atmel microcontroller board.

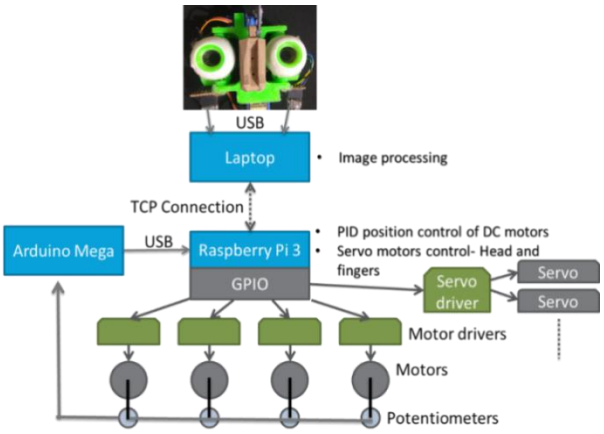

Figure 7. Electronics layout

IV. SOFTWARE DESIGN

The main function of the Arduino is to read analog values and send them to the Rpi. This goal is to send these values in fewest number of bytes as possible, as frequently as possible. Failing which, the PID position control of joints fail. The ADC on the Arduino has a resolution of 10 bits for each channel. Serial communication only allows transfer of 8 bits at a time, therefore, each channel's value must be split into 2 bytes. The first byte contains the 2 MSBs of an analog value and the second byte contains the rest of 8 bits. These two are then recombined at the RPi. The Arduino reads 8 such values from 8 potentiometers corresponding to the joint motors in both arms. All 8 values are read and split into 2 bytes, adding up to 16 bytes. A delimiting byte is added, and all 17 bytes are sent to the RPi after each cycle.

The RPi's software consists of a multi-threaded program, with each thread running on an infinite loop. This is done so because multiple independent tasks need to be executed without waiting for unrelated actions. All threads, share a set of global variables. These variables are volatile, that is, their value can be changed by any thread at any time, making them unpredictable. The main global variable is the 'setpoint', controlled by the GUI thread, and sequence thread. The GUI allows manual control of joint angles useful for testing and debugging. The sequence thread is a program snippet that can move the robot arms in a pre-recorded motion. It does so by iterating through each position in the list of target positions, assigning the setpoint to the target positions for each joint, and pausing the sequence thread for the specified amount to allow the arm to come to the set position. The PID thread takes care of bringing the arm to the setpoints as it is being continuously scanned. It receives the analog position values from the Arduino. This thread waits for 17 bytes to get queued in the serial buffer before processing them.

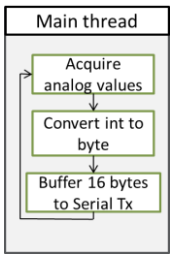

Figure 8. Flowchart of the code running on Arduino

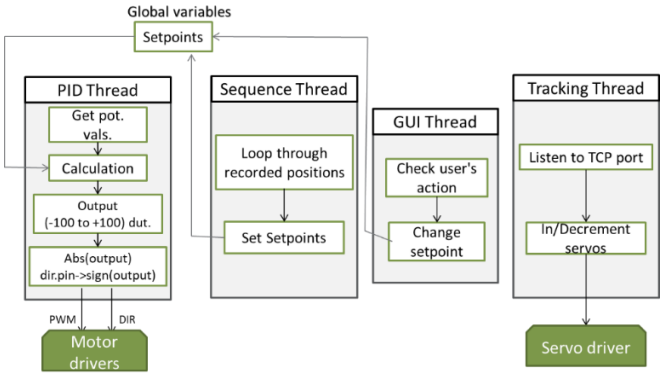

Figure 9. RPi code flow chart

These are 8 position values (2 bytes each) and a known delimiting byte, 17 bytes in total per scan. If the 17th byte is not the delimiting byte, the data is considered corrupted and the serial buffer is flushed to allow fresh data in the next cycle. The PID thread does not modify the setpoints, it only accesses it.

The tracking thread starts a TCP connection to the server on the laptop which has the cameras connected to it. From there, it listens to the port for incoming messages. The server sends JSON objects to the RPi (the client) when a specific object such as a face or a hand sign is in its field of view after processing the images from the cameras. The received JSON objects from the server contains the

percentage deviation of the object being tracked, from the center pixel (-100 to 100) both horizontally and vertically. This thread then adjusts the 2 servo motor angles of the eyes such that the deviations is zero. The 2 head servos are also adjusted such that the deviations of the eye servo motors from its mean position, is zero. The result is a tracking effect, where the eyes look at the object first, and then the head adjusts such that it is facing the object of interest, much like a human reaction.

A. Image processing thread

We use a library that utilizes HOG (Histogram of Oriented Gradients) [8], to detect faces, where each pixel is compared with the surrounding pixels. Then, an arrow id draws pointing in the direction of the change in pixel brightness. The image is broken into small squares of 16x16 pixels, each box containing the gradient pointing in the major direction. This turns the image into a simple representation. A simple pattern matching is done with a standard face HOG to find the location of the face (if any) in the image. A bounding box is drawn around it and the cropped image is sent to the face recognition program.

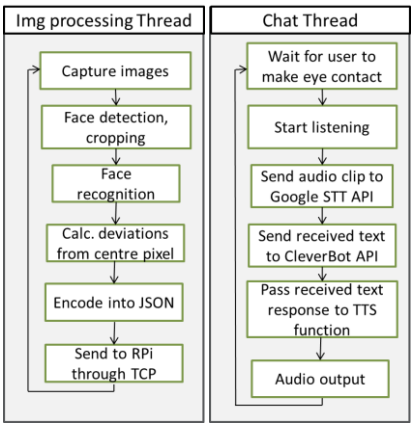

Figure 10. Off-board Python flow chart

The face recognition program uses a pretrained Deep Convolutional Neural Network that generates 128 measurements that are unique to each person[8]. These measurements are known as an encoding. Initially, a sample image of a person and the label (name) is given. The DNN produces the encoding and stores it in a database. During runtime, when a face is detected, the face is passed through the DNN and the encoding is determined. A pattern matching is performed with the produced encoding and the database of encodings. If a match occurs with the encoding, the person is labelled with the name given at the beginning. If there are no matches, the person is labelled as "unknown". Therefore, when given a person's sample image, and a label (their name), the program can identify that person. The bounding box around the person's face gives us the middle pixel location of the face. The distance between this pixel and the center pixel of the original image from the camera, is calculated in x and y directions. This x and y axis deviations are encoded and packed into a JSON object and then sent to the RPi through the TCP port, where it would be unpacked and used for adjusting the servo motors.

Figure 11: Screenshot of face recognition test showing identification of user by processing the image on the GPU on a Windows 10 platform.

B. Chatbot thread

This part of the program has 3 main steps

i) Speech to text

ii) Cleverbot API

iii) Text to speech

We use the google API for Speech to text conversion. Cleverbot is a chatterbot web application that uses an artificial intelligence (AI) algorithm to have conversations with humans. It was created by British AI scientist Rollo Carpenter. Unlike some other chatterbots, Cleverbot's responses are not pre-programmed. Instead, it learns from human input. Text generated by the STT service when an audio input is given to the robot, is passed on to the

cleverbot JSON API. The API generates a response which is a conversational response. This text response is fed through a text-to-speech service in python, using the library "pyttsx".

V. OPERATION

When a person comes in front of the robot, the robot looks for a face using the HOG algorithm. Once the face is detected, the part of the image within the bounding box of the face detection, is passed through the face landmark estimation followed by pose correction by basic image transformations. This image is now used to generate face encodings[8] by passing it through a D-CNN. The 128 measurements are then looked up in the database of previously recorded faces. If the encodings match any, the robot addresses the person with their name from the database. If the encodings do not match, a new entry is made in the database and the robot prompts the user to say their name. With the face detection, the robot is programmed to look for the user's eyes using Haar cascades[9], to indicate whether the user is making eye contact with the robot. If found so, the robot listens through its microphone. When the user says something, the average energy of audio input increases above the threshold. Listening stops when the energy falls back to the normal energy, indicating that the user has stopped speaking. This part of the audio is sent to the STT service to convert it to text. The text is then sent to the cleverbot API, whose response is converted to speech and output through the robot's speakers.

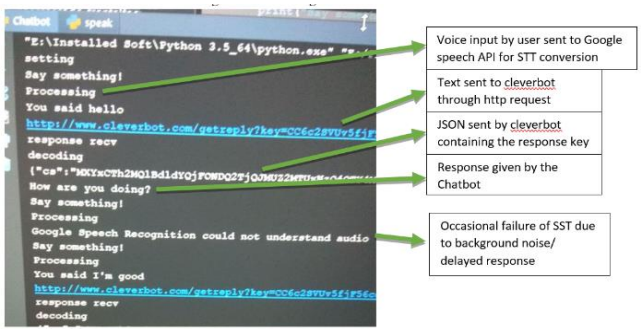

Figure 12. Screenshot of the output screen of PyCharm showing the Chatbot thread verbose.

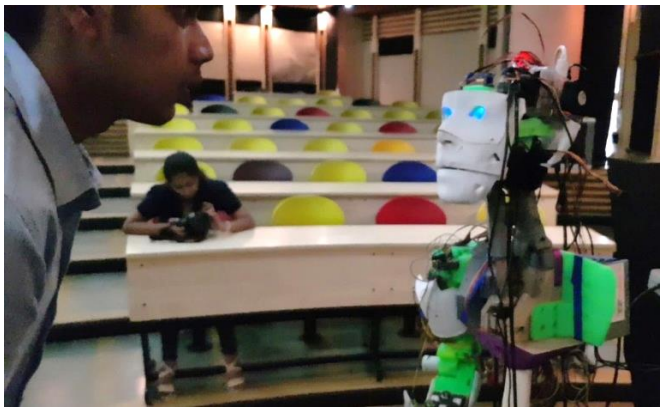

Figure 13. Charlie robot interacting with the user while maintaining eye-contact.

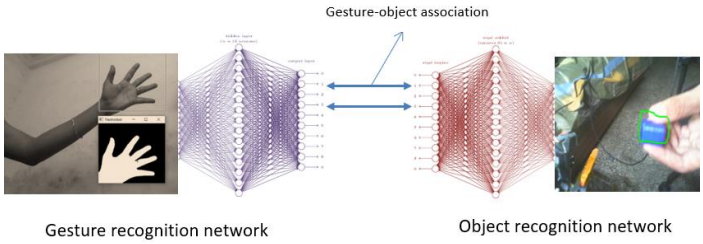

Figure 14. Associative learning

There are a series of Artificial Neural Networks that identifies objects, gestures, and helps in physical control of the movement of its arms. When the robot encounters a new object and a gesture command, its Networks are evolved through Neuro Evolution to accommodate this new information. When it is presented with a previously encountered gesture, the robot will be programmed to look for the object it is field of view. Once found, a Visual Servo-ing Network takes over, to control the robot's arm to reach for it. Training of these Neural Networks is done one after the other in the case of gesture and object identification and will be done simultaneously in the case of the servoing network and the PID motor control. The Neural Networks depicted are concerned with Gesture recognition, object recognition and mechanical control of the robot's arm, shown in purple, red and blue respectively. Each output neuron of a neural network is built to correspond to a certain predefined result. In the case of the Gesture Identification NN, each result is the inference that the input image is that of a hand-sign. For the neural network to be "trained" to infer the right result, the weight of the connections leading to the output neuron associated with the expected

result is increased, promoting this network path in the process. This method of deriving the expected result by tweaking the connections is called back-propagation. The same method is implemented in the Object Recognition NN, with the connections to the output neuron associated with the right object encouraged in a similar fashion. The association between the identified object and hand-sign is established by displaying the sign and the object consecutively, completing the training process.

VI. CONCLUSION

Our robotic platform can successfully manipulate its arm, recognizes stored faces and can learn to recognize new ones, and interact with the familiar face aurally with the ease of an average human being with an above-average amount of cheek. We expect Charlie to be able to serve as a multi-purpose platform for various interfacing operations because of the versatility of programming of his on-board microcomputer and the natural adaptability of his humanoid arm. Though various automation techniques have been developed, only a select few have been implemented in humanoid form. We plan on further developing the robot by adding more social behaviors. A Neural network implementation can greatly increase the robot's capacity of grasping information such as objects. Current work is being done on improving the Associative Learning algorithm to make it more interactive, while working on the mechanical design for greater efficiency and cost reduction.

VII. REFERENCES

[1] Hirose, Masato, and Kenichi Ogawa. "Honda humanoid robots development." Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences 365.1850 (2007): 11-19.

[2] Kaneko, Kenji, et al. "Humanoid robot hrp-4-humanoid robotics platform with lightweight and slim body." Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on. IEEE, 2011.

[3] Lee, Kwan Min, et al. "Are physically embodied social agents better than disembodied social agents?: The effects of physical embodiment, tactile interaction, and people's loneliness in human–robot interaction." International Journal of Human-Computer Studies 64.10 (2006): 962-973.

[4] Robins, Ben, et al. "Robotic assistants in therapy and education of children with autism: can a small humanoid robot help encourage social interaction skills?." Universal Access in the Information Society 4.2 (2005): 105-120.

[5] International Journal of Computer Science & Information Technology (IJCSIT), Vol 2, No 6, December 2010 DOI : 10.5121/ijcsit.2010.2614 153 "Segmentation and object recognition using edge detection techniques"

[6] Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. Vol. 1. IEEE, 2005..

[7] Kazemi, Vahid, and Sullivan Josephine. "One millisecond face alignment with an ensemble of regression trees." 27th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, United States, 23 June 2014 through 28 June 2014. IEEE Computer Society, 2014.

[8] Schroff, Florian, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recognition and clustering." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[9] Viola, Paul, and Michael Jones. "Rapid object detection using a boosted cascade of simple features." Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on. Vol. 1. IEEE, 2001.