Computer Engineering Capstone Project

System Design Document

Authors: Mohammed Hoque
Jonathan Lobaugh
Troy Tancraitor
Lee Steen
Joe Mallozzi

Date: 6/18/2004
Team: VisionPSU
Course: CENBD 481
Senior Design
Advisor: Ralph Ford
Ron McCarty

Table of Contents

Table of Figures

1. Abstract

VisionPSU is a human interactive robot with voice and audio recognition capabilities. It is being designed and developed to differentiate between a specific set of users based on video and audio input in an ideal environment.

The recognition is solely limited to the five group members and project mentors. The system attempts to recognize the person in front of the camera, and the name of that person is displayed through a LCD mounted on the system. The robot is capable of responding to users by physical movements. This project is being integrated with six separate modules: Robotics, Communication Layer, RTS (real time control system), Audio Processing Layer, Image Processing Layer and Neural Network. Designing a human interface device with a high accuracy rate will likely lead to more sophisticated systems in the future.

1.1. Acronyms and Terms

- **HID** – Human Interface device
- **IP** – Image processing
- **NN** – Neural networks
- **AP** – Audio Processing
- **CF** – Communication functions
- **IC** – Intercommunications
- **RTS** – Real-time control system
- **M** – Mechanics
- **USB** – Universal Serial Bus
- **FFT** – Fast Fourier transform
- **LED** – Light emitting diode
- **LCD** – Liquid crystal display
- **RQ** – Requirement

2. Problem Statement

2.1. Need

As computation becomes ubiquitous and communications get enriched, the field of Human-computer interaction (HCI) faces new challenges of platform development of interaction between humans and machines. While we work toward an age of further dependence on machines even to perform the simplest of tasks, the issue of having a better platform of communication between humans and machines becomes a motivating topic of research for engineers.

2.2. Objectives

The objective of this project is to develop a system that allows the user to interact effortlessly, simplify user requests, and transfer information to and from a digital medium. It also allows for an improvement of the interface with a conventional machine in daily applications. The real-life presence of the robotic interface gives a more "personal" and one-to-one type of interaction between the user and computer**.**

- **Primary Objectives**
    - Being able to detect the user (mainly the five group members and the project mentors).
    - Being able to detect different human faces and remember them
    - Being able to detect different voices
    - Being able to have the robot move according to the user's gesture.

3. Problem Analysis & Research

3.1. Detect different human faces and remember them

Among the different technologies that are being used to detect human faces and to remember them, the "Eigenface Method" **(Eigenfaces** are the eigenvectors of the upper-dimensional vector space of potential faces of human beings. It is widely used for its simplicity and computational efficiency) appears to be applicable.

The Eigenface method attempts to find the principal component of the distribution of faces, or the Eigen vectors of the covariance matrix of the set of face images by using an information theory approach of coding facial characteristics. The following limitation may, however, apply;

1) This approach only recognizes and remembers people who interact with the system on a regular basis; not every single person the system comes across. 2) It may fail to recognize a person despite of his/her frequent encounters.

Intel's open source computer vision (openCV) library has the low level routines written for object detection, which is now known to be one of the most accurate detection systems in a real time environment. The one disadvantage to this approach is, since it's a fairly new technique, most of its functions are still highly experimental.

3.2. Extraction of distinctive features of the face

The following technologies were analyzed as possible candidates for features extraction.

3.2.1. Principal Components Analysis

"Principal Components Analysis (PCA) is a well known method for dimension reduction. PCA has been used widely in computer vision applications. It allows high dimension data to be represented in a lower dimension subspace. PCA is the optimal transform for dimension reduction as it allows correlated data in a N-Dimension space to be modeled in a lower dimension space without losing a significant amount of information." (Oziem, David).

3.2.2. A Hough Transform

Hough transform is a mapping from an observation space into a parameter space. The observation space is the image and there are certain structures contained in the observation space. The parameters of the structure define the parameter space. In a Hough Transform, each point in an image space "votes" for that part of parameter space, which describes structures which include the point.

3.2.3. Face Recognition in Fourier Space

Fourier space recognition is done by finding the closest match between feature vectors containing the Fourier coefficients at selected frequencies.

3.2.4. Template Matching

"Template Matching techniques compare portions of images against one another. Sample image may be used to recognize similar objects in source image. If standard deviation of the template image compared to the source image is small enough, template matching may be used.

The matching process moves the template image to all possible positions in a larger source image and computes a numerical index that indicates how well the template matches the image in that position. Match is done on a pixel-by-pixel basis."

Template matching is useful for identifying printed characters, numbers, and other small objects.

There are other face recognition techniques that exist but we believe that understanding those methods and implementing them is beyond the scope of this project given the time frame.

- Bayesian Intrapersonal/Extra personal classifier
- Gabor filter matching algorithm.
- Linear Discriminant Analysis
- Discrete Cosine Transformation

Among all the different feature recognition methods that are looked into, Template Matching seems to be the viable option to implement within the proposed time constraint

3.3. Detect different voices

There are currently many different algorithms that can be used to identify humans. Unfortunately, these algorithms tend to be very computational expensive, and therefore extremely slow. Also, the algorithms can only be used in a closed system (or a system of known users) and a key phrase or password must be clearly stated to the machine. There are currently no algorithms, to our knowledge, that work in real time.

Our proposed method takes ideas' based from many of these algorithms, and try to make real time detection. The voice recognition component is used as a conformation to the face detection. Our proposed method uses time domain to detect when a person is talking at normal tone as s/he usually would, and then use a Fast Fourier Transform to extract features that may be unique to humans (currently under study at University of Waterloo).

3.4. Have the robot move according to the user's gesture.

Motion can be accomplished with the use of four major technologies. Servo motors, solenoids & position detection sensors, stepper motors, and finally conventional motors with position detection sensors.

Servos	Steppers	Solenoids	Position relational conventional motors
Continuous control signal	Low torque	High power requirements	Development required for control circuitry.
Open loop interface	Closed loop interface required	Choices are limited	Not very accurate
Expensive	Expensive	Linear motion only	
	Needs reset cycle		

Table 1: Motion controllers

In addition a controller is needed to be implemented to operate the devices. This could compose of a programmable logic device (PLD) or micro-controller.

PLD	Microprocessors
Requires extensive design	Can be expensive for high speed microprocessors
Difficult to upgrade	Proprietary code implementation
Needs special programming equipment	

Table 2: Microprocessors

In motion control, a logical hardware component needs to interface to the computing platform and drive the motion with a supply of position feedback. The motion conversion section can be implemented with cabling and a centralized motor region, or with a distributed motor layout throughout the robotics.

In the general control interface, the microprocessor, or PLD implementation needs to conform to the computing platform standard. The primary choices in the area of interface are universal serial bus (USB), RS232 serial, PCMCIA, or parallel port.

USB	RS232 Serial	PCMCLA	Parallel port
Difficult to interface to	Slow transfer rates	Difficult to interface to	Slower speeds
Limited wire distance	Being phased out of modern PC's	Requires low level programming access	
		Not available on all laptops	

Table 3: Communications

4. The Engineering Requirements

Marketing requirements	Engineering requirements	Rationale
Ease of use	Standardized adaptors	Prevents unneeded proprietary hardware connectors
	Centralized control	Alterations and adjustments only needed in one place
	<= 2 required physical connections	Minimal connections reduce complexly of hookup
Cost	Target production price < $250	This is based upon generalized price ranges for materials required
Reliable	Detects a human voice 80% of the time	Reasonable success rate based on current achieved results
	Detects a human face 80% of the time	Reasonable success rate based on current achieved results
Accurate	Detects a specific user voice 80% of the time	Reasonable success rate based on current achieved results
	Detects a specific user 80% of the time	Reasonable success rate based on current achieved results
Medium sized	Must be easy to transport from place to place by one average person.	Due to cost constraints, durability issues, and environment
Efficient	Must be able to finish visual and audio computations in less than 2 sec	To increase accuracy, reliability, and interaction between user and system

Table 4: Requirements

5. Standards and Realistic Constraints

5.1. Constraints

- **Economic constraint**
  - Materials must be under $250.00 (constrained by ECE department)
- **Environmental constraint**
  - **Audio Processing**
    - A minimum level of 15 decibels must be output by the person talking to the robot. This is a level of decibels that is what the average human begins to detect sound at.
    - For audio processing, the background noise in the surrounding environment would be ideally a minimum.
    - The number of users on the audio system is set at a static number. VisionPSU project team has 5 users. The audio is left open for dynamic changes so the number of users can changed dynamically.
    - The average human voice has a range of 130-2000htz. Receive audio data must range from 300-600, which is the average talking voice. Anything higher may result in an outlier in the data. Anything lower may not be recorded.
  - Video Processing
    - The user does not have to be to close to the camera; however the greater the distance the user is from the camera the less accurate the results will be.
    - During presence of multiple people in front of the camera, the image processing module is only concerned with the face closest to the camera. Processing might not be accurate or time consuming if there are a large number of faces in the picture.
    - Unit only works in standard lighting environments
- **Ethical constraints**
  - Design will not infringe on any existing patents or copyrights
  - Unit will not harm the user
- Health & Safety constraints
  - Unit will not provide more than 20V of electrical charge
- Manufacturability
  - Main circuitry will be fabricated on a PCB (printed circuit board)
  - Standard adaptors will be used
- Political

        - The system will not require government approval or trade barriers
    - Social
        - Unit will be designed to be socially and visually appealing
        - Unit will not frighten the user especially users under the age of 7
        - Unit must be easy to use and operate by the general public
    - Sustainability
        - The unit must last at least 1 year
        - System software will be modular to allow reusability for further development

5.2. Applicable Standards

- **Communications**
    - USB v2.0 (universal serial bus)
    - Wireless - 802.11b
    - Audio signal
    - Bluetooth
- **Data Formats**
    - Bitmap (image)
    - JPEG Compression (image)
    - PCM Waveform (Audio)
- **Programming languages**
    - C / C++ (OOP)
    - Assembly
    - JAVA
    - MatLab
- **Connectors**
    - USB
    - RCA
    - Mini-jacks
    - Ribbon/Pin

6. Design

A high level design of this project is displayed in figure 1, in which the communications between different modules are defined. *The Robotic Layer*, which has the camera and microphone mounted on it, is used to interact with users by displaying the results of human identification as well as to acquire inputs from the environment. *The Communications Package* facilitates intercommunication between robotic and processing layers through the *RTS (Real Time Control System),* which is the main operator of the system processes, audio and image processing.

The *Audio Processing* layer, one of the units of the processing layer, takes the data coming from the microphone as inputs, and outputs the processed data consisting of user specific identity vectors.

The *Image Processing* layer, other unit of the processing layer, takes the data coming from the video camera as inputs, and outputs the normalized profile vectors of distinctive features of the face of the user.

The processed data from the processing layer are fed into the *Neural Network*, which determines the person that has been identified, and sends the result back to the robotic layer through RTS.

6.1. Design Level 0

The software system works along with a robotic interface. Camera and microphone data flow from the robotic interface into the communication package. From the communication package the data is distributed throughout the processing layer. Upon completion of the processing of data the communication package receives responses and sends them to the robotic interface. See Figure 1.

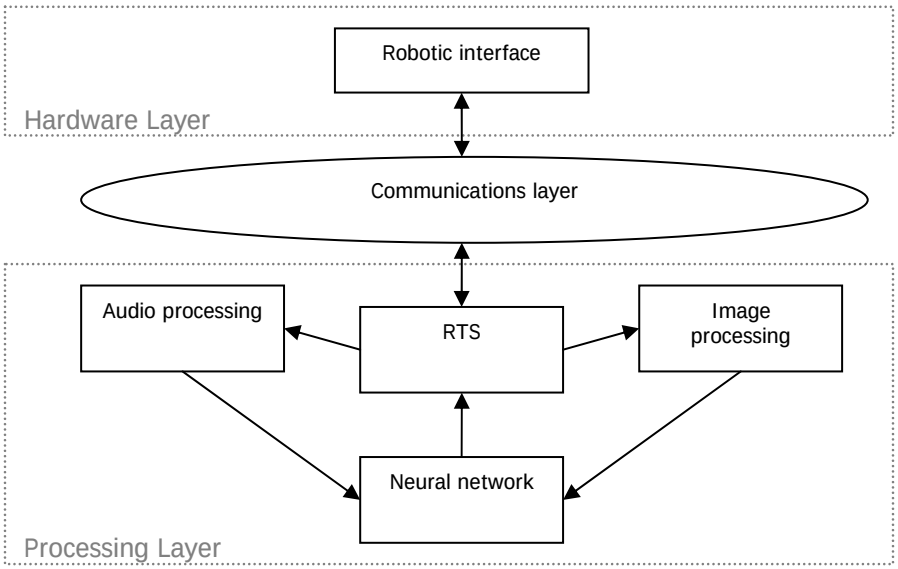


Figure 1:Overall system layout

6.2. Design Level 1

6.2.1. Robotic Interface

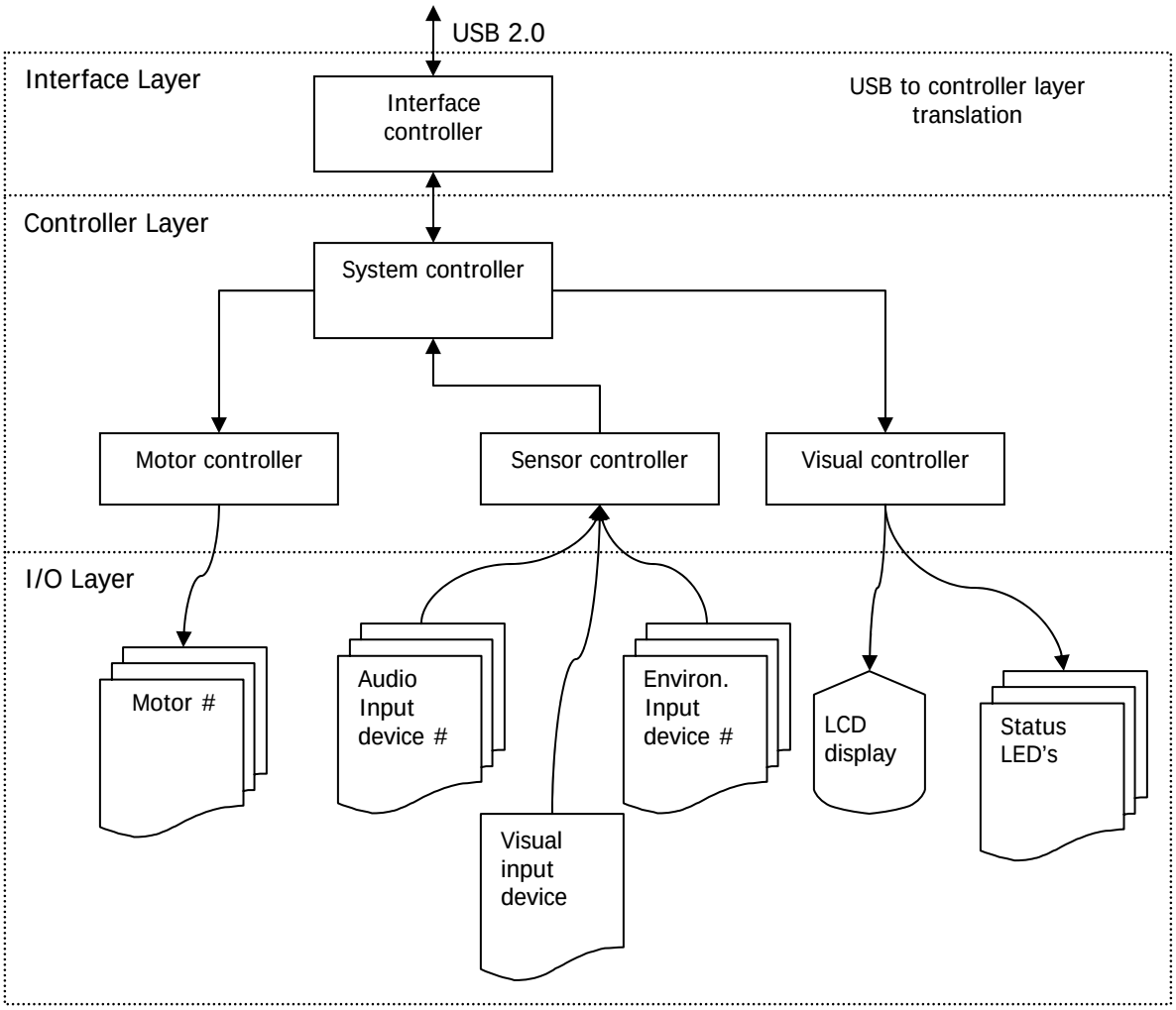


Figure 2: Robotic interface

The overall system has three main sections comprising of the interface for the robotic system. The interface layer is primarily responsible for encoding the USB packets into a central data stream and sending them to the processing layers via the communications layer. The system controller is responsible for dividing the allotted resources of the USB bandwidth stream into each of the sub components which may require access to this. This controller then sends the needed information streams to the Input/Output layer which consists of the motor controller circuitry, Sensor controller circuitry and lastly the Visual output controller.

6.2.2. Communications layer

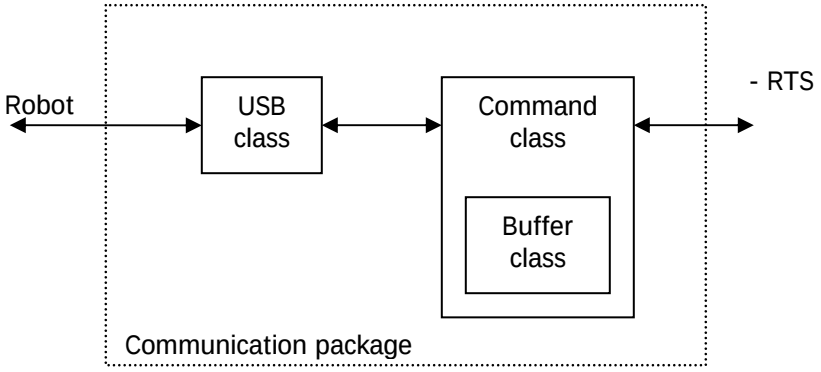


Figure 3: Communication package

The purpose of the communication package is to control the flow of data between the robotic interface and the processing layer, as well as communication between processing packages. The robotic interface sends both audio and image data via the USB connection to the communication package. The data is then stored in a lock and key buffer and distributed to requesting processes. Robotic responses are queued into the communication package and sent to the interface when the upload transmission time is available.

The communication layer consists of three high level classes. The classes work together to allow all necessary communication between packages. The USB's purpose is to communicate information to and from the robotic interface using the USB protocol. Image and audio data will be received by the communications USB class. The USB class will also output response commands to the robotic interface. The Command class receives incoming data from the USB class, stores the data onto the appropriate buffer, and then sends it, when called, to the appropriate requesting processes. The command class determines which aspects of the robotic interface must receive the response commands. The Buffer class is where the audio and image data is stored. The Buffer class will store a set amount of data and utilizes a lock and key protection to avoid data collisions

6.2.3. Real-time control system(RTS)

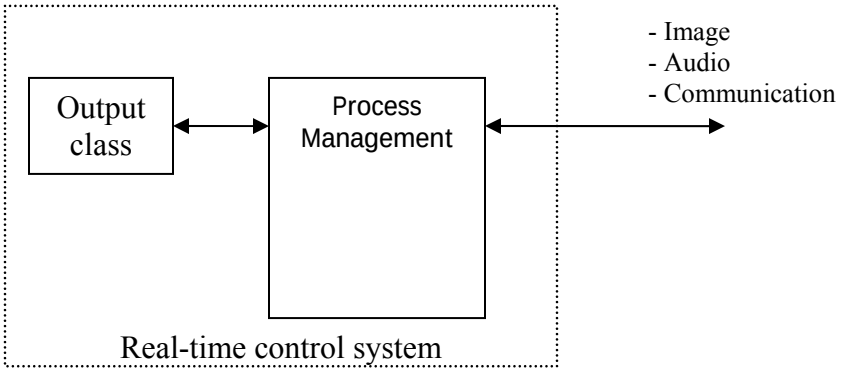


Figure 4: Real Time Control System

The real-time control system (RTS) is the overall management system of the running processes. The RTS is also responsible for determining the appropriate robotic response after user authentication. The scriptReader class is designed to allow the system to be improved upon for future applications. The Process management class is in charge of maintaining audio and image processing threads. Process management is an interface for both the audio and image package to communicate to the communication package. Finally, the script class is used to read then parse through a series of user commands in a predefined script file.

6.2.4. Audio processing

The data coming from the microphone is sent to the audio processing layer via RTS. After the processing on the audio data, the results (FFT points, details below) are sent to the Neural Networks for identification, and the final result is sent back to RTS.

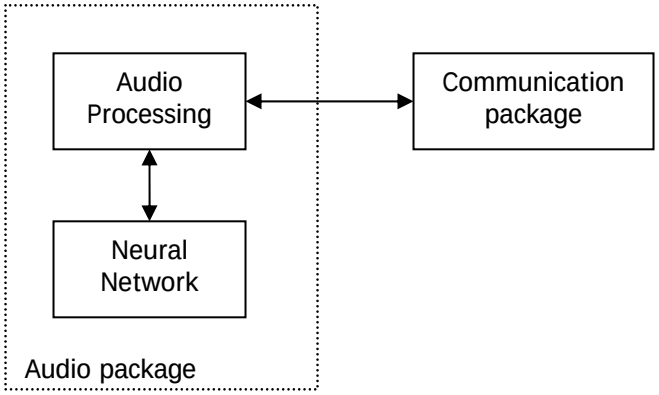


Figure 5: Real Time Control System

The audio processing functions of the system determine the user by specific voice tones through the use of neural networks. These specific tones are determined by recording digitized sound in .5-second increments that are of 8-bit quality, on the robotic interface. The digitized sounds are stored in an array by the buffer as points of amplitudes. The audio system calls a function and receives a pointer from the buffer class for these points. The points are then analyzed to find the top 80% (as signified in Figure 6 with the red lines) of the sound. The first point that is found, which is not the max, is used as a key point (as signified in Figure 6 by the green X). Then 1/600 of a second is used (approximately 40 points) to allow a Fast Fourier Transform (FFT) to be performed on them. The 40 points are stored in an array along with 984 points padded at the end in order to receive 1024 points from the FFT. From the result, the first 400 points are sent to neural networks as absolute values. Figure 6 shows a visual graph of this idea.

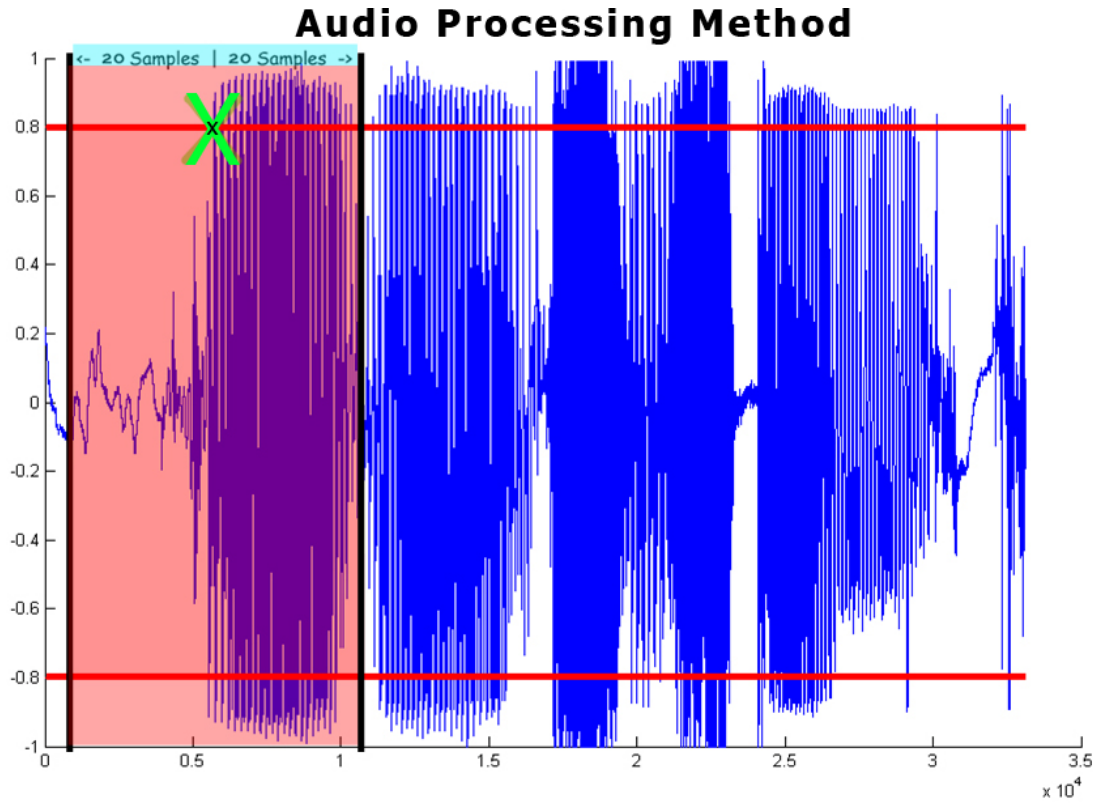


Figure 6: Visual graph of audio algorithm

6.2.5. Image processing

Like audio processing, data from the video camera is sent to the image processing module. Processed data from the image processing layer is sent to the Neural Networks for person identification. The results are sent back to the RTS.

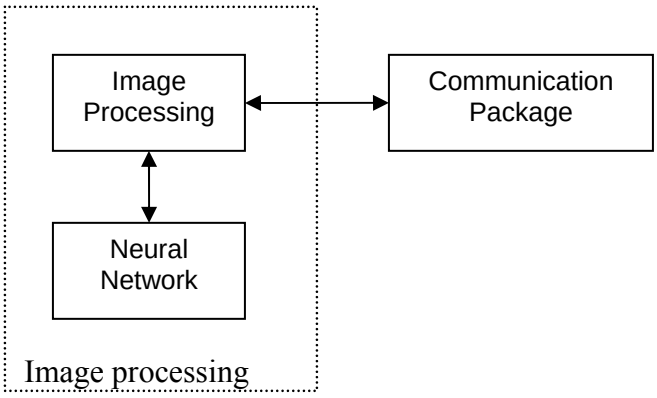


Figure 7: Image processing

The image-processing module uses pictures obtained by the robotic interface to determine the user. The main purpose of this layer is to grab the image, detect the face of the person and finally extract the distinctive features of the face for further computation with a neural network (details on it can be found in design level 2).

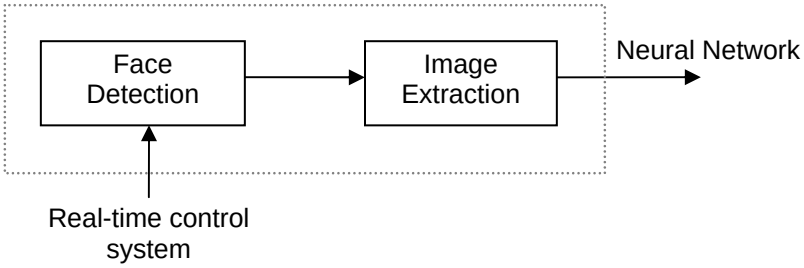


Figure 8: Image process algorithm

6.2.6. Neural Network

The neural network's main purpose is to take inputs from either audio or image processing and apply those inputs to the network of neurons. Using the generalized model of a neuron, the system is able to process the inputs and return a solution using previous training sessions. In the case of the audio, the system will use a built in training algorithm to allow the neural network to "learn" the audio from the users. This can be done because of the small amount of information that is being applied to the network. Image processing is much larger than audio processing and therefore does not have the liberty of utilizing unsupervised training. The Image neural network must be trained previously to its use by the 5 primary users that are defined in the requirements documentation.

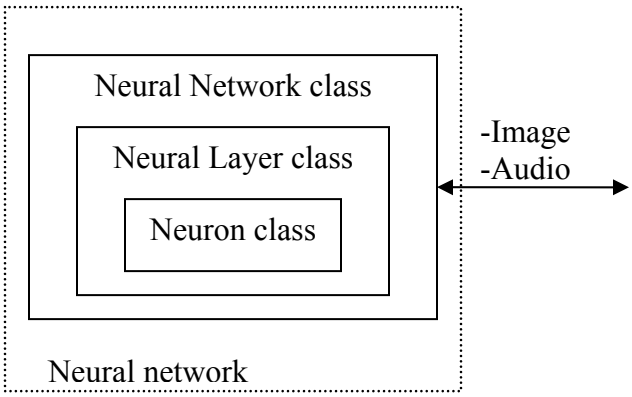


Figure 9: Neural Network model

6.3. Design Level 2

The interface layer translates the command bytes into system commands and function calls of the controller layer. The audio and video aspects of the hardware interface layer are to be processed by standard third party devices. These consist of a USB driven audio sensor, a USB driven camera device, and a USB driven LCD display. These devices are connected in conjunction with the USB servo controller assembly. They are then addressed using the provided third party drivers provided with the associated parts. This device is constructed using

a conventional USB 2.0 hub, and has relevant subcomponent connections attached to the hub to reduce this discussed data-flow to a single stream.

The interface layer also has a power feedback to the status LED's to indicate the subsystem that a USB cable has been connected and that the main power feed has been connected. In addition the system powers on a cold-cathode light to indicate the power to the system is applied on the front panel.

The sub-system elements for audio and video consist of a Logitech Quick-Cam zoom. This device was chosen for its support in Linux, common availability, and inexpensive costs. This device connects via the interface layer and communicated to the processing layer with relevant collected information.

The RS232 communications is converted into a USB stream using a Baro USB to serial adapter. This adapter was chosen specifically for its low cost and Linux driver support.

6.3.1. Control Layer

This layer is exclusive to the Motion. This layer functions to translate the inputted Motion command into a described low-level control PWM signal for the servo control systems. This is generated with a centered square wave and pulse width will effect overall servo position. Thus, the signals are generated with a set of internal timers. There is a master timer with predefined high speed resolution that will call a function with each tick. This function compares a position register of each sub servo motion system to calculate when the outputted control bit is to be changed. This has been implemented using a mini SSC II servo controller assembly from Scott Electronics. This assembly allows RS232 communication to the connected controller and is a open loop (no feedback) control model. This gives acceptable response time while taking as little resources in bandwidth as possible. The communication protocol is discussed in length in the communications layer details.

6.3.2. IO Layer (motion)

The IO layer will control the motion based on output of the control system. Here low level logic signals are translated into Motion thru the use of the servo motor control system respectively. The servo motor control system translates the PWM signals from the low level logic into a position with the help of a separate power grid circuit. The power Grid circuit uses voltage regulators for high and low power requirements of the subjected devices. In general, the power grid has a master power status for the purpose of restarting and powering down / up. This will be implemented with a conventional 12V switch capable of supplying all the associated hardware. In the interest of safety, a fuse assembly is used to guard against excessive current usage. This assembly uses standard RC servo assemblies with the Fubata Servo protocol to perform the conversion to motion from the PWM signals.

6.3.3. IO Mechanical layout and configuration

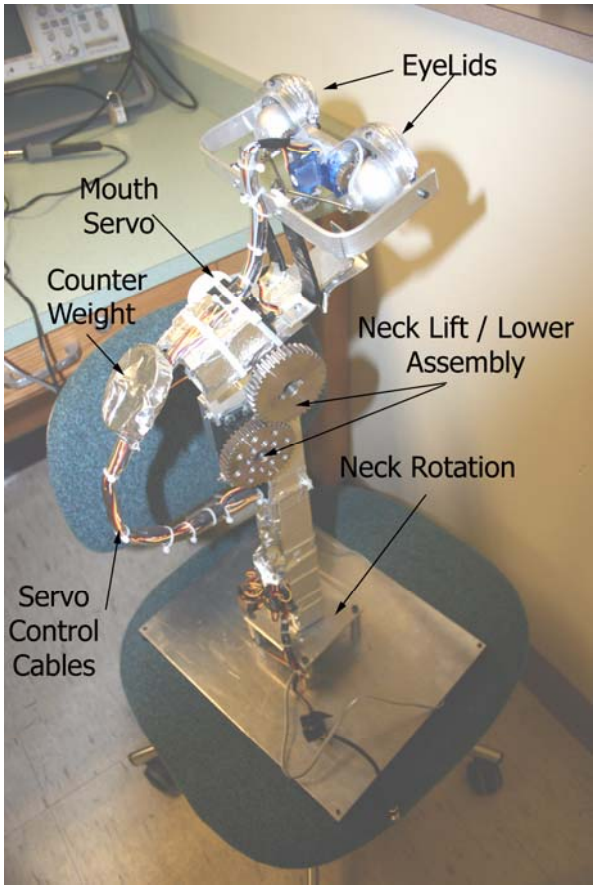


Figure 10: Robotic interface image 1

The mechanical layout (figure 10) was designed primarily with modularity in consideration. The visual appearance of the robot is mostly a result of this design method. The eyes of the robot (figure 11) are created mostly to give a personal feeling to the systems and to direct the gaze of the user towards the camera assembly. The mouth assembly is intended to be used in future applications as well as give the robot one more axis of motion for expansion. The remaining aspects of the layout are implemented to place the robot high enough such that the robot is average face to face height when placed upon a table.

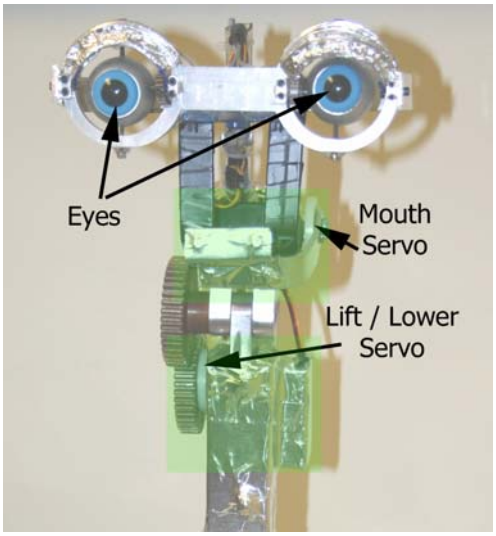


Figure 11: Robotic interface image 2

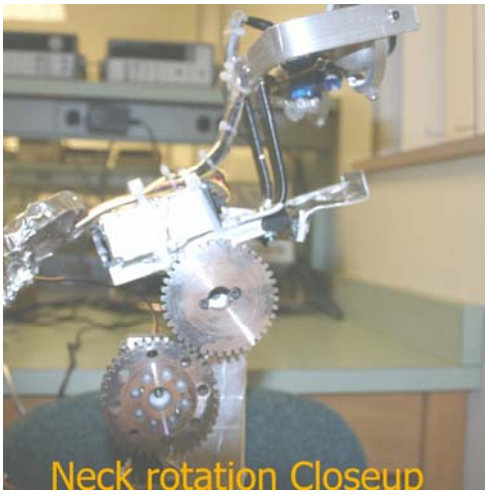


Figure 12: Robotic interface image 3

The Neck assembly (figure 12) has a base attached to it to facilitate movement left and right of the sensor and head assembly. The neck motor allows the head sensor arrangement and eye assembly to be lifted or lowered in conjunction to the directed location. Lastly the eye assembly (figure 13) consists of three servos to implement motion of independent eyelids and linked eyes. The eyes were backlit with blue LED's to further direct the gaze of the user towards the assembly containing the camera and microphone sensor.

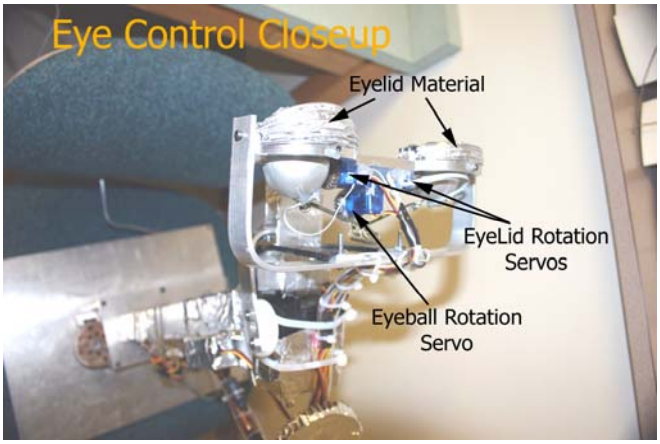


Figure 13: Robotic interface image 4

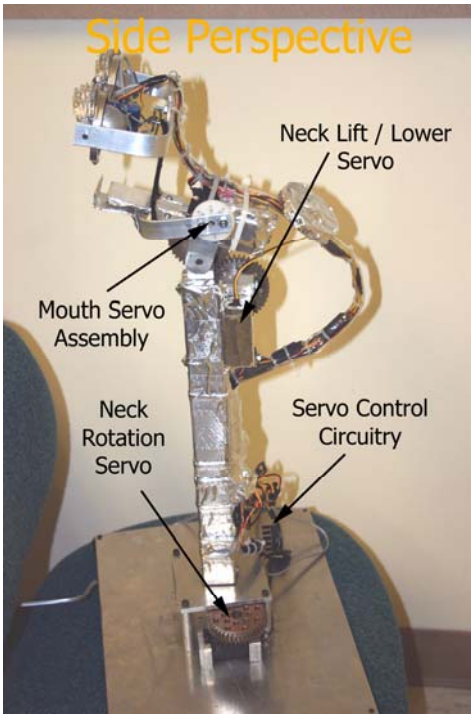


Figure 14: Robotic interface image 5

The base of the robot is covered with a protective case to prevent injury from rotating parts and hazardous temperatures. This case assembly houses the control circuitry hardware and gives a clean and finished look to the assembly. It has been hinged for easy access to the inner components for repair or upgrade. Lastly, the control cables of the assembly have been collected into a main “spinal cord” configuration and are run down the back of the assembly to give the robotic interface access to the control subset located in the base of the hardware unit.

6.3.4. USB class detail

The USB class is composed of smaller device classes used to receive and send data to its respective device. Communication to the devices is performed using a USB v2.0 connection. The USB class creates a high-level interface for the command class.

- Interface description
  - The USB class sends and receives data from the USB v2.0 connection.
  - The USB class provides calls for low-level device functions.
  - The USB class converts the robotic interface data into a form readable by the command class.
  - The USB class will continuously receive and send data from the robotic interface.

6.3.4.1. Restrictions/limitations

- The data stream sends approximately 9 images per second.
- Receive and send data is restricted to the limitation of the USB v2.0 connection speed.

6.3.4.2. Local classes

- LCDDisplay
  - This class is the interface to the LCD device and provides both text and image display functions
- MotorCtl
  - This class is the interface to the servo motor controller and provides a means to move separate servos within a specific range of motion.
- Webcam
  - This class is the interface to the web cam device and continually streams incoming image data from the web cam.
- Microphone
  - This class is the interface to the microphone device and continually streams incoming audio data from the microphone.

6.3.4.3. Performance issues

The performance of the USB class is directly linked to the performance of the USB communication speed as well as the device status. If the devices malfunctions then the USB class with be affected, and incomplete or invalid data will be received or sent.

6.3.4.4. Design constraints

Since the USB system is meant to be an interface between the hardware devices and the software systems, the device classes must respond quickly and accommodate for the device's own speed and function limitations

6.3.5. Command class detail

The command class is the communication interface for all software packages. This class represents the link between the software modules and the robotic interface. Its primary purpose is to interpret the output command data, push and pop data to the lock and key buffer, thread both the image and audio data gatherers, and send data to the respective packages.

6.3.5.1. Interface description

- The command class is in charge of sending and receiving data that will be processed via one of the system packages.
- The command class interprets the outputCmd struct and performs the respective device command.
- The command class writes the received data to the buffer class.
- The command class is the central control of the communication package. It allows for data to be dispersed to the processing layer.

6.3.5.2. Algorithmic model

The command class generates three continuously running threads; two of the threads are used for gathering audio and image data, the third thread is used to send output commands to the robotic interface. The image and audio threads have access to the commands buffer and push their data onto the buffer. The output command thread reads from a command queue and calls the respective USB device method.

6.3.5.3. Restrictions/limitations

- Command is restricted by the amount of data that is received from the USB devices. This restriction is based on the devices required data acquisition time.
- The output command queue is limited to 50 queued output commands.

6.3.5.4. Local data structures

- **`struct outputCmd`** – determines what the output commands purpose
  - **`int audio`** – If equal to 1 then audio output is specified.
  - **`int display`** - If equal to 1 then display output is specified.
  - **`int motor`** - If equal to 1 then motor output is specified.
  - **`char* stringVal`** – contains parameterized string data.
  - **`int posVal`** – contains parameterized integer data.

6.3.5.5. Local Methods

- **`void receiveData ( bool )`** – Appropriate data is sent to the calling class.
- **`void sendCommand (outpuCmd)`** – Sends output command information to the output queue the command information is interpreted via the struct.

6.3.5.6. Performance issues

The amount of data being sent between the robotic interface and the communication layer is extensive. To maintain quick performance the command and USB class are quick in their process time. This eliminates bottlenecking the data throughput.

To maintain updated output commands the output command queue is limited to 50 commands and when overflowed the queue removes the oldest command and pushes the new command.

6.3.6. Buffer class detail

The buffer class is where the audio and image data is stored. It consists of two arrays of equal size. The buffer class receives data from the command class through the public 'push' and 'pop' methods. The arrays store recent data and overwrite the old data while preventing data collisions.

6.3.6.1. Interface description

- The USB class is the source of the data to be stored in the array; however the buffer class does not deal directly with the USB class. Data is received through calls from the command class.
- The command class reads data from the buffer. The reading of data occurs when the Real-time control system requests either image or audio data.
- The image and audio processing modules both rely on the data stored in this class.
- Recent data will always be available to the command class for use.
- The buffer class communicates with the command class only.
- The command class is in charge or sending and extracting data from the buffer class.

6.3.6.2. Algorithmic model

The Buffer class consists of two arrays whose size will remain constant. The class creates a lock on the data element that is being read from. More information on the lock is discussed in local data structures (section 1.2.6.4).

6.3.6.3. Restrictions/limitations

- The size of the arrays will not be changed. This size is used, because it allows for the data stored in these arrays to be recent.
- The amount of data stored in each element in the array is limited.
- The audio data is stored in the audio array in .5-second intervals.
- The image data stored are 640x480 grayscale bitmap images.

6.3.6.4. Local data structures

- **`data imageBuf[]`** – This is the storage place for the images data.
- **`data audioBuf[]`** – This is the storage place for the audio data.
- **`int imagePos`** – next array element available for popping data off.

- **`int audioPos`** – next array element available for popping data off.
- **`int imageLock`** – next array element available for pushing data into.
- **`int audioLock`** – next array element available for pushing data into.

6.3.6.5. Local Methods

- **`void pushBuff(bool,data)`** – writes to the appropriate buffer while using the locking feature of the priority array.
- **`data popBuff(bool)`** – returns the most recent data from the lock & key priority buffer

6.3.6.6. Performance issues

The main issue for the buffer class is the timing of a receive call by the command class and writing to an element. Since an element is locked when a receive call is done it is important that at that instance the element is not written to. Simultaneous execution of the lock and write function will cause significant problems. There is timing done in order to avoid this situation.

The other issue is when to cease a receive call. If the data is taking to long to extract some class (either the buffer class itself or another, like a class contained in the RTS) halts this and resumes. This can occur if processing takes to long or if the transfer rates between the computers slow down.

6.3.6.7. Design constraints

- The buffer class is written in C++.
- The size of the array is five and will not change as mentioned in restrictions/limitations (section 1.2.6.3).

6.3.7. scriptReader detail

The scriptReader's main purpose is to receive the user's identity from the process management class and send out the correct preprogrammed response to the communications layer. The reason this is removed from the process management is two-fold. First, the process management is meant to control and maintain processes, not communicate information back to the robot. Second, later development can rewrite this class to allow the system's responses to become dynamic towards different users.

6.3.7.1. Algorithmic model

The scriptReader class is mainly a large switch statement with send commands to the communication interface

6.3.7.2. Restrictions/limitations

The only restrictions that are apparent are the command structure limitations.

6.3.8. Process Management detail

Process management is the main core of RTS. Its purpose is to maintain and manage the two main system processes, audio and imaging. The process management system is dynamic to the flow of data and data analysis is performed on a first come first serve basis.

6.3.8.1. Algorithmic model

The two running threads, audio and image, continually run a buffer 'pop' command. Only when data is available does the data analysis run and give output to the neural network. Since the system is running concurrently the systems are independent of each others performance load. However, due to the concurrent running the overall performances of both systems are degraded.

6.3.8.2. Restrictions/limitations

- The two running systems cannot allocate memory outside of their domain.
- Computation is limited to only being performed when valid data is available to analyze.

6.3.8.3. Performance issues

Process management is the core of the RTS and a therefore is built to high standards. This system is not a large system resource hog, however, it needs to remain in memory and complete its computations with little or no interruptions. Maintaining only two processes helps to remove a lot of the recurrent calls to the process manager.

6.3.9. Audio Processing detail

6.3.9.1. Interface description

- The audio processing begins with a call for audio data. The audio processing module will ask for audio data through the RTS. The RTS will then retrieve audio data from the communication package. This process does not directly communicate with the communication package.
- Upon completion of audio processing the results are sent to a neural network for analysis of the results.

6.3.9.2. Restrictions/limitation

- The audio processing module must get digitized sound in order for this algorithm to take place.
- Sound must be received and must have maximum amplitude of .8 (or about 300Hz) in order for the audio system to begin analysis.

6.3.9.3. Local data structures

- **`double keypointarray[1024]`** – This array will hold the first threshold (40 points) of the top 80% of the sound received by pointarray, and then padded with zeros to make a 1024 FFT.
- **`double *pointarray`** – This is a pointer to the array that is set to the array pointed to by the buffer class

6.3.9.4. Local Methods

- **`double *getdata()`** – Request data from RTS and receives a pointer from Communications class.

- **`void rftfft(int numberpoints, -1, double *keyarray)`** – Processes the FFT of the sent array keyarray (which is always pointskeyarray[1024]). The first parameter numberpoints is the number of points to be processed. The second parameter will process the inverse FFT if a 1 is placed. The results of the FFT overwrite the current data in pointskeyarray as all Real Numbers.
- **`Void sendnn(*keyarray)`** – sends the first 400 absolute values that are stored in pointskeyarray after rftfft is called to the neural network.

6.3.9.5. Performance issues

For the audio processing to perform at a highly successful rate, the digitized sound must filter out as much of the background noise as possible. If there is a loud ring in the background, the calculations will be inaccurate.

6.3.9.6. Design constraints

In order for this algorithm to work, we use the FFT. This speeds up the processing that must be done in order to extract useful features from a sound wave. Due to the fact that background noise will affect the validity of the calculations, visual identification are weighted more.

6.3.10. Audio Processing detail

6.3.10.1. Interface description

- The audio processing begins with a call for audio data. The audio processing module will ask for audio data through the RTS. The RTS will then retrieve audio data from the communication package. This process does not directly communicate with the communication package.
- Upon completion of audio processing the results are sent to a neural network for analysis of the results.

6.3.10.2. Restrictions/limitation

- The audio processing module must get digitized sound in order for this algorithm to take place.
- Sound must be received and must have maximum amplitude of .8 (or about 300Hz) in order for the audio system to begin analysis.

6.3.10.3. Local data structures

- **`double keypointarray[1024]`** – This array will hold the first threshold (40 points) of the top 80% of the sound received by pointarray, and then padded with zeros to make a 1024 FFT.
- **`double *pointarray`** – This is a pointer to the array that is set to the array pointed to by the buffer class

6.3.10.4. Local Methods

- **`double *getdata()`** – Request data from RTS and receives a pointer from Communications class.
- **`void rftfft(int numberpoints, -1, double *keyarray)`** – Processes the FFT of the sent array keyarray (which is always pointskeyarray[1024]). The first parameter numberpoints is the number of points to be processed. The second parameter will process the inverse FFT if a 1 is placed. The results of the FFT overwrite the current data in pointskeyarray as all Real Numbers.
- **`Void sendnn(*keyarray)`** – sends the first 400 absolute values that are stored in pointskeyarray after rftfft is called to the neural network.

6.3.10.5. Performance issues

For the audio processing to perform at a highly successful rate, the digitized sound must filter out as much of the background noise as possible. If there is a loud ring in the background, the calculations will be inaccurate.

6.3.10.6. Design constraints

In order for this algorithm to work, we use the FFT. This speeds up the processing that must be done in order to extract useful features from a sound wave. Due to the fact that background noise will affect the validity of the calculations, visual identification are weighted more.

6.3.11. Image Processing detail

6.3.11.1. Interface description

- Receives a pointer to the image data. This data is received only when a request for data is made to the RTS.
- The processing sends eye and nose profiles concatenated in one-dimensional vector.
- Profile vector is sent to the neural network for further analysis of results.
- The image module communicates with the RTS only and does not receive information until given by the RTS.

6.3.11.2. Algorithmic model

- There are two parts to the Image Processing Module.
  - Face Detection
  - Features (of the face) extraction

6.3.11.3. Face Detection

Intel's open source computer vision library (OpenCV) is used to train a classifier for face like objects, which is then applied to the input image to detect the "human face" like objects. The three key features of the face detection algorithm are Integral Image (which allows the features used by the detector to be computed very quickly), AdaBoost

(which selects a small number of critical visual features and yields extremely efficient classifiers) and “cascade” (combination of classifiers which allows background regions of the image to be quickly discarded while spending more computation on promising object-like regions).

“After a classifier is trained, it can be applied to a region of interest (of the same size as used during the training) in an input image. The classifier outputs a "1" if the region is likely to show the object (i.e., face/car etc), and a "0" otherwise. To search for the object in the whole image one can move the search window across the image and check every location using the classifier. The classifier is designed so that it can be easily "resized" in order to be able to find the objects of interest at different sizes, which is more efficient than resizing the image itself. So, to find an object of an unknown size in the image the scan procedure should be done several times at different scales. “

Rectangles are drawn around all the found face like objects and the closest person from the camera is considered to be analyzed further for features extraction.

6.3.11.4. Image Extraction

Template matching (correlation - a statistical technique which can show whether and how strongly pairs of variables are related) is used in Image Extraction layer for eye and nose recognition.

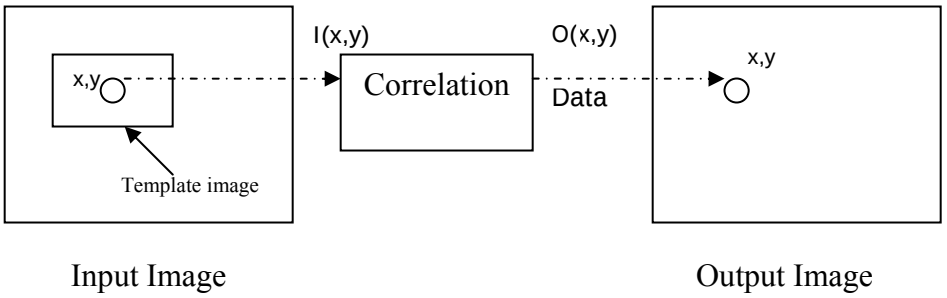


Figure 15: Correlation model

This matching process moves the template image to all possible positions of the input image and computes a numerical index that indicates how well the template matches the image in that position. This is shown in Figure 15. Match is done on a pixel-by-pixel basis.

The normalized profiles of the eye and nose are sent to neural net for person identification.

6.3.11.5. Restrictions/limitations

- The functions available to detect the face by openCV are fairly new and experimental.
- The accuracy of detection may vary on different background and lighting conditions.

6.3.11.6. Local data structures

The chief data structure for image processing is the Image, which is a struct. It contains information about the image such as: image height, image width, number of channels in an image, image header etc.

6.3.11.7. Local Methods

- **`cvLoadHaarClassifierCascade`** - This function loads a trained cascade of haar classifiers (for our case, face like objects) from a file or the classifier database embedded in OpenCV.
- **`detect_and_draw_faces`** -This routine detects face like objects in an input image and draws rectangle around them (A lot of modifications are made to this function by the imaging group).
- **`cvResize`** – This routine resizes the image using Interpolation method.
- **`cvMatchTemplate`** – This function implements template matching.
- **`cvMinMaxLoc`** - The function *MinMaxLoc* finds minimum and maximum element values and their positions in an Image.
- **`cvReleaseImage`** **–** Deallocates the memory that after processing.

6.3.11.8. Performance issues

- The object detection functions used by openCV has been initially proposed by Paul Viola and improved by Rainer Lienhart. According to the research at CMU and Yale University, these methods should yield the best performance among all the face recognition/detection methods that are currently available. Therefore the image processing should function accurately and efficiently.
- RTS must control the flow of data coming to this layer from the camera, so that the existing data do not get overwritten while it is being processed.

6.3.11.9. Design constraints

- The software OpenCV is new and the design of this module is done with the little knowledge known of its efficiency. There is a large learning curve using this software, but it is beneficial overall.

6.3.12. Neuron detail

The neuron is a model of a biological system that uses a variable number of inputs and a decision function to generate an output. The model allows for a system of weights applied to each input. The weighting of inputs allows the model to simulate a trained neuron. The neuron model is basic and consists of 4 main components; 1) the input vector, 2) the input weight vector, 3) the function bias, 4) and the decision function. Together the system generates an output based on variable inputs applied to the neuron

6.3.12.1. Interface description

- Receives vector data from the neural layer
- Outputs a single output in double form

- Allows the weights and bias to be adjusted through protected functions

6.3.12.2. Algorithmic model

The neuron is a mathematical model based on the biological neuron. The generalized model is

$$n = \sum \left(\underline{P} * \underline{W} \right) + b$$
$$a = \tanh(n)$$

The neuron sums the vector of inputs and their associated weights along with the neuron bias. The weights and bias are attained through the training sessions and are statically assigned. This summed value is then applied to a decision function which outputs a value dependent on the summed input.

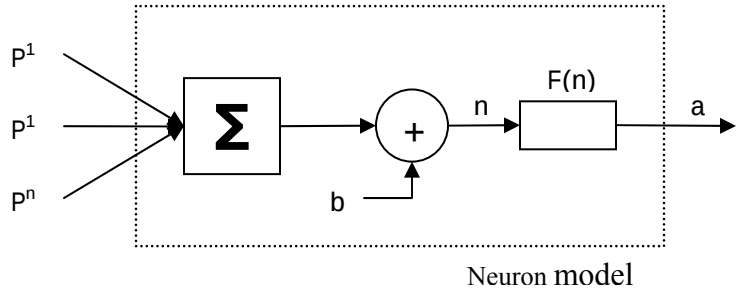


Figure 16: Neuron Model

6.3.12.3. Restrictions/limitations

- The neuron is limited to 100 inputs and one single output per neuron.
- Inputs, outputs, weights, and bias are limited to the range of $-1 \leq x \leq 1$

6.3.12.4. Local data structures

- **`vector<double> input`** - The vector of inputs to the neuron
- **`vector<double> weight`** - The vector of input weights
- **`double b`** - Function bias variable
- **`double n`** - Summation of all the inputs times their weights as well as the bias
- **`double a`** - Output from the decision function

6.3.12.5. Performance issues

With a large amount of inputs applied to the neuron, the task of summing the inputs and their weights becomes a huge computation. This in turn affects the overall speed of the neural network and overall memory usage.

6.3.12.6. Design constraints

Due to the large amount of memory needed for the vectors of both the inputs and the weights becomes a factor as the number of neurons increases with the number of inputs. Our system has been limited to 100 inputs to minimize the risk of overflowing the

memory due to the lack of space. Training is accomplished in the MATLAB environment and then finally ported to the C++ language when instantiated in the final product.

6.3.13. Neural Layer detail

The neural layer is a class to bundle the neurons into a useable structure for programming and layering. The neural layer class is a management class that generates the layer of neurons and makes sure that inputs are distributed among the neurons without invading into further layers. The class is also responsible for arranging the outputs into a usable vector to be passed to further neural layers.

6.3.13.1. Interface description

- Receives vector data from the neural network class
- Outputs a vector of the neuron's outputs

6.3.13.2. Restrictions/limitations

- The number of neurons is limited to 1000 per layer.
- Inputs and outputs are limited to the range of $-1 \leq x \leq 1$

6.3.13.3. Local data structures

- **`vector<double> input`** - The vector of inputs to the neural layer
- **`vector<neuron> nLayer`** - The vector of neurons
- **`vector<double> a`** – The vector of output from the neurons

6.3.13.4. Performance issues

Performance issues that affect the neuron are multiplied linearly within the neural layer class due to its accumulation of neurons. The computational requirements for the layer become larger as the number of neurons increases. This in turn affects the overall speed of the neural network and overall memory usage.

6.3.13.5. Design constraints

As with the neuron class, the neural layer class is constrained to the amount of memory that it can occupy. Constraining the use of no more than 1000 neurons per layer minimizes the risk of overflowing the memory due to the lack of space. Training is accomplished in the MATLAB environment and then finally ported to the C++ language when instantiated in the final product.

6.3.14. Neural Network detail

The neural network class is the overall container for the neurons and their component container class, the neural layer. The neural network class manages the inputs among the neural layers and generates the variable number of layers requested by the user.

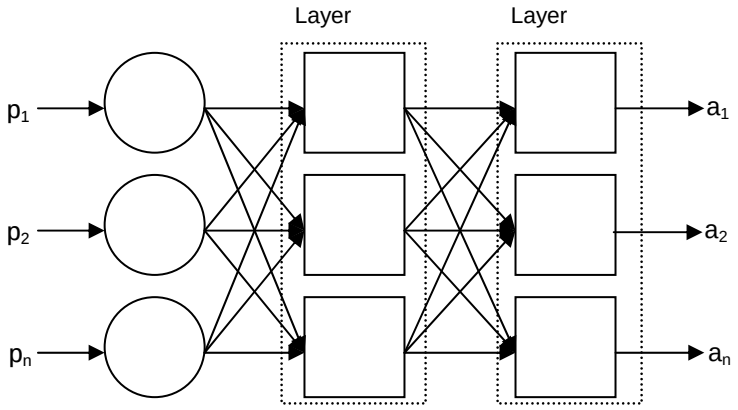


Figure 17: Two Layer Neural Network model

6.3.14.1. Interface description

- Receives vector data from the RTS
- Outputs a vector of the neuron's outputs
- The audio network implements a training algorithm to allow it to have unsupervised learning

6.3.14.2. Restrictions/limitations

- The number of neural layers is limited to 5 per network.
- Inputs and outputs are limited to the range of $-1 \leq x \leq 1$

6.3.14.3. Local data structures

- **`vector<double> input`** - The vector of inputs to the input layer
- **`vector<neuron> nNetwork`** - The vector of neural layers
- **`vector<double> a`** – The vector of output from the output layer

6.3.14.4. Performance issues

Training is a computation intensive process; due to our time constraints the training is supervised and completed previous to the system's run time on another system.

6.3.14.5. Design constraints

The neural network has the same issues as the neural layer, due its primary function of multiplying the number of layers within the network. Memory and computational speed are greatly affected by the number of layers present in the network. Training is accomplished in the MATLAB environment and then finally ported to the C++ language when instantiated in the final product. [Requirement 4.2.2]

7. Testing, Results, and Discussion

The VisionPSU project has five modules. Each one of these goes through extensive testing. The goal of the testing is to ensure that the system can detect the users specified by the software. The project also has a lot of integration testing to validate that a module works with the others.

7.1. Image

The results of the face detection, feature recognition, and graphs of eye profile for different people are described here.

Figure 18 depicts the results of face detection where all the face-like objects are detected within the rectangles. With the exception of Image 18(b) (it's a picture of the earth) and Image 18(d), all the images yielded close to 100% accuracy.

Figure 18: Results of Face Detection

Figure 19 illustrates the step by step process of template matching in which the input image is the face shown in Image 19(a). Image 19(b), Image 19(c), and Image 19(d) are the examples of features that are being correlated with the input image. The resultant maximum correlation values are shown in Image 19(e), 19(f) and 19(g), in which the central point of nose, mouth and eye are identified.

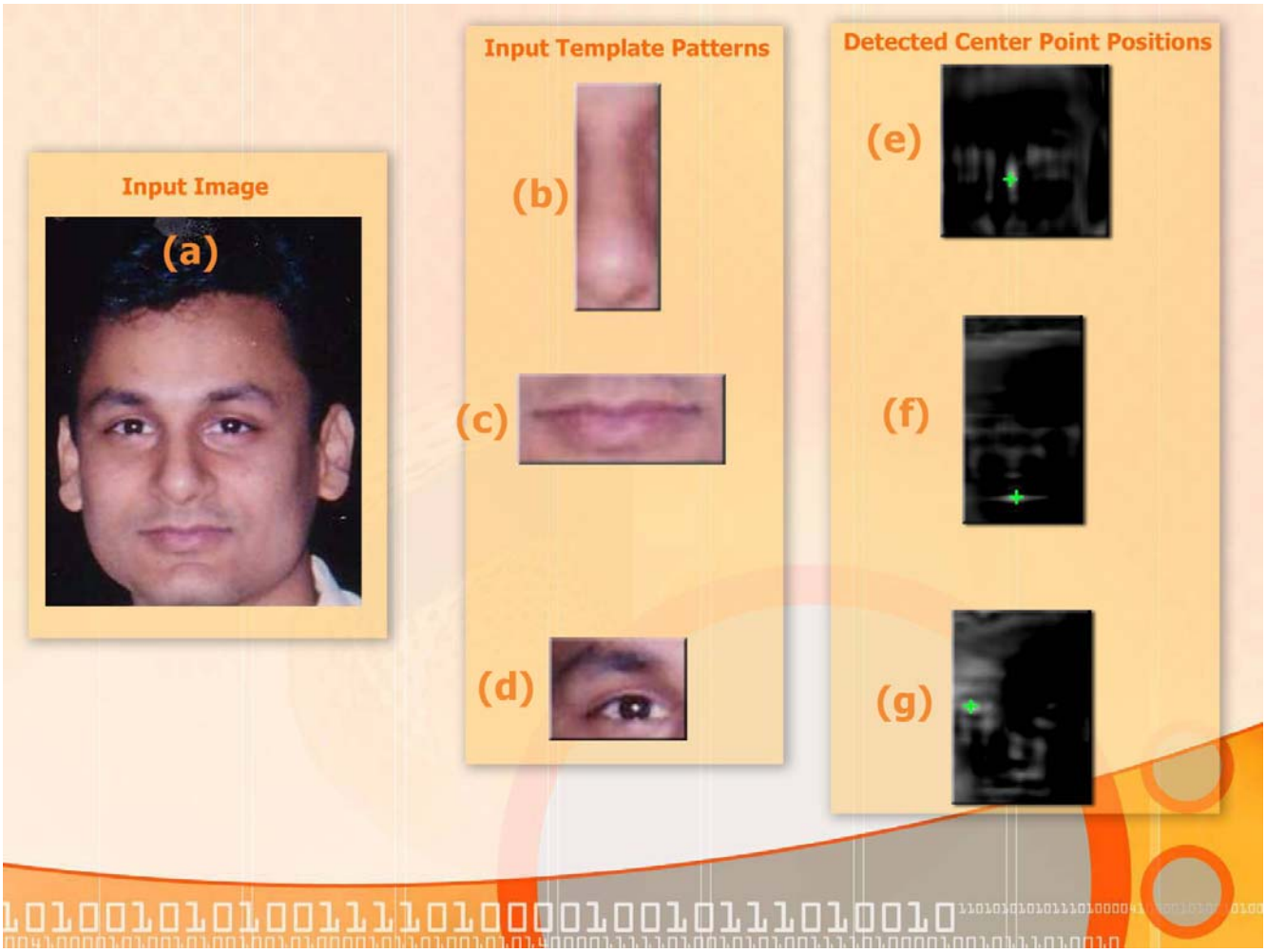


Figure 19: Results of template matching

The retrieved normalized profiles of the eyes and nose for different individuals are plotted in excel and attached below. For four different images of five people, the eyes and nose profiles seem to have unique pattern for each individual. Even though, the eye profiles for Troy (Figure 16) and Mohammed (Figure 19) looked the same, their nose patterns (Figure 22 and Figure 24) are completely different. This distinctive pattern of eyes and nose profiles for different people became significant train the neural network for person identification yielding an accuracy rate of close to 90%.

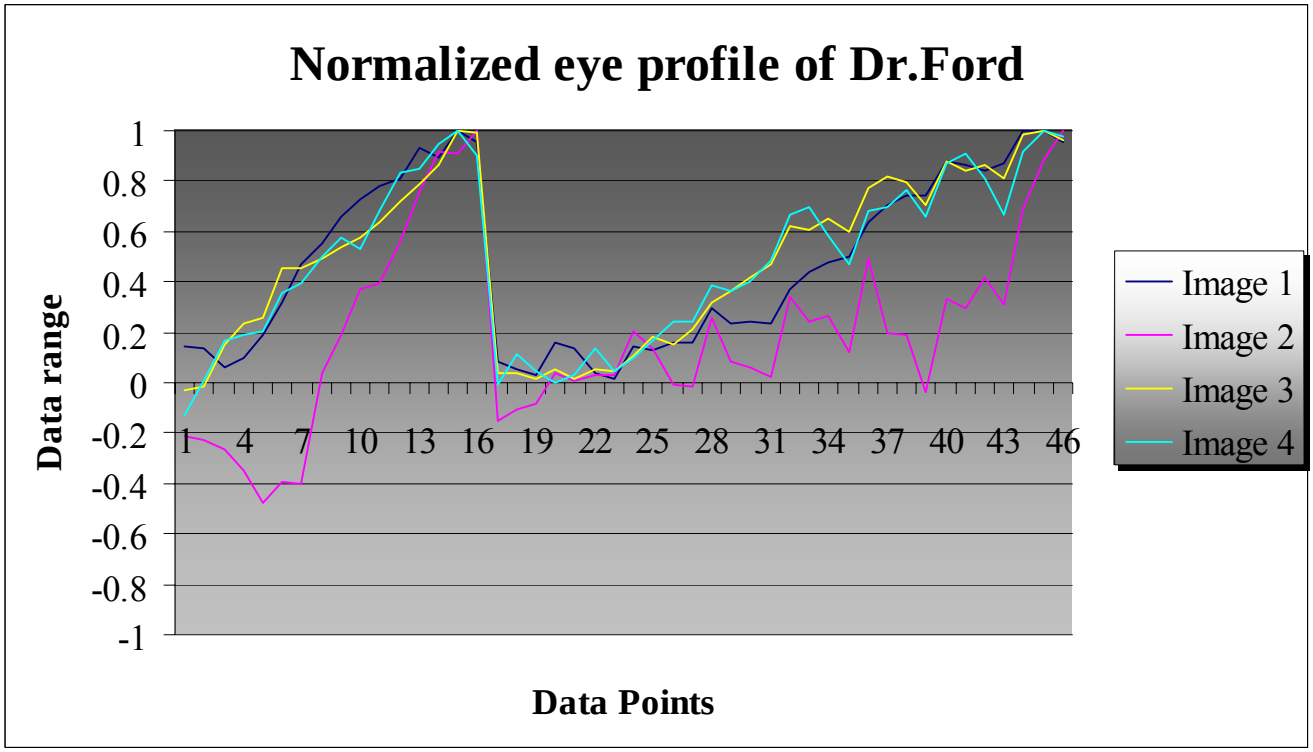


Figure 20: Normalized eye profile of Dr. Ford

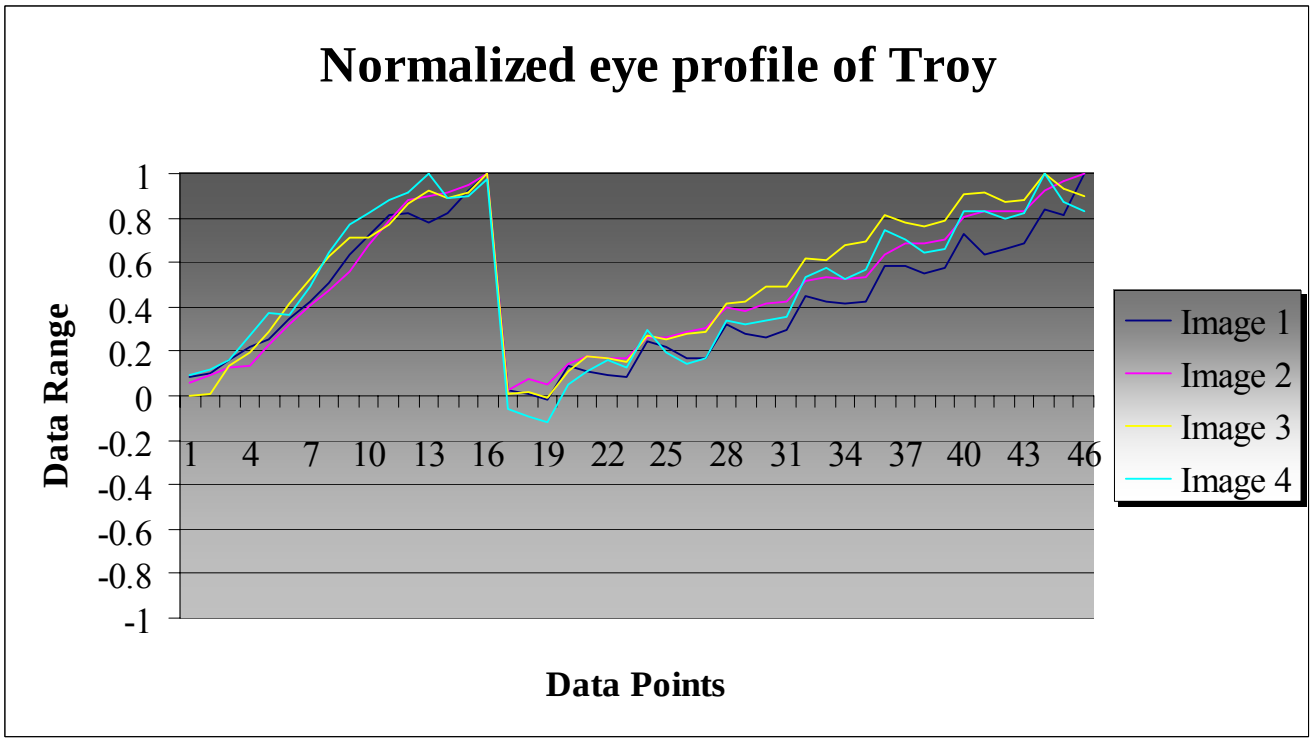


Figure 21: Normalized eye profile of Troy

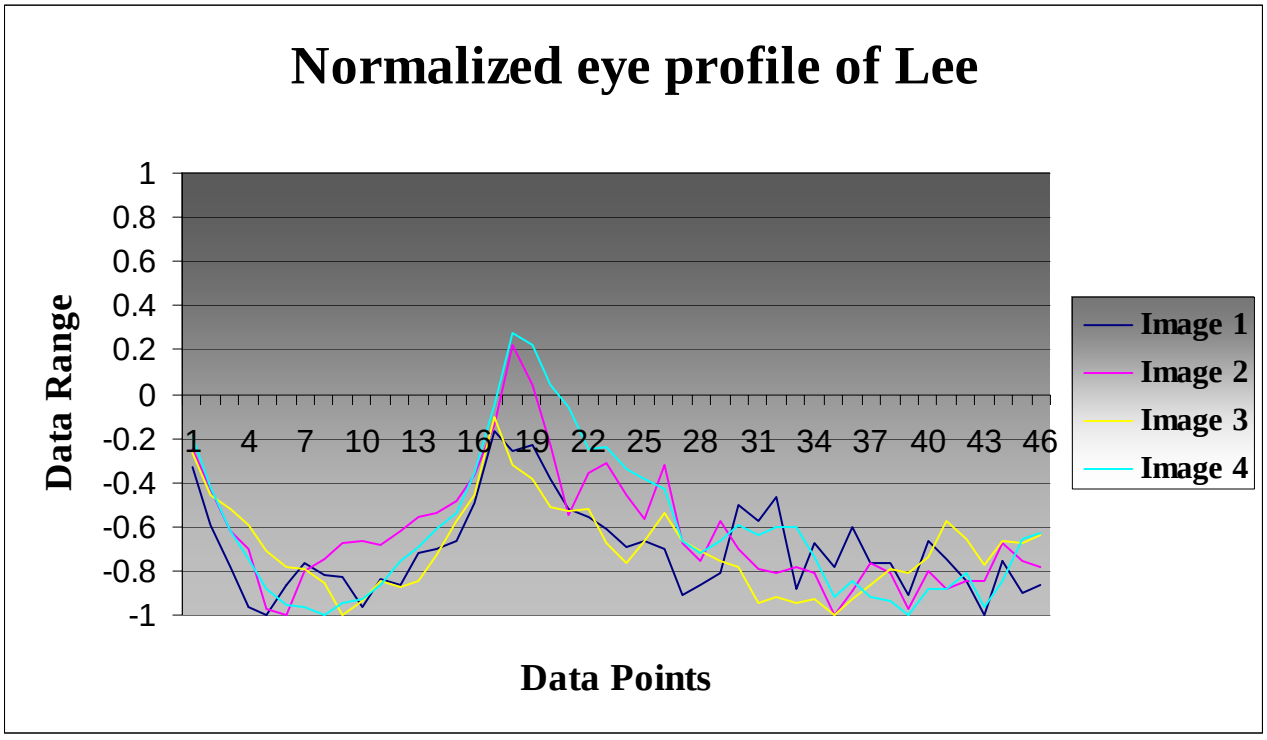


Figure 22: Normalized eye profile of Lee

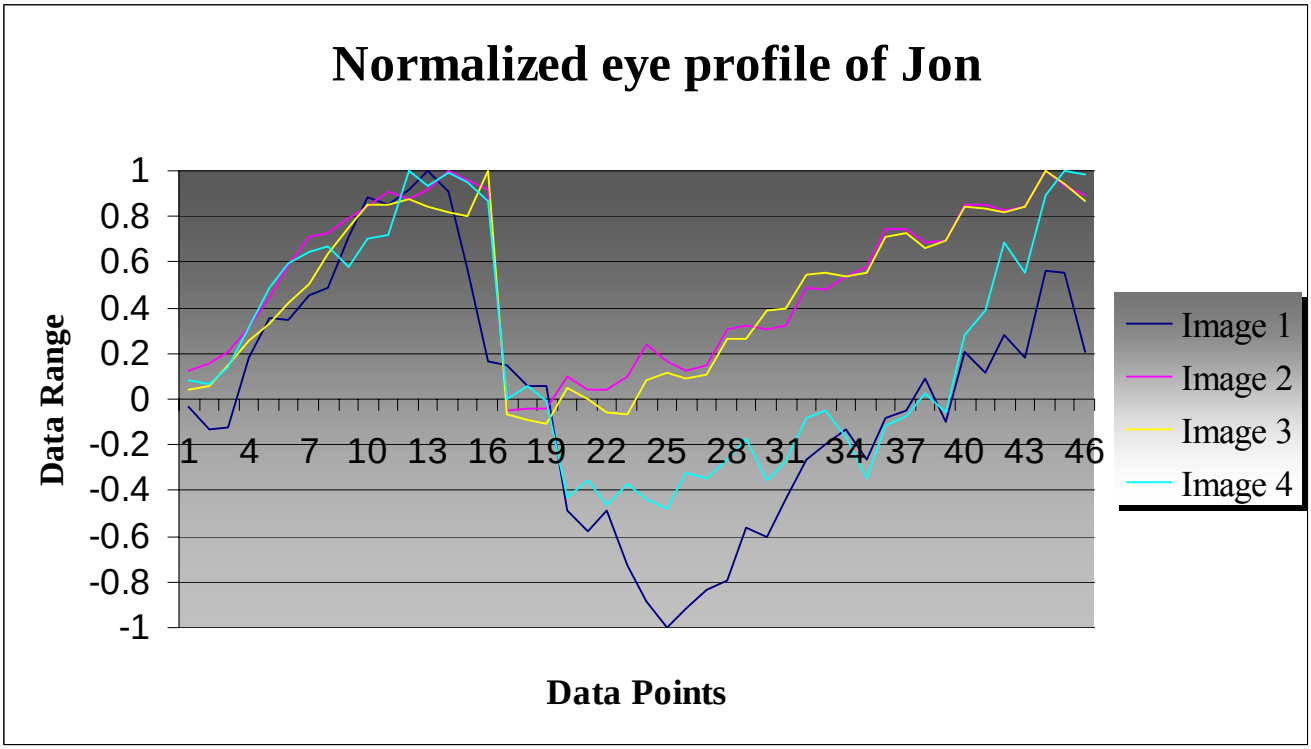


Figure 23: Normalized eye profile of Jon

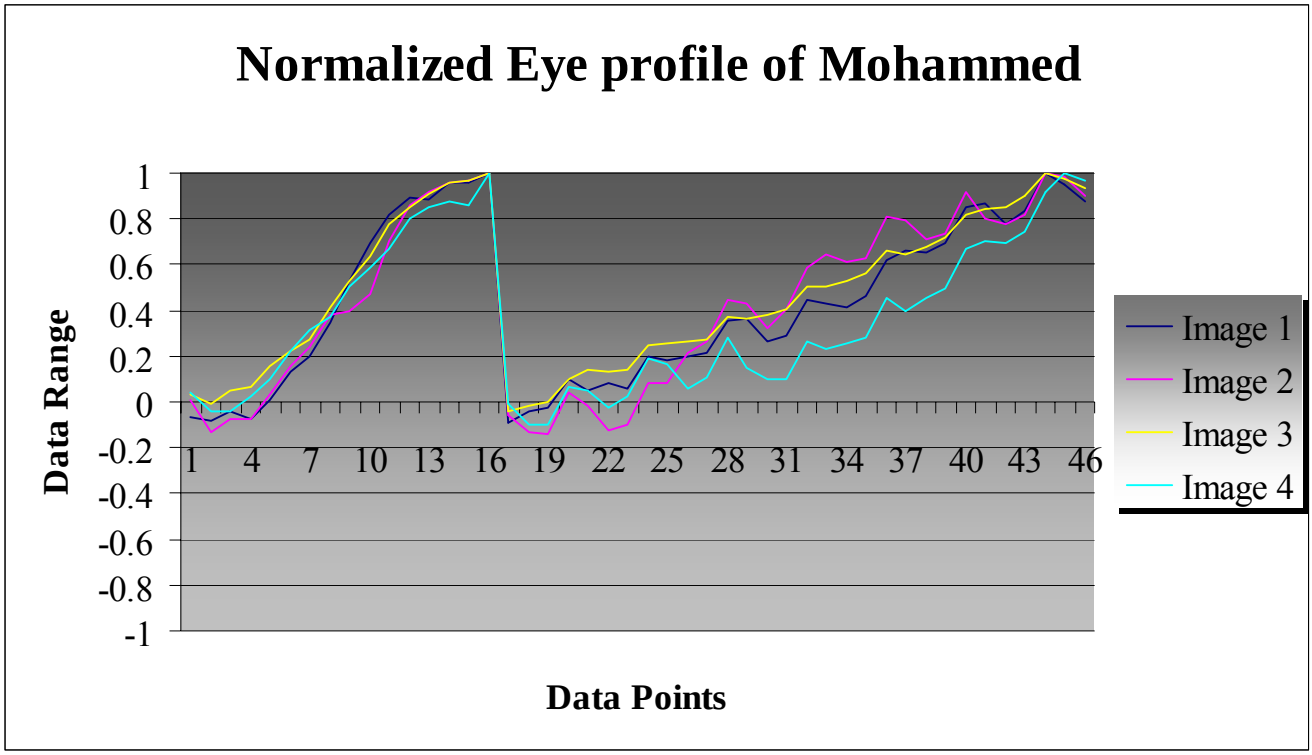


Figure 24: Normalized eye profile of Mohammed

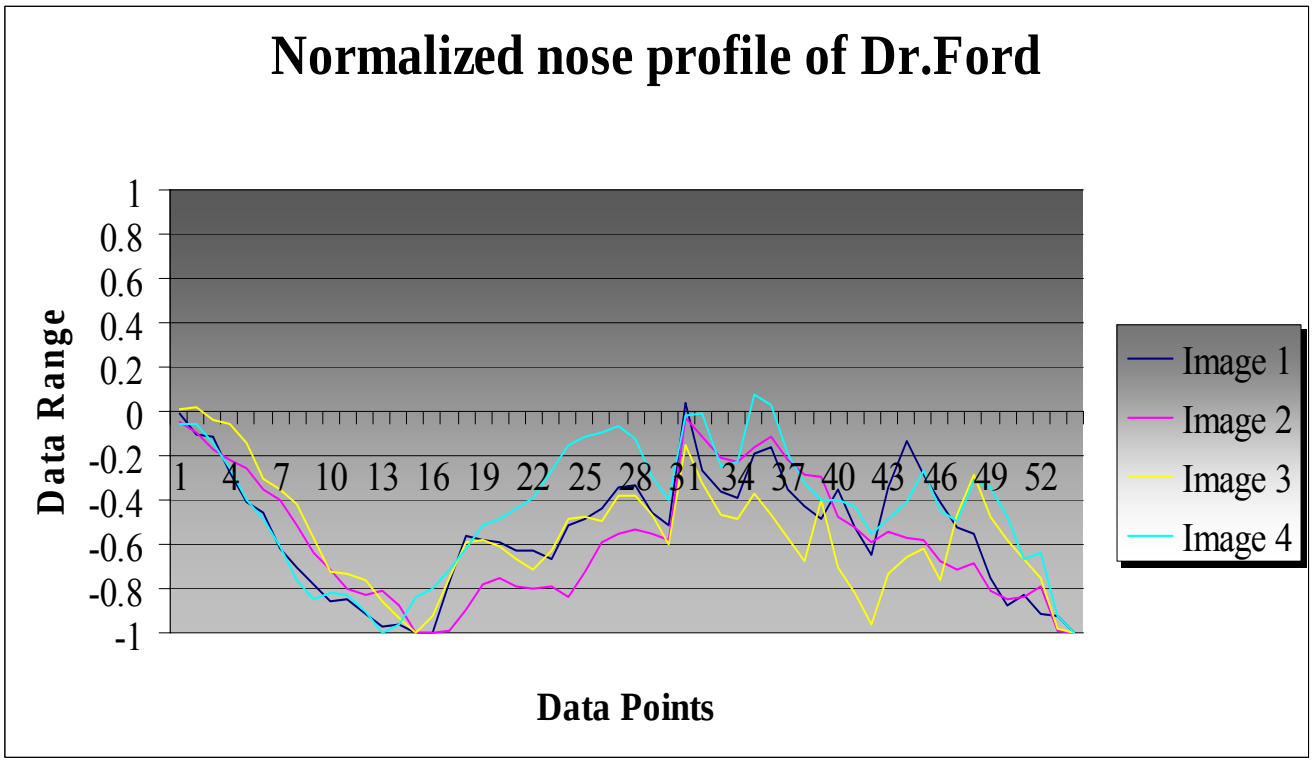


Figure 25: Normalized nose profile of Dr. Ford

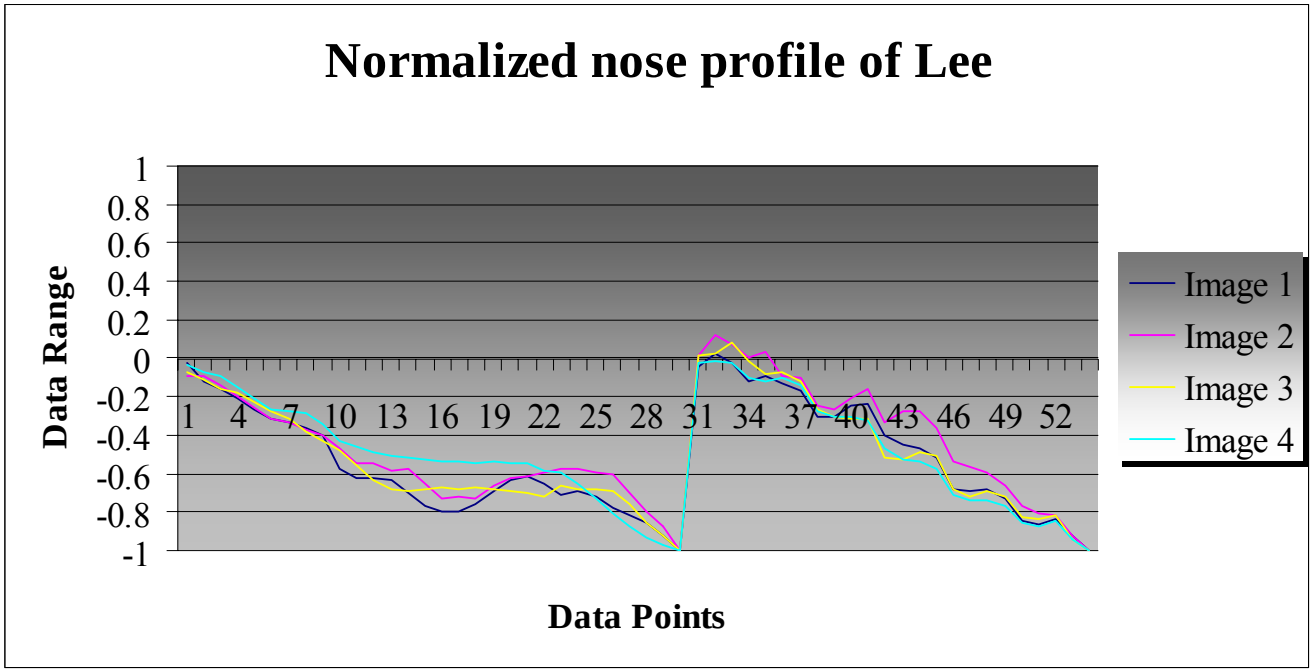


Figure 26: Normalized nose profile of Lee

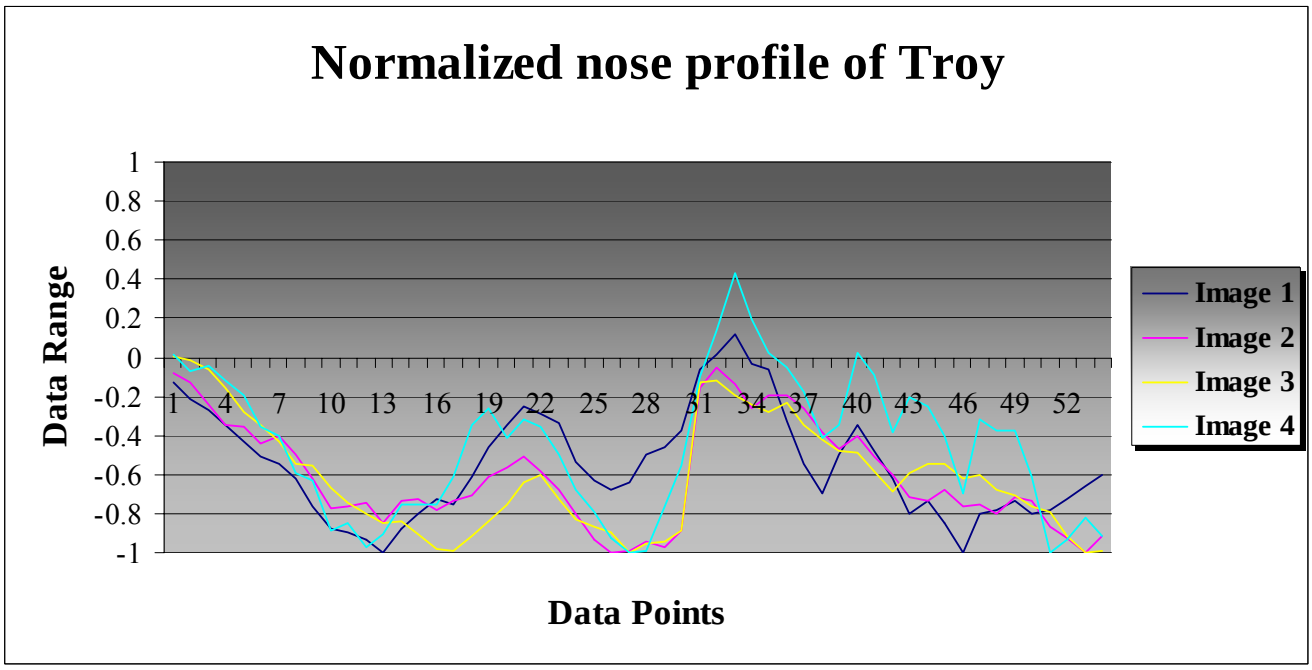


Figure 27: Normalized nose profile of Troy

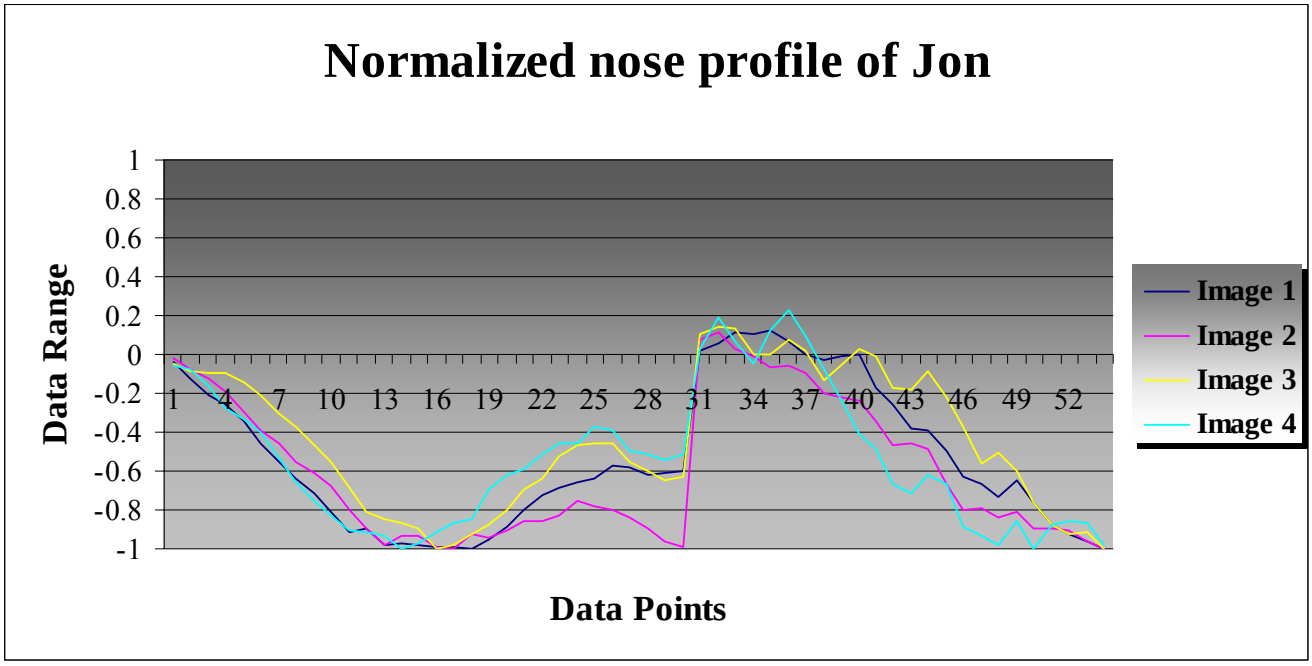


Figure 28: Normalized nose profile of Jon

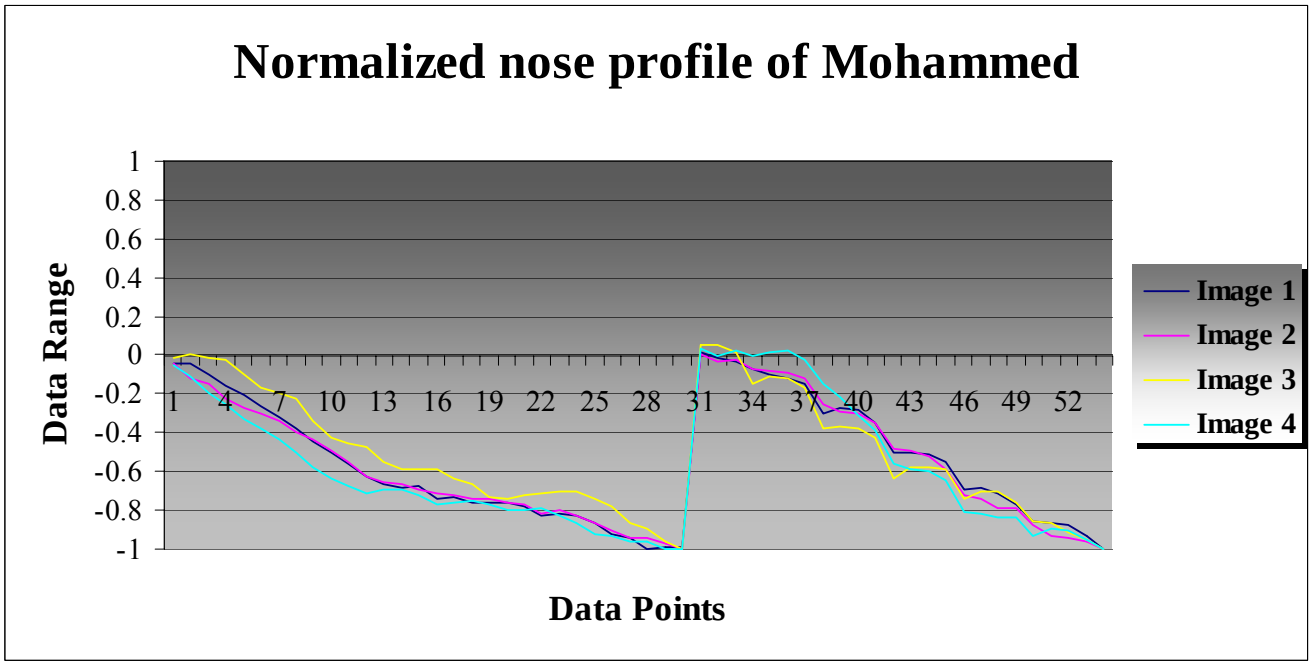


Figure 29: Normalized nose profile of Mohammed

7.2. Audio Results

Figure 30 displays the results of the FFT performed on the sample sets in Absolute Real form. The results show a clear correlation in the vocal prints. The darkened areas of each symbolize an absolute median, meaning it has the strongest correlation centralized about those points. The data was taken from a real-time test for audio from the microphone we are using. With the

full ten sample sets of each person, the correlation among the absolute median points becomes very strong. Full ten sample sets were used to train the neural-networks. The output in real time showed an accuracy rate of about 80%.

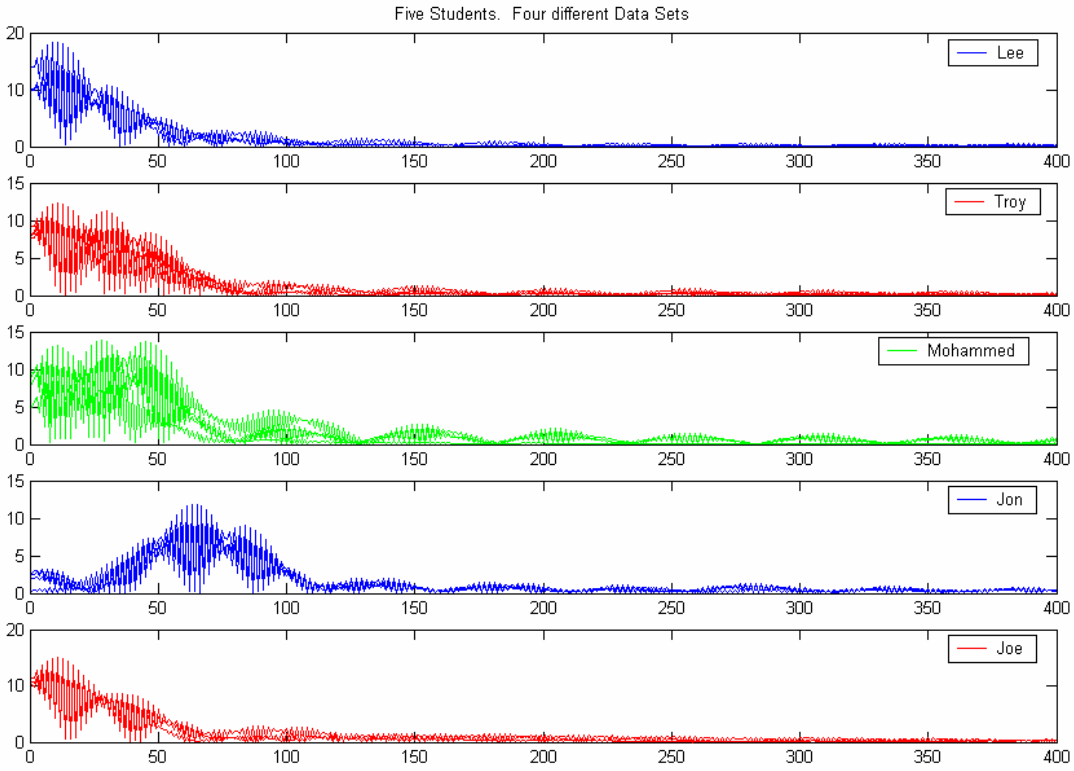


Figure 30: Graphs of 5 different students' FFTs at high level voice

7. Realization of Requirements, Constraints, and Standards

7.1 Realization of Specifications

The software utilized in this project was designed to run on the Linux operating system.

The video stream from the Logitech Quick Cam Zoom has been successfully captured at 10fps (frames per second), and the audio is sampled at 22 kbps.

The Robot incorporates standardized connectors for all inner connections. The RS232 connection is traversed via a USB to Serial adapter to pipe the bandwidth to the processing layer into one stream.

The robot is centrally controlled with one computer, and connected with a single USB connector to the computer.

Image computations are completed, for an input frame size of 640 X 480, in less than .7 seconds with an overall accuracy rate of more than 80%.

Audio computations are done, for an input sample of .5 seconds (roughly 10,000 points of amplitude) duration, in less than .2 seconds with an overall accuracy rate of close to 80%.

Neural Network runs parallel with audio and image processing and takes about .2 seconds to make a decision of person identification based on audio or video data.

The propagation time to get a response on identification of a person from the system approximately is a little over 2 seconds, which fulfills our engineering requirement on accuracy and efficiency.

All the computations are done on a conventional 2.2 GHz Pentium IV machine with 512 of RAM.

We have met the requirements of completing audio and image computations in less than 2 seconds on a conventional 2.2 GHz Pentium IV machine with an overall accuracy rate of more than 80%.

7.2 Realization of Constraints and Standards

7.2.1. Economical

The overall development cost for this project turned out to be approximately $722.40, which is more than the initial funding made available by the university. Among the total expenditure, $200 came from the resources that were previously owned either by the team members or university. Details of the expenses can be found at the development cost section.

Environmental

The system's accuracy is directly proportional to ideal environmental condition. The imaging system works the best in ideal lighting condition with user directly looking at the camera from a reasonable distance (3-4 feet). The audio system is susceptible to background noise and works the best if the user is close to the microphone.

The testing and final demo was conducted in the Engineering building classroom under ideal condition to yield expected performance.

7.2.2. Ethical

PWCX Logitech video decompression modules are used to decompress the captured video image in Linux operating system. These modules are free to try as long as not used for business purposes.

7.2.3. Health and Safety

Unit has a maximum input voltage of 11 volts for the control circuitry, and 5 Volts for the servo motors. Harmful moving assemblies (IE. Gears / Linkage arms) are held in closed containers, or have low torque limiters on the servo systems to prevent injuries.

7.2.4. Manufacturability

Main circuitry is fabricated on a PCB (printed circuit board) and requires minimal soldering. All connections are standard and conventional tools can be used to interconnect all main components. Standard adaptors are used to minimize special parts. The system is built mainly from stock metals and assembled with bolts and adhesives. This reduces special tools requirements. Only one component (eye assembly) requires and special machining and it is only a two cut pass.

7.2.5. Political

The system does not require government approval or trade barriers.

7.2.6. Social

The robot is built with a human like look to symbolize improved platform of human-computer interaction. The robot is capable of moving its head based on the motion of the user and can move its eyes if necessary.

7.2.7. Sustainability

System software is designed to be modular for other teams to extend the work in the future. The feature extraction routines to detect eyes, nose, and mouth are modular and can easily be used to integrate emotion detection based on video data. Neural Networks are designed as such that adding a new person in the identification list can be fairly simply done.

7.3. Standards

The project uses and complies to all the following standards and connectors where relevant:

USB
RJ11 (serial)
RS232
Pin out headers / Ribbon / Pin connectors
MiniJack

Protocols

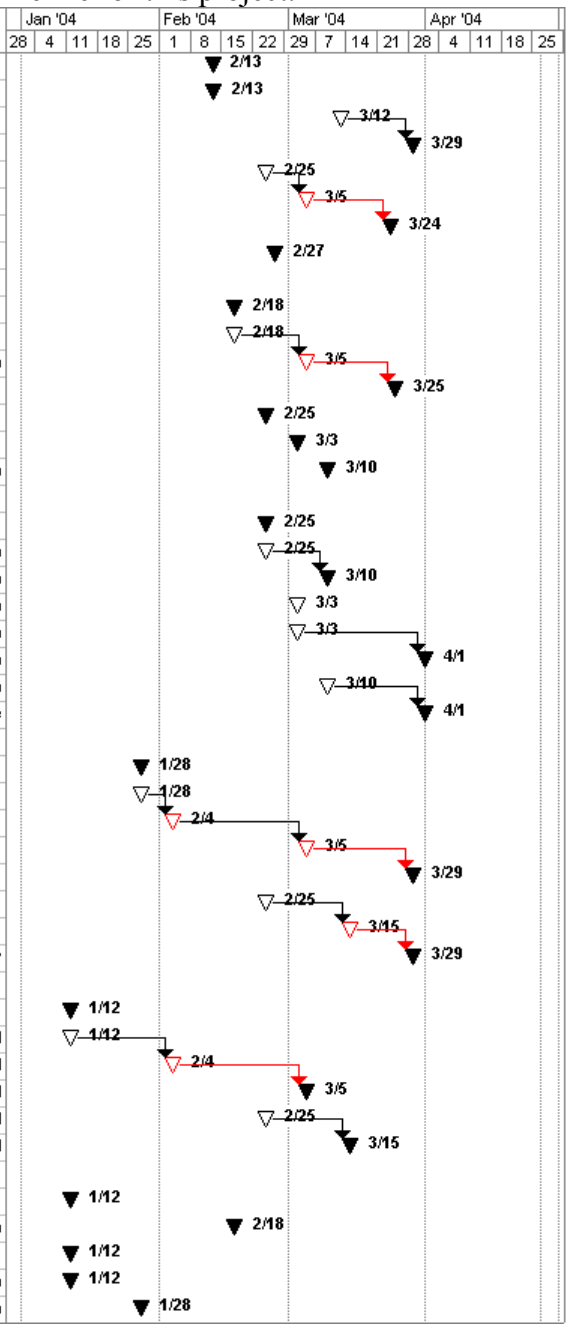
USB 2.0 1.1
RS232
Fubata PWM

8. Project Management Plan

8.1. Task Sheet

Below is the task sheet and GANNT chart for this project. It indicates each task and who was responsible for the completion of each one along with the timeline for this project.

ID	Task Name	Start	Resource Names
1	**Robotic Interface**	**Fri 2/13/04**	**Troy**
2	Structural	Fri 2/13/04	Troy
3	Structure extension	Fri 3/12/04	Troy
4	Structure extension	Mon 3/29/04	Troy
5	USB interfacing	Wed 2/25/04	Troy
6	USB interfacing	Fri 3/5/04	Troy
7	USB interfacing	Wed 3/24/04	Troy
8	Articulation control	Fri 2/27/04	Troy
9			
10	**Communication**	**Wed 2/18/04**	**Joe**
11	USB interfacing	Wed 2/18/04	Joe
12	USB interfacing	Fri 3/5/04	Joe / Jonathan
13	USB interfacing	Thu 3/25/04	Joe
14	Command	Wed 2/25/04	Joe
15	Buffer	Wed 3/3/04	Joe
16	Intercommunication wrapper	Wed 3/10/04	Joe / Jonathan
17			
18	**RTS**	**Wed 2/25/04**	**Jonathan**
19	Interrupt control	Wed 2/25/04	Jonathan
20	Interrupt control	Wed 3/10/04	Jonathan
21	Output control	Wed 3/3/04	Jonathan
22	Process communication	Wed 3/3/04	Jonathan
23	Process communication	Thu 4/1/04	Jonathan
24	Process management	Wed 3/10/04	Jonathan
25	Process management	Thu 4/1/04	Jonathan / Joe
26			
27	**Audio processing**	**Wed 1/28/04**	**Lee**
28	Experimentation	Wed 1/28/04	Lee
29	Experimentation	Wed 2/4/04	Lee
30	Experimentation	Fri 3/5/04	Lee
31	Experimentation	Mon 3/29/04	Lee
32	Processing detail	Wed 2/25/04	Lee
33	Processing detail	Mon 3/15/04	Lee
34	Processing detail	Mon 3/29/04	Lee / Troy
35			
36	**Image processing**	**Mon 1/12/04**	**Mohammed**
37	Face detection	Mon 1/12/04	Mohammed
38	Feature recognition	Wed 2/4/04	Mohammed
39	Experimentation	Fri 3/5/04	Mohammed
40	Experimentation	Wed 2/25/04	Mohammed
41	Processing detail	Mon 3/15/04	Mohammed
42			
43	**Neural Networks**	**Mon 1/12/04**	**Jonathan**
44	Audio Training algorithm	Wed 2/18/04	Jonathan
45	**Object implementation**	**Mon 1/12/04**	**Jonathan**
46	Object design	Mon 1/12/04	Jonathan
47	Redesign of NN	Wed 1/28/04	Jonathan



8.2. Summary of Accomplishments

The following is a summary of the accomplishments of this project up to date. Most of the accomplishments for this project were not an individual effort but a team effort.

8.2.1. Mohammed Hoque

Mohammed was the lead of Image Processing group. His researched the current existing face detection/recognition algorithms and recommended the team to use open source computer vision library (openCV), one of the most accurate and efficient face recognition tools, by Intel. He learned how to use the library fairly quickly and used them efficiently. He successfully transported his program into Linux from Windows environment without having any major difficulties. His face detection and feature extraction program yielded close to 90% accuracy, which made imaging system the major medium of identification in this project. Mohammed finished his tasks by mid March '04 providing more than a month time for the Neural Networks to train. Mohammed was also partially responsible for project documentation, system layout, and external presentations at the IEEE undergraduate conference.

8.2.2. Jonathan Lobaugh

Jonathan was involved in the design and creation of the system's neural network module, the real-time control system (RTS) and the overall programming integration. He was also involved in helping to design and layout the command module, as well as the robotic interface. Jonathan installed and maintained the Linux machines being used for the processing system. He also helped with project documentation and overall system layout.

8.2.3. Troy Tancraitor

Troy was responsible for all hardware construction and design. He facilitated creating the robotic interface from available parts and headed ordering needed additional components. Primarily, mechanical design was developed with available resources in mind. Only two components required fabrication from the machine shop, the eye assembly housing plate and the cover box assembly. Troy completed all assembly and assisted with testing and hardware interface software development. He primarily assisted in writing testing code for camera sensor access and helping configure the operating system parameters.

8.2.4. Lee Steen

Lee was the lead for audio processing. Lee had a hard time on getting audio processing to work since audio processing in real time, to our knowledge, is ground breaking. Almost every algorithm (well over one hundred) that the Lee tried was proven wrong by one of the sets of data he was working with, which was very frustrating at times. It wasn't until the last minute that the algorithm used in the system was found. The algorithm was found 2 full months after scheduled, which caused a rush to integrate audio into the system, which was tentatively schedule ahead of time for debugging purposes.

8.2.5. Joseph Mallozzi

Joe was the lead for the communication package. His role was to get the devices to communicate and store data. From this stored data the processing layer was to access the stored data. The robotic interface also receives commands from the communications layer. Joe role was to have these functionalities working efficiently. Joe also proofread a lot of the documentation for the project.

9. Development Costs

9.1. School funded expenses

Item	Quantity	Cost	Total cost
Camera and microphone	1	$60.00	$60.00
USB controllers	3	$5.80	$17.40
USB Hub	1	$15.00	$15.00
USB cables	2	$5.00	$10.00
		Total	**$102.40**

9.2. Team funded expenses

Item	Quantity	Cost	Total cost
LCD Display	1	$200.00	$200.00
Servo Controller	1	$70.00	$70.00
Glue/tapes/wires etc	N/A	$50	$50.00
USB cables	2	$5.00	$10.00
Power circuitry	1	20.00	$20.00
LEDs/ Cool cathode tube	N/A	$40.00	$40.00
Switches	1	$7.00	$7.00
Wire Cover	1 pack	$6.00	$6.00
Wire ties	N/A	$7.00	$7.00
		Total	**$420.00**

9.3. Previously owned

Item	Quantity	Cost	Total cost
Housing assembly	1	$0.00	$0.00
Fuses/Assembly	1	$0.00	$0.00
Computer	1	$0.00	$0.00
Mounting Hardware	1	$0.00	$0.00
		Total	**~$200.00 (excluding computer)**

10. Future work:

Most of our time was spent on exploring the existing technologies that suitable and realistic to implement within the given time frame for our design. By the time, our decisions were made final; the existing available time frame did not support integrating the features we wanted to. The immediate extension of this project could have been integrating emotion detection. Analyzing the mouth region of the face could have been useful to determine the basic emotions such as sadness, happiness, neutrality, madness etc.

Future additions to the robotic interface could be incorporating cameras into the eye assembly. In addition, work on adding a pan motion to the neck section of the interface, and incorporating a artificial material to the robots skeleton to represent skin could be perused.

11. Conclusion

The most important lesson learned in this project was that the requirement or design may change based on any unforeseen limitations during the design period, and the team should address those issues dynamically. We realized that proper research prior to the design phase allows the design process to be efficient.

We had a difficult time with audio processing due to the lack of understanding on its limitations in real time environments. After applying more than 25 algorithms, we finally developed an approach which gave us the anticipated results. This project proves that sometimes the best results can yield the design method that is most applicable.

It is impossible to predict all the problems that a design team may encounter in the process of development, but using the principles of designs, a well organized team can work through these problems and still achieve their goals. As expected, sometimes strong team work at the end can compensate for the lack of communication in the beginning.

The goal of this project was to facilitate effective, disciplinary and effortless interaction between human beings and machines. Therefore, we anticipate that this project will be a first step to interactive platform between humans and machines and will motivate other design teams to extend this research.

11. References

1. Viola, P, Michael, Jones, "Robust Real-time Object Detection", Second International Workshop on Statistical and Computational Theories of Vision-Modeling, Learning, Computing, and Sampling (pp. 1-2). Vancouver, Canada, July 13, 2001.

2. Schapire, Robert (1999) Theoretical views of boosting and applications [Online] <http://kiew.cs.uni-dortmund.de:8001/mlnet/instances/81d91e8d-dc15ed23e9>

3. Jianzhong, Guoping Qiu, School of Computer Science, University of Nottingham. "A color histogram based approach to human face detection" < http://www.cs.nott.ac.uk/~jzf/publication/vie2003.pdf>

4. Jang Kim Fung , The Chinese University of Hong Kong. "Face recognition by eigenface and elastic bunch graph matching" <http://www.cs.uwa.edu.au/~dwedge/docs/opencv/ref/OpenCVRef_Experimental.htm#aux_facedetection>

5. The University of Birmingham [Online] <http://www.cs.bham.ac.uk/resources/courses/robotics/doc/opencvdocs/ref/OpenCVRef_ImageProcessing.htm>

6. Temple University [Online] <http://www.cis.temple.edu/~latecki/CIS581-02/Project2/4>

Appendix A – Progress Reports

Team: VisionPSU

Progress Report: January 04

Accomplishment:

- **Audio:** The previous algorithm adopted by the group for audio recognition has been changed due to some skeptism showed by some faculty members who specialized in this area. Upon discussion with Dr. Hemminger, a new technique has been taken into consideration, which is now in the process of implementation.

- **Image Processing:** The Intel computer vision software, openCV, has been successfully installed and familiarity with the new platform of this software has been achieved. A few programs have been written to test the various functionality of openCV resolving all the linking issues.

- **Neural Network:** The neural networks have been fully built. The only remaining feature is to train/configure them based on the data and weight values.

- **Mechanics:** The layout of the robot is done. We are planning on sending the design to the technicians to build it.

- **Interface:** USB wrapper functions have been fully coded.

Adherence to Schedule:

So far, we are on schedule on building all the sub-components of the projects.

Issues:

None yet!

Plan:

In the following month, we plan to achieve the following things:

- The robot is fully functioning as described in the design document. However, the only function to add are the cosmetic features.
  Responsible individual: Troy

- We will finish interfacing the servo control board.
  Responsible individual: Joe

- We will have the audio algorithm working (tentative).
  Responsible individual: Lee

- Be able to detect a person's face in an image and, if possible, extract different features of it as well within this time frame.
  Responsible Individual: Mohammed

- Train the neural network depending on the audio and video data output (tentative).
  Responsible Individual: Jonathan

Team: VisionPSU

Progress Report : February 04

Accomplishment:

- **Image Processing:** A preliminary face detection routine has been written, which takes video images as input and draws rectangle around the face like objects. However, the accuracy of this outcome is not as we expected.
- **Mechanics:** Robotic system is built, extension to the system is underway

- **Interface:** Command class and buffer class are finished.

Adherence to Schedule:

Since the experimentation has been extended the schedule is somewhat behind.

Issues:

Image and audio experimentation NEEDS to be finished very soon.

Plan:

In the following month, we plan to achieve the following things:

- Get the USB chip working with the software
  Responsible individual: Joe

- We will have the audio algorithm working (tentative).
  Responsible individual: Lee

- Robotic system is being upgraded to the contain the LCD and LED systems
  Responsible individual: Troy

- Be able to detect a face like object with at least 80% accuracy according to our requirement and do the feature recognition.
  Responsible Individual: Mohammed

Team: VisionPSU

Progress Report: March

Accomplishment:

- **Image Processing:**
  - The face detection and feature recognition programs are working properly together.
  - The program returns a normalized array of the pixel values of the eye of the person closest from the camera.
  - The pixel values of eye for different individuals have been graphed in excel, which turned out to be different (and unique) from each individual.
- **Mechanics:**
  - All the mechanic parts are purchased.
  - Assembly has begun.
  - Preliminary motor testing has begun.
  - Testing for LCD has begun. We are still waiting for LED display.

- **Interface:**
  - Command class and buffer class are finished.
  - Preliminary communication with USB chip and LCD has begun.
  - Intercommunication between computers has been tested.
  - The operating system installation into the three computers has begun.

Adherence to Schedule:

With the exception of audio, all the separate pieces of this project are coming along good. We have one full month for system integration.

Issues:

We need to take a decision on whether we are going to give up on audio or not by analyzing the chances of getting it to work within the given time frame. This decision has to be taken by the end of this month.

Plan:

In the following month, we plan to achieve the following things:

Completions by Wednesday 3/24:

- Communications completed → DONE!
- USB Soldered onto the robot → DONE!
- Communications communicating with robot → DONE!
- All parts should be purchased → DONE!
- Final Audio approach decided
- Image communicating with robot.
- Clearly define RTS and everything it needs → DONE!

Completions by Wednesday 3/31:

- Robot mostly assembled (Troy)

- Audio system near complete (if applies) (Lee)
- NN testing with Image module (Jon and Mohammed)
- Fix any communication bugs. (Joe)
- RTS work.

Completions by Wednesday 4/7:

- Robot completed (Troy)
- Audio completed (Lee)
- NN testing with Image completed (Jon and Mohammed)
- Audio system communicating with robot. (Lee and Troy)
- Full RTS devotion if necessary.

Completions by Wednesday 4/14:

- RTS complete.
- Full system testing.
- More testing