REALISTIC INTERACTION WITH SOCIAL ROBOTS VIA FACIAL EXPRESSIONS AND NECK-EYE COORDINATION

by

SUMIT KUMAR DAS

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2015

Acknowledgements

I would like to thank Dr. Dan Popa for guiding and advising me during my Masters studies. I am very grateful for constant support and opportunities he provided. I had an enriching experience working at the Next Generation Systems lab. I would also like to thank Dr. Michael Manry and Dr. Frank Lewis for taking the time out of their busy schedule to serve my thesis defense committee.

I would like to thank and acknowledge Isura Ranatunga and Dr. Indika Wijayasinghe who mentored me and helped me develop an aptitude for research work. I have always learned from our discussions on robotics and their various possible applications. I would also like to thank Joe Sanford, Oguz Yetkin, Sven Cremer, Namrata Balakrishnan, Sandesh Gowda, Fahad Mirza, Rommel Alonzo and Abhishek Thakurdesai for their support.

I am humbled and grateful to my parents, Sunil Kumar Das and Namita Das, for their unconditional love, sacrifices and support. I would like to thank them for always believing in me and encouraging me to pursue my dreams. I am lucky to have the support of my fiancée, Ankita Sahu, who stood by me and supported my dreams without hesitation. I am grateful to have the support of my friends and family members who have helped me throughout my career. I would also like to thank Arun Pokharna, who has been my roommate since the day I have arrived in the USA, for being an understanding friend and making me feel at home away from home.

Special thanks to my best friend, Biswaprakash Sarangi, who believed in me way before I started believing in myself, and standing by my side when I needed him the most. The support and guidance provided by him helped me avoid failing high-school. I am privileged to have a friend like him.

May 8, 2015

Abstract

REALISTIC INTERACTION WITH SOCIAL ROBOTS VIA FACIAL EXPRESSIONS AND NECK-EYE COORDINATION

Sumit Kumar Das, M.S

The University of Texas at Arlington, 2015

Supervising Professor: Dan O. Popa

In this thesis, we report on research work concerning various aspects of interaction between an android head and its environment. The research aims at building a reusable framework of interaction through facial expressions and vision which can be used for a wide range of humanoid robotic heads. The framework contains specially programmed modules that enable the android head to effectively interact with its environment via facial expressions and neck-eye coordinated motion.

The object detection and tracking module assists the android head to detect and track objects and users through coordinated motions of neck and eye. The module has the capability of using multiple cameras in order to track a target. This helps for developing a wider coverage of tracking area. We present the work conducted in implementing and evaluating several controller algorithms to imitate human-like tracking in an android head.

The facial expressions learning and imitation module comprises different methods for generating facial expressions in the android head. This module uses batch training of neural networks and can automatically recalibrate facial expressions for the android, a very tedious and time consuming process. This helps in carrying out realistic conversations with users. During conversation, lip syncing capabilities in an android head is of utmost importance for making it human-like. In the thesis, we also discuss the necessity of having a proper lip sync module in place and the “McGurk effect” to be avoided during conversation.

Table of Contents

List of Illustrations

List of Tables

Chapter 1

Introduction

1.1 Motivation

As there is an increase in the use of robots, in areas such as manufacturing industries and hospitality, contact with humans has become unavoidable and is oftentimes desirable. Robots can be defined as intelligent electro-mechanical machines that can perform tasks, for which they were designed. There are different fields in robotics depending on application such as social robotics, industrial robotics, mobile robotics etc. Social robotics, which is our focus of study, requires a robot's appearance and actions to be as human as possible in order to get acceptance for widespread use.

Studies have indicated that robots looking like human beings may fall into the "uncanny valley" [1]. According to Mori, social robots, having facial appearance similar to a human, can become scary and there is a higher risk for such robots to fall into the uncanny valley if they are in moving state, as illustrated in Figure 1-1.

Studies also indicate that in order for android robots to look appealing to people, they need to have social responsivity and aesthetic refinement [2]. "The Path of Engagement" (POE) theory proposes the idea that realistic looking android robots that can be appealing if they have good aesthetic designs [2]. Evidence suggests that if a human-like robot has the intelligence framework to enable it to interact with its surrounding as a human would, the probability of acceptance in the society increases [2]. This idea of bridging the uncanny valley via improving android head robot's realistic interaction capabilities is the focus of study in this thesis.

There are two kinds of interaction that an android head can partake in, verbal and non-verbal communication [3]. Verbal communication represents speech and lip movement abilities. Non-verbal communication includes the facial expressions to convey emotions or feelings throughout the interaction with user, maintaining eye contact during conversations, and tracking users or targets. To ensure the robot's ability to have a realistic interaction with its environment, the control system for the above mentioned areas of communication have to be developed. It is also important to ensure that the android head mimics a human as closely as possible during conversation. This gives it the aesthetic edge needed to look more appealing and friendly. This has been the constant motivation behind the development of different modules during the course of the study which addresses these issues.

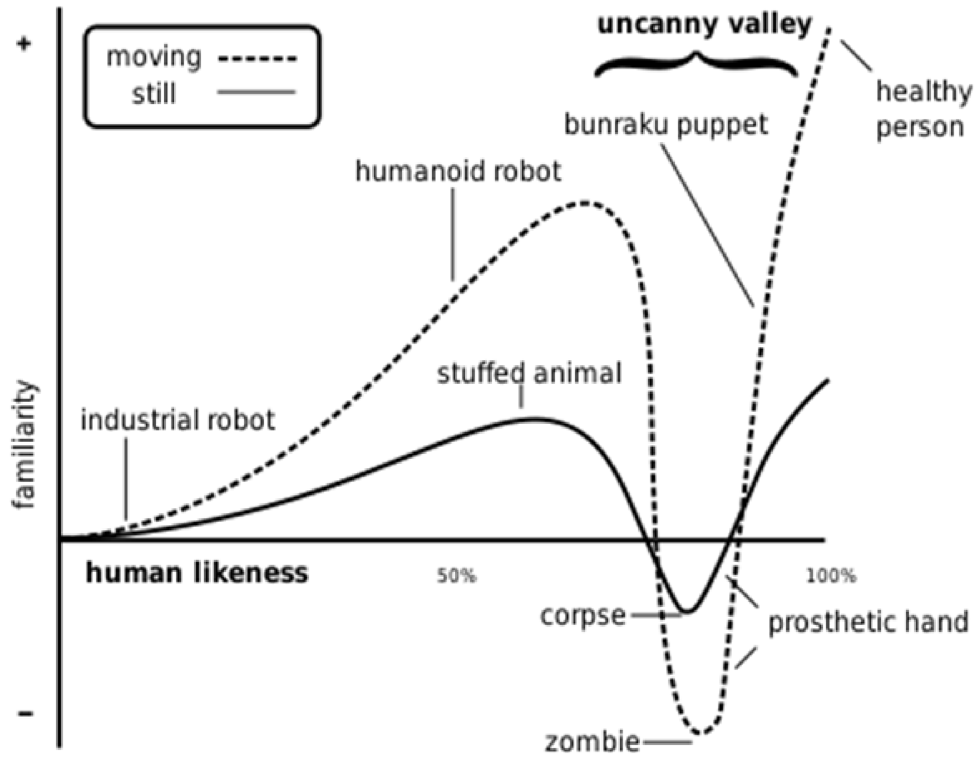


Figure 1-1 Graph representing the uncanny valley [4]

Robot operating system, or ROS as it is popularly known, is a Linux based collection of software libraries and packages which can be used to develop custom software packages for any intended robot. ROS is a popular platform for developing software for robots as it provided reusable modules to be used and built upon.

The absence of any simulator for testing the software modules developed in this work motivated the development of a robot operating system (ROS) package for android heads. Using a simulator makes it safer for the robot as it buffers it from erroneous code. In the case of an android head, since the skin is made up of spongy and light elastic materials, any unreasoned stretching due to faulty code can result in tearing of the skin.

This strives to bridge the uncanny valley to make social robots with human faces to be accepted by human users without any reservations. This will increase the use of social robots everywhere for different purposes ranging from healthcare to telepresence services.

1.2 Challenges in Human Robot Interaction

Social robotics involves continued interaction with human beings. In order to make the robots appear friendly, its movements has to be controlled. The challenges of controlling the movements of a robot are both in the software and hardware employed.

For instance, the hardware of an android head contains several actuators which work in tandem to create expressive face for the android head. During the execution of movements, the mechanical constraints of the head should accounted for. Damage to the artificial skin of the android head, if any actuator malfunctions, has to be avoided. The placement of the actuators also plays an important role in generating realistic facial expressions. Other important features that makes an android head less human-like is mechanical noises as well as jittering motions of actuators, which give away the fact that it’s not similar to a human being.

During verbal communication with users, lip syncing by an android head plays an important role in communicating the intended idea effectively. Lip syncing is concerned with right timing of the actuators to make the lip movements along with the audio file. The timing has to be meticulously planned and programmed in order to keep up with the audio. Any ambiguity can cause the android head to look unnatural.

Every word is associated with its respective set of visemes. Visemes are the lip shapes that are formed during the process of enunciating words. A word is made up of different sounds and, in order to articulate them, a combination of visemes are used. While any word is planned to be spoken, the pronunciation and the corresponding visemes has to be mapped. Studies point out that if the lip shape of a robot doesn't match the words it is speaking, it can create confusion for the user when attempting to perceive the spoken word [5]. The timing of the visemes and their transition have to be accounted for to make the speech process appear human-like.

The ultimate goal being to make human like expressions and be aware of the surrounding environment, the challenges include identifying the mechanical constraints, quantifying the expressions, driving the actuators with correct velocity and acceleration to the correct position and making the software package generic which can allow any other android head to be plugged in.

1.3 Description of Work Conducted

In the work presented, an android head robot named Philip K. Dick, or PKD, was programmed to learn and then mimic the facial expressions of a human user. The facial expression of the user was captured by Microsoft Kinect® and the corresponding messages were sent to the actuators to create the same facial expression. The advantage of using the Kinect is to perform marker-less tracking of the facial expressions.

Different types of controllers were used for object and human tracking by PKD such as potential controller, back-stepping controller, and open loop controller. The coordinates of the target are transformed to PKD's coordinate system using appropriate transformation matrices. Then, the yaw, pitch and roll angles of PKD are calculated in order to point its gaze direction towards the target. Different controllers help in ensuring continuous tracking.

Two cameras are used to track objects or users. A stationary Kinect® is used for depth sensing of the target. A web camera, embedded in the eye of PKD, which changes orientation and position with head or eye movement always aligns with the gaze direction of PKD. If the web camera alone is used for tracking purposes as in previous work in our lab [6], its range of detection is limited. But if it is augmented with the Kinect®, the range of detection increases and the distance of the target with respect to PKD can also be calculated.

Lip movements and lip-syncing capabilities of PKD was explored to enable speech capability. Literature study was done to understand speech process in humans. Visemes were studied and actuators controlling the lip region of PKD were programmed to recreate lip contours. These visemes can be used to mimic human lip movements and implement speech process in PKD.

A ROS package is also developed in order to address different aspects of an android head. The ROS package will contain packages which can enable any android head robot to have environment awareness and can interact with it at ease. A Unified Robot Description Format (URDF) model of PKD was created using its CAD model. Axes were defined for the joints and restriction on joint movements were defined in the CAD model using which the URDF model was developed. The advantage of using an URDF model of PKD is to have a standardized format of its kinematics and dynamics properties. These properties help in calculation of the transformation matrices of PKD while it is in motion and changes orientation with respect to its base. Also the controls package of the android head has the ability to run different controller algorithms which can be implemented according to the need. A configuration file will record the configuration of the android head that has to be operated and will have the ability to adapt to any robot. The open source nature of the software will make it possible for others to contribute as well thus furthering the research.

1.4 Research Contributions of Thesis

PKD is a very expressive robot, capable with a wide range of facial expressions due to its 24 face actuators. Four more actuators are used for neck movement of PKD.

The research contributions are aimed at making PKD more human-like and improve the controls of the robot. For interacting with humans, PKD has to be able to carry out a conversation and should be able to express through verbal as well as non-verbal communication. Non-verbal communication includes maintaining eye-contact and responding with appropriate facial expressions which is the focus of our work.

1.4.1 Facial Expressions of PKD

We propose using neural networks to learn the facial expressions for PKD via an automated image acquisition and processing loop. A neural network is trained to generate actuator values for any required expression which eliminates the need of heuristic calculation of the required servo values to generate appropriate expressions.

We contributed an automated algorithm for calibration of PKD's expressions without manual tuning of the actuator values. This is extremely important in reducing calibration time and effort for android heads that operate over long periods of time. It also addresses non-linearities in the mapping between the expressions and the corresponding actuators that are needed to form them.

For the purpose of lip syncing, visemes have to be articulated by PKD's lip in sync with the work being spoken. Research study was conducted to identify the process of speech, lip movements, control methods and the "McGurk Effect" [5] encountered due to loss of synchronization between lip movements and audio played.

Different lip shapes, which are part of visemes mapping list, were executed by heuristic method. The actuator data required to form the visemes were stored on the controller of PKD which can be used during the speech process.

1.4.2 Human Robot Interaction by Target Tracking

Target tracking involves recognition and localization of the target. Using the data about the target, the robot is actuated to point its gaze towards it. For a social robot, such as PKD, the motion executed has to be human-like to enable realistic human-robot interaction.

The use of a stationary secondary RGB-D based camera, i.e. Microsoft Kinect®, is proposed to increase the range of detection and boost the target localization process. The camera inside PKD's eye and Microsoft Kinect® are used together for tracking and feedback purposes using a position and image based visual servoing scheme.

Several control algorithms, such as tracking using external RGBD based camera, potential controller and back-stepping controller, were formulated and implemented. Experimental results are presented in the thesis.

1.4.3 ROS Open Source Package

A ROS simulation package was developed for PKD. Along with the simulation package, a joystick controls package was also developed for controlling PKD's head and eye motions. This package can be found at: https://bitbucket.org/nextgensystems/animus

1.4.4 Papers Published

A paper documenting the use of neural networks to train and generate human facial expressions was published at the Automation Science and Engineering (CASE), 2014 IEEE International Conference conference – “Learning human-like facial expressions for Android Phillip K. Dick” [7].

Another paper titled – “Gaze Control of Humanoid Robot with Potential Functions and Donders’ Constraint” was submitted at the ICDL-EPIROB 2015 conference and is currently under review.

1.5 Thesis Organization

Chapter 2 contains a literature review on the following topics: human-robot interaction, speech synthesis and lip syncing by humanoid head, linear and non-linear control systems for an android head, facial expressions generation using an android head, neck-eye coordination of an android head while tracking targets and social robotics.

Chapter 3 describes in details the hardware and software that was used for conducting experiments for the thesis. The system specification of the android head, PKD, is documented along with the controllers used to control the actuators in PKD. The use of the Faceshift software is also described.

Chapter 4 explains the importance of the facial expressions for an android head in order to interact with the users and its surrounding. It also explains why facial expressions are needed during interaction. It goes on to describe the two methods that were used for the purpose of the facial expressions mimicking by PKD, namely the linear mapping method and the neural network method.

Chapter 5 contains the work related to the neck-eye coordination problem of an android while tracking any target. It explains the need for an external RGB-D based camera during the tracking process. It contains the target acquisition and localization process implemented during the process. There are three types of control systems which are described in the chapter that were implemented. It describes comprehensively the open loop control system, potential control systems and the back-stepping control system which is a non-linear control system that were developed for the thesis study.

Chapter 6 explains the need for developing the lip synchronization ability for PKD. It explains in details the various challenges in the process of lip synchronization for an android head. The adverse effect of incorrect lip synchronization by an android head is also discussed in this chapter. It contains the various issues that has to be addressed in order to develop a lip synchronization system for PKD.

Chapter 7 explores new prospects of developing ROS (Robot Operating System) package for PKD which will make the system open source and standardize the code base developed for PKD. It also explains the need for a simulator for PKD.

Chapter 8 finally discusses the contributions made in the thesis and the future work that can be done extending from it.

Chapter 2

Literature Review

Human-robot interaction has been a long discussed topic in the context of social robotics. The research in the area of social robotics strives to make the robots act and behave like humans. This would help societies accept robots that can improve their quality of life provided that humans are safe and satisfied during the interaction process.

2.1 Androids

There has been much research in the field of androids, where the significance of the appearance and the behavior of such robots were studied [8]. The development of the android robots aims at building a human-like robots which can behave and interact like a human. Figure 2-1 shows the different types of robots being developed.

The use of the androids as a telepresence device has been a major area of research study [9]. The research studies by Ishiguro Hiroshi’s team have strived towards teleoperating the androids and using them as telepresence devices [9] [10]. There have also been studies about the educational application of an android robot [11]. The effectiveness of SAYA robot as an educator at an elementary school was studied and documented [11].

The most important aspect of the humanoid research has been to bridge the “uncanny valley” [1]. There are many studies conducted to determine the acceptability of a robot when its appearance starts becoming indistinguishable to that of a human. Mori in 1970 proposed the theory of uncanny valley [1]. It has been seen time and again that the human like appearance of a robot contributes to the eeriness of a robot [1] [4] [12]. There has however been an unperturbed effort to bridge the valley by the use of mechanical engineering, AI and art [2].

Studies reveal that robots with human like appearance are not always rejected outright by the society if can attain human like social interaction [2]. It was concluded in the study that an android with an AI system in place and using mechanical engineering and art, the acceptability of the android increases [2]. The social interaction requires the robot to behave in every way like a human being and every subtle movement should be indistinguishable from that of a human.

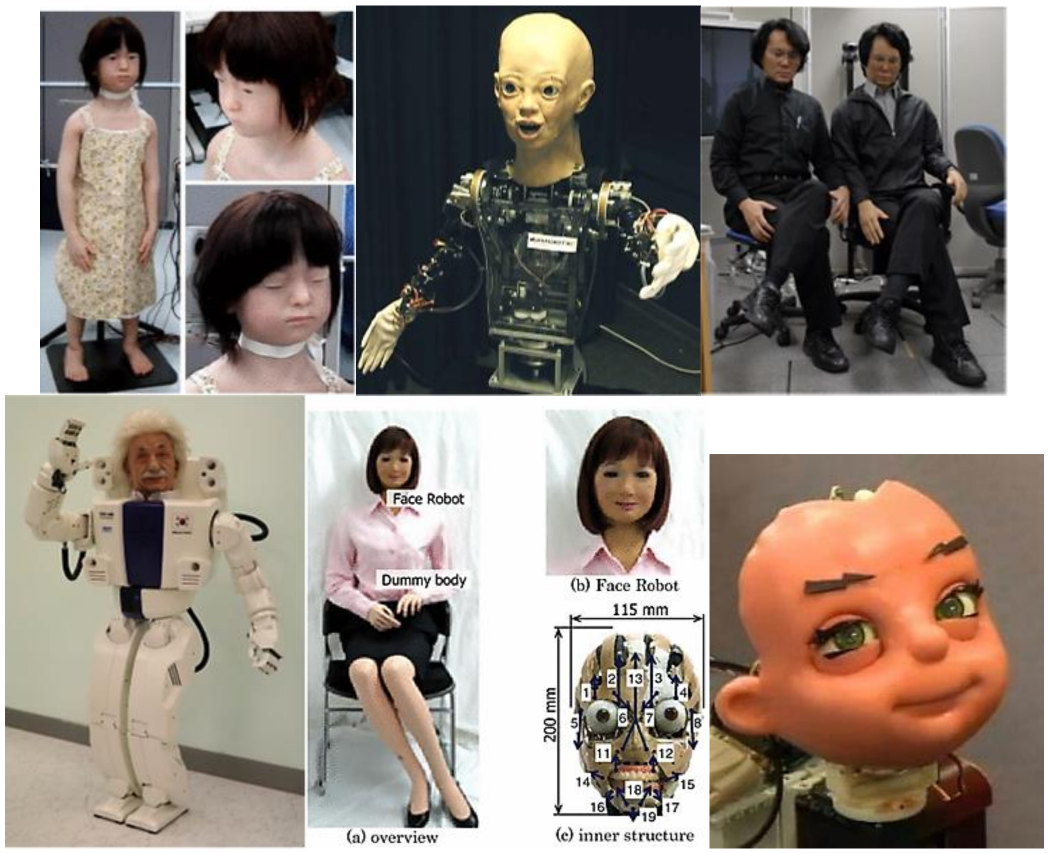


Figure 2-1 From L to R: Repliee R1 robot [8], BARTHOC Jr. [13], Geminoid HI-I [9], Albert HUBO [14], SAYA [11], ZENO [15]

2.2 Human-Robot Interaction with Facial Expressions

An android head, in order to behave like a human, has to express emotions, which is also a part of non-verbal communication. Facial expressions have been used to develop empathetic robots which are used for interaction purposes [13]. As shown in Figure 2-2 an android robot named Geminoid F is displaying various human like expressions.

There have also been studies to identify the confusion matrix in terms of identifying the expressions displayed by an android [16]. It is really important to convey the emotions correctly and not to be confused with a wrong emotion. For e.g. it was observed that fearful expression by Geminoid F android robot was also confused with the surprised emotion [16]. Similar confusion may lead to difficulties in having a realistic interaction between androids and human.

The facial expressions in android heads are generated by breaking down a human expression into basic units called animation units. Any human emotion is then described as a combination of the weighted animation units [17] [18] [19]. In terms of mimicking the human facial expressions, there has been usage of linear mapping [20] [21] between the animation unit and corresponding actuator controlling the part of the face. However it was observed that the mapping between the actuators and the animation units cannot be linear as there are many points in the android's face which are not controlled by a single actuator and cannot be moved linearly to form appropriate expressions. These studies have proved that mimicking the human facial expressions is in itself a challenge that has to be overcome in order to make the android heads human-like.

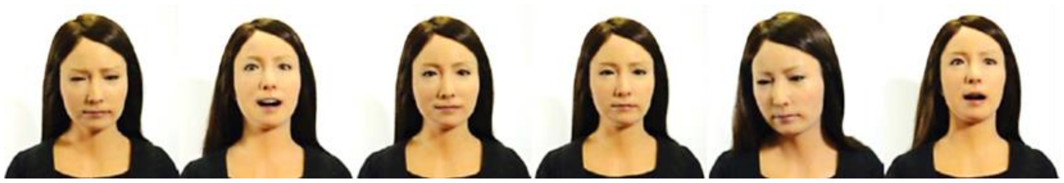


Figure 2-2 Geminoid F displaying facial expressions [16]

2.3 Human-Robot Interaction with Speech

During interaction with any user, the android robot has to talk and during the process lip synchronization which is important to convey verbal information. The mismatch between the spoken word and the lip formation gives a perception of a new syllable or word [22] [5]. This can cause confusion and be distracting during conversational interaction.

As shown in Figure 2-3, the android is programmed to perform lip movements like a human. During the speech process, it has to be always ensured that the lip shape formed and the intended viseme are the same. Else we will be facing issues with communication between and android and a human due to the “McGurk effect” [5].

There have been many studies on phonemes and visemes which indicate the smallest unit of a word and the lip formation during the speaking process of the word respectively [23]. Along with the lip formation the motion of the lip is also considered for the purpose of human like lip synchronization [22]. In order to achieve the synchronization all the delays induced during the process of execution are taken into account in order to make the process real-time. The studies indicate meticulous planning and executing the process to achieve perfect lip synchronization.

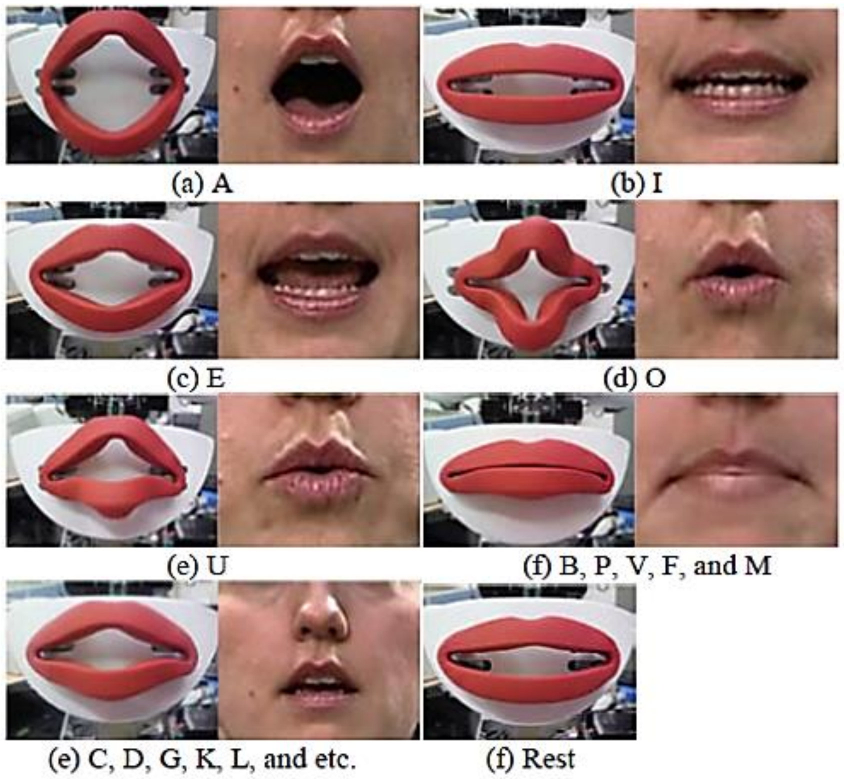


Figure 2-3 Lip Shapes formed by FR-i robot as compared to a human [22]

2.4 Human-Robot Interaction with Head-Eye Coordination

Another aspect of interacting with social robots is maintaining eye contact during conversation. In this context, the neck-eye coordination problem for android heads has been an interesting area of study. There are also cases where the robot might be instructed to track other targets or person. While doing so, it is desirable that the robot obeys kinesiology laws governing human head dynamics [24].

The initial studies in the field of target tracking include visual based target detection and actuation of robot [6]. Using a PID controller, the pan and tilt motion was executed to track any target. But apart from target tracking, human neck-eye coordination also has to be imitated by an android in order to appear similar to a human.

There has also been studies regarding mimicking the human neck eye coordination using robust learning algorithms [25]. It includes learning the human motion during the process of target tracking and implementing the same in the android robot for mimicking purposes.

The law governing the human head and eye movements has been documented as Donders' law and Listing's law [26]. These laws have to be obeyed the android head. The study regarding this problem also depends on the robustness of the input devices and the loss of target from the robot's visible frame during the tracking process. The execution of the saccadic eye movements similar to a human being has to be replicated in the android head to achieve human-like motions.

Although these laws govern the human head and eye movements, the dynamics of a human head is very different from that of an android. In order to drive an android to point its gaze direction towards a target and also ensuring that the motion is human-like at all times, the dynamics of the android has to be taken into account.

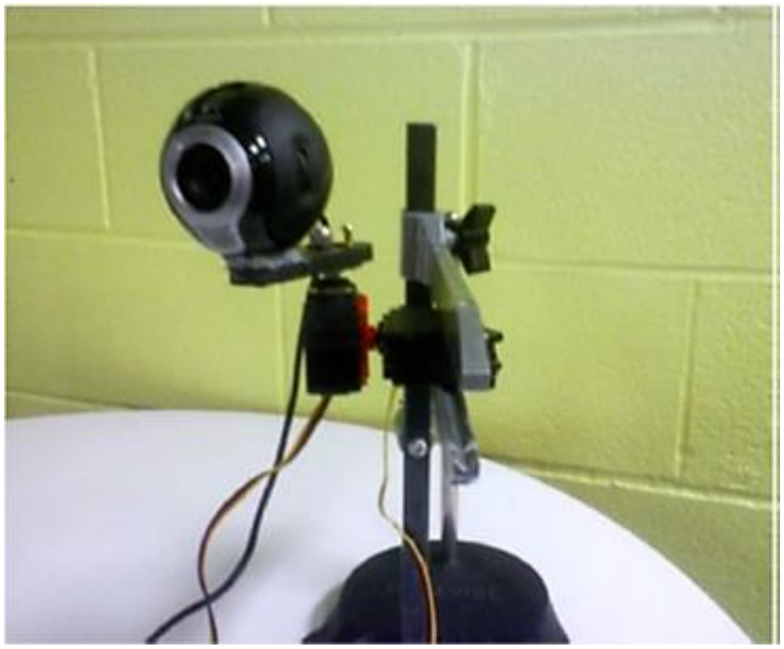


Figure 2-4 Pan and tilt servoing using visual based target tracking [6]

Using the studies and research in the field of humanoid head, the thesis work presented strives to develop and build on it a new framework for human-robot interaction that caters to the need of the android head in various aspects of its interaction with outside world.

Chapter 3

Description of Hardware and Software

In this chapter, we describe the hardware as well as the software system that has been employed in our research. This chapter contains details about the android head hardware, the servo controllers, the Microsoft Kinect® and the software development aspects of the system that were used for our research.

3.1 PKD Hardware Description

An android head modeled after the famous science fiction writer Philip K. Dick (PKD) was used to implement the learning and control algorithms and to conduct experiments. PKD's skin is made of Frubber®, a patented compliant silicone elastomer, which can be molded and deformed to form realistic facial features and contours. PKD is a third generation android head created by David Hanson and his company, Hanson Robotics. It is an expressive robot which has the ability to closely mimic human facial features

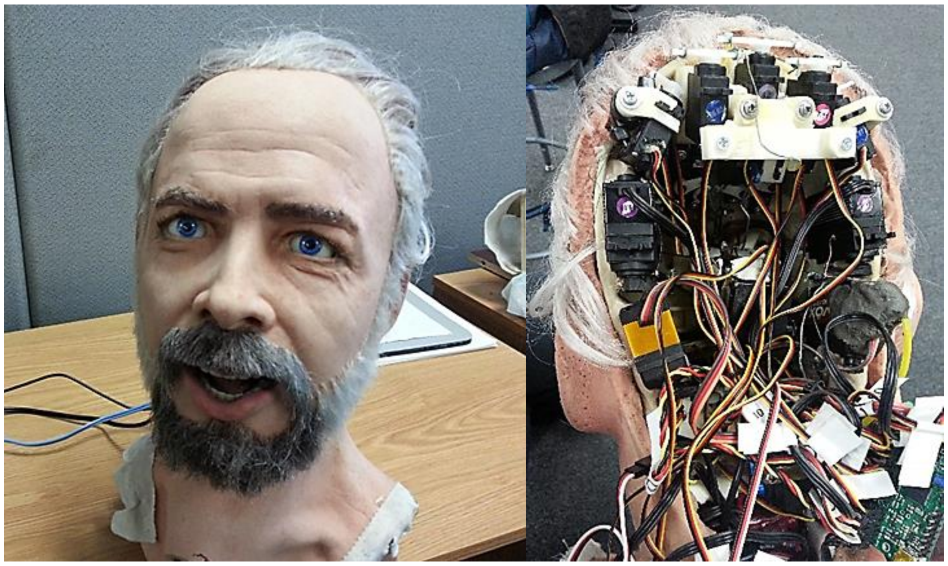


Figure 3-1 PKD in NGS lab

Dr. David Hanson patented Frubber®, or Flesh Rubber, and is being extensively used in other robots such as Hubo, Face, and so on. Frubber®'s material properties make it an ideal candidate to be used as the facial skin. It requires as low as 1/20th of the mechanical power of other skin materials, in order to pull or push the skin to create facial expressions [27]. Frubber® can be sculpted to have detailed human facial contours which makes Hanson robot look very realistic. The actuators are connected to the skin and depending on the position of the servos, the facial expressions are created. Frubber® can be sculpted to have detailed human facial contours which makes Hanson robots look very realistic.

Frubber® covers a plastic skull shell, representing a human head, which contains the actuators. Servo actuators in the robot are connected to the skin and create facial expressions by pinching selective locations on the face. There is 1 actuator connected to the jaw, 5 actuators are connected to the brow region to simulate Procerus, Orbicularis Oculi and Frontalis muscle movements, 2 actuators are connected to the cheek region on left and right side of the face, 2 actuators are responsible for duchenne smile, 2 actuators are responsible for smile action, 6 actuators are connected to the lip region, 1 actuator in the nose which represents nasalis muscle, 2 actuators control the upper lids and lower lids of the eye, 2 actuators control the pitch of the individual eye and 1 actuator is responsible for the roll of the eyes. With 25 DOF, PKD can be used to generate a wide range of facial expressions. There are 4 actuators positioned at the neck, where 2 actuators control the yaw action of the neck and the other 2 control the roll and pitch action.

The actuators controlling neck and eye of PKD are responsible for neck-eye coordination. During the tracking of a target, these actuators are used to direct the gaze direction of PKD. The uneven weight distribution of PKD head induces controller instability at the extremities which causes overshoot of the actuators. The controls and error correction have to be smoothened to give it a characteristic of a human head movement.

The camera in the left eye of PKD can be used for tracking purposes. The camera also assists in feedback for the control system during closed loop target tracking purposes.

3.2 Pololu Servo Controllers

Pololu servo controllers are used for controlling the servos inside PKD's head. The Pololu servo controller has 24 channels to control the actuators simultaneously. It is also capable of establishing TTL serial communication along with UART communication.

There are two controllers to control facial expressions and neck-eye motions separately. They are connected via USB based UART connection to the computer. It has speed, position and acceleration control for each channel. The maestro control program allows to setup individual channels with min and max range (to safeguard the connected servos and robot) and the 8-bit (0 - 254) range to ensure control though external program from the computer and reduced change in code due to re-calibration. The pin diagram of Pololu maestro 24-channel controller is shown below:

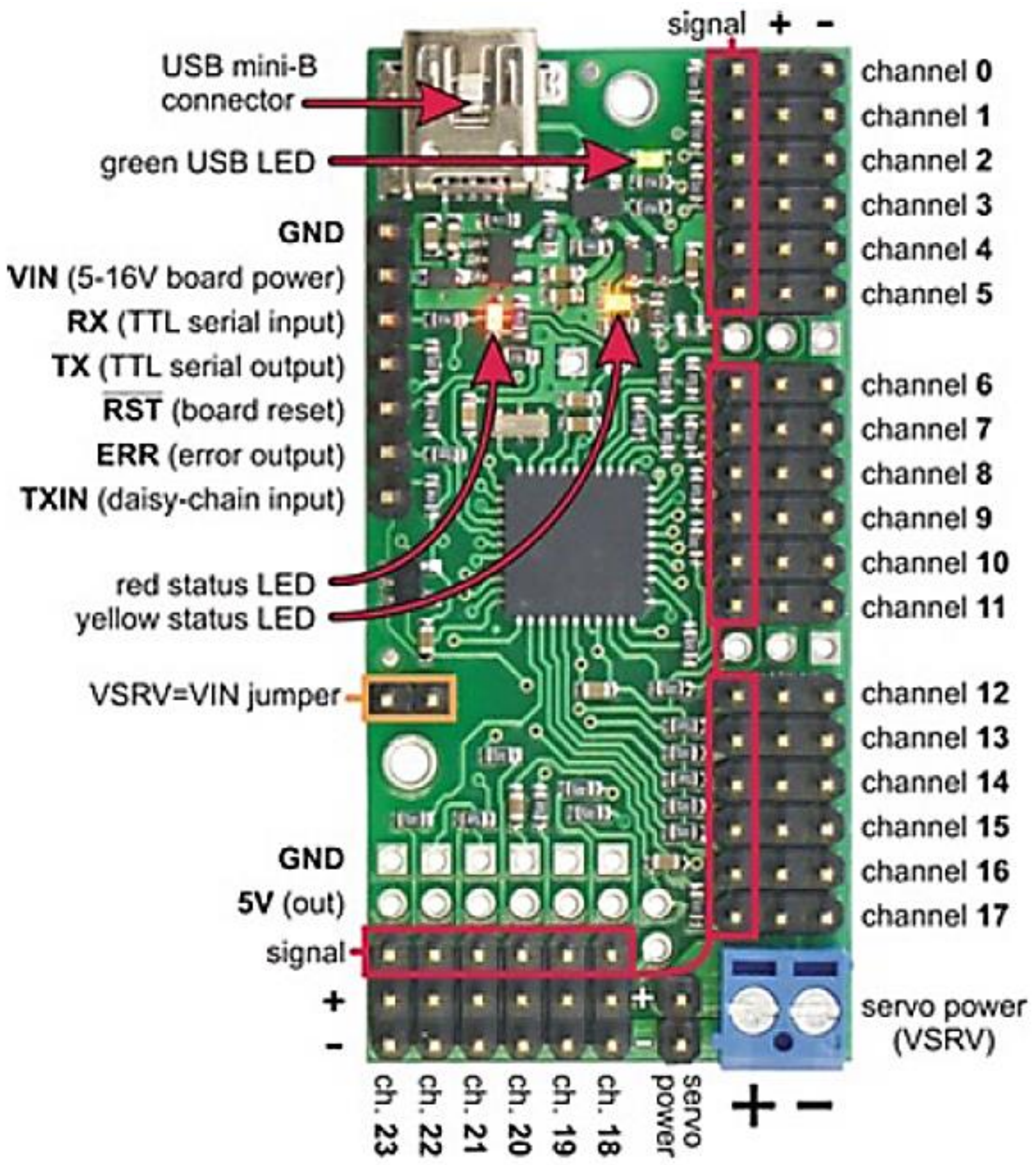


Figure 3-2 Pololu 24 channel Maestro Controller [28]

3.2.1 Different Modes of Operation

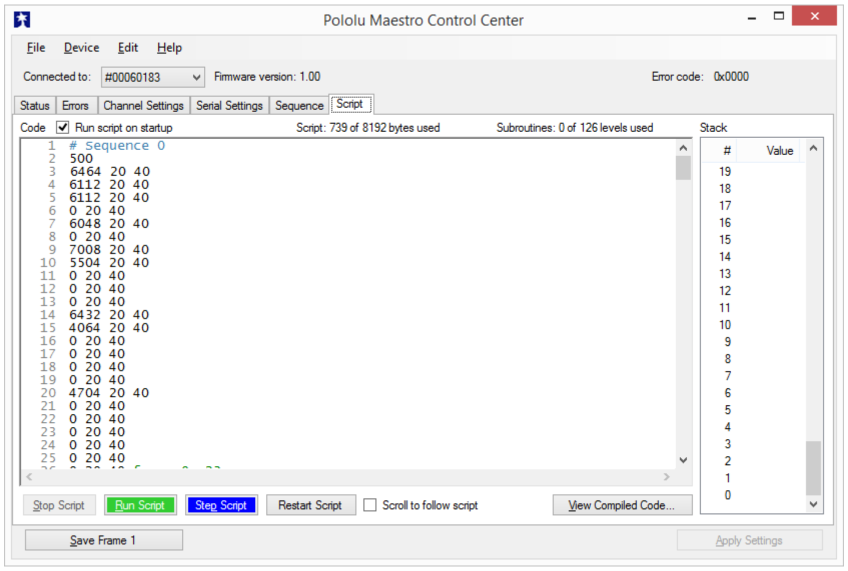
There are three modes of operation, namely USB, TTL serial and internal scripts [28]. At present the USB and internal scripting is being used to control PKD.

In the USB method, port programming is required to control the channels. We create a UART serial communication port with a baud rate of 115200. Then the message packets are formed as per requirement and sent to the controller.

The internal scripting makes it easier to load and save predefined settings and startup scripts. These internal functions can be invoked through the serial communication packets from the computer. This eliminates the need of constant reconfiguration and recalibrations. The programming details are described in details in the next section.

3.2.2 Programming the Controller

The Pololu maestro controller has its own scripting language which allows configuring the servo values, speed and acceleration [28]. It is a stack based scripting language with First-In-Last-Out approach where the commands and data are pushed into the stack and are executed. The values in the stack are integers which can range from -32768 to +32768 and there can be a maximum of 126 commands or data combined in the controller stack [28] as shown in Figure 3-3.



Connected to: #00060183 Firmware version: 1.00 Error code: 0x0000

☑ Run script on startup Script: 739 of 8192 bytes used Subroutines: 0 of 126 levels used

☐ Scroll to follow script

```
# Sequence 0
500
6464 20 40
6112 20 40
6112 20 40
0 20 40
6048 20 40
0 20 40
7008 20 40
5504 20 40
0 20 40
0 20 40
0 20 40
6432 20 40
4064 20 40
0 20 40
0 20 40
0 20 40
0 20 40
4704 20 40
0 20 40
0 20 40
0 20 40
0 20 40
0 20 40
```

Figure 3-3 Pololu script window

The scripting language allows the usage of predefined control commands, loop commands, timing commands, stack-based commands, mathematical commands and servo or other output based commands. This allows the user to have a real-time feedback control on board the controller. Once these programs are loaded on board, it can be triggered from a UART port programming based C or C++ program by mentioning the subroutine number of the function. It is also possible to load arguments for the function to be triggered.

In conclusion, the controller provides the necessary tools to have full control over the actuators and reduce damages and time over recalibrations.

3.3 Microsoft Kinect®

The Microsoft Kinect®, as shown in Figure 3-4, is a RGB-D based camera which provides color-stream as well as depth-stream. It has an IR emitter and sensor, color sensor, tilt motors and an array of microphones. The field-of-view (FOV) for the camera is 43 degrees vertically and 57 degrees horizontally. The depth data is streamed at a rate of 30FPS. It also has an inbuilt accelerometer which is accurate to 1 degree [29].

Figure 3-4 Microsoft Kinect® [30]

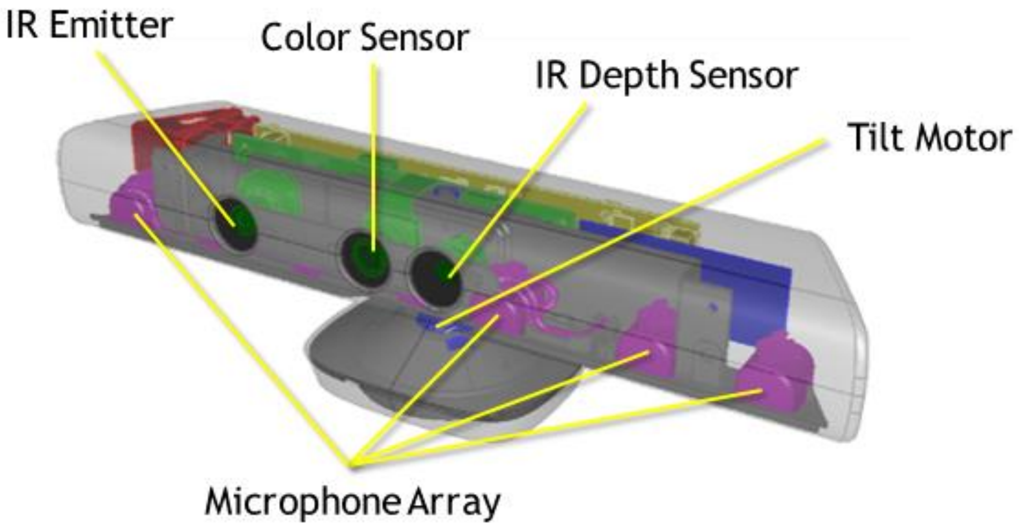


Figure 3-5 Microsoft Kinect components [29]

Microsoft provides a SDK which assists in facial recognition, seated-mode or far-mode based depth perception and skeletal tracking. We use kinect to get detect object or face and get their depth in order to calculate their position with respect to the Kinect's base.

3.4 Faceshift®

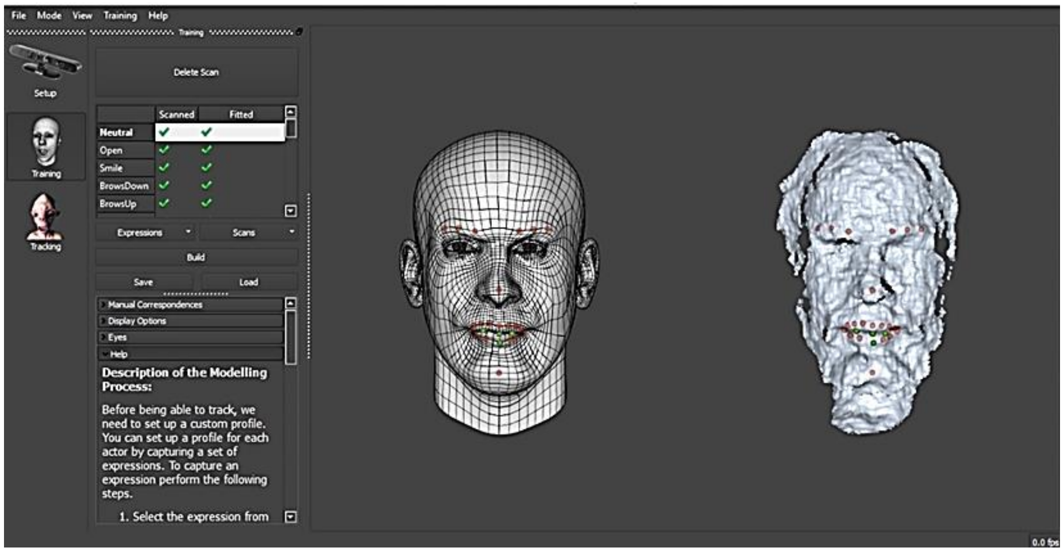
Faceshift® is a facial expression capturing software which is marker less and takes Kinect's data stream as input. Faceshift recognizes a face and streams datagram which consists of breakdown of expression values ranging from 0 to 1. The facial expressions or the blendshapes that are detected are [31]:

Table 3-1 Faceshift Blend shapes [31]

1	EyeBlink_L	25	MouthLeft
2	EyeBlink_R	26	MouthRight
3	EyeSquint_L	27	MouthFrown_L
4	EyeSquint_R	28	MouthFrown_R
5	EyeDown_L	29	MouthSmile_L
6	EyeDown_R	30	MouthSmile_R
7	EyeIn_L	31	MouthDimple_L
8	EyeIn_R	32	MouthDimple_R
9	EyeOpen_L	33	LipsStretch_L

Table 3-1—*Continued*

10	EyeOpen_R	34	LipsStretch_R
11	EyeOut_L	35	LipsUpperClose
12	EyeOut_R	36	LipsLowerClose
13	EyeUp_L	37	LipsUpperUp
14	EyeUp_R	38	LipsLowerDown
15	BrowsD_L	39	LipsUpperOpen
16	BrowsD_R	40	LipsLowerOpen
17	BrowsU_C	41	LipsFunnel
18	BrowsU_L	42	LipsPucker
19	BrowsU_R	43	ChinLowerRaise
20	JawFwd	44	ChinUpperRaise
21	JawLeft	45	Sneer
22	JawOpen	46	Puff
23	JawChew	47	CheekSquint_L
24	JawRight	48	CheekSquint_R



	Scanned	Fitted
Neutral	✓	✓
Open	✓	✓
Smile	✓	✓
BrowsDown	✓	✓
BrowsUp	✓	✓

Description of the Modelling Process:

Before being able to track, we need to set up a custom profile. You can set up a profile for each actor by capturing a set of expressions. To capture an expression perform the following steps.

1. Select the expression from

Figure 3-6 Faceshift training

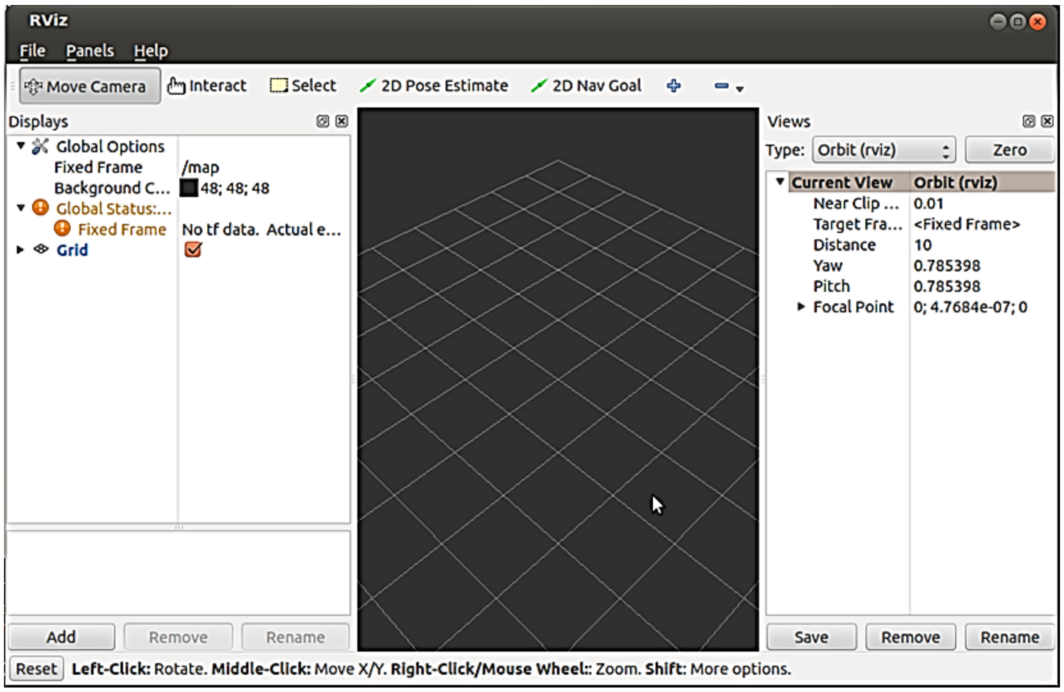
These facial expressions from the user can be recorded and be used for driving the facial actuators of PKD. For the purpose of detection, each user has to have an individual profile trained in the Faceshift. As the values of expressions are relative to the trained model of any user, it can be used as a standard measure for the purpose of data analysis and driving PKD without going over the limits of the actuators.

3.5 ROS

Robot Operating System (ROS) is a Linux based software platform, which offers a standardized platform for driving and simulating robots. It can be programmed in C++ or Python. It also has a real-world simulator called Gazebo. Robot Operating System is used in the project to prepare a reusable package to cater to the needs of PKD.

ROS provides a lot of reusable packages to be used and built upon which is really efficient and helps avoid reinventing the wheel. For e.g. ROS provides access to already build packages such as SLAM (simultaneous localization and mapping), object detection and many other reusable packages related to the field of robotics in including the control systems. The objective of using ROS is to provide a simulation package for PKD so as to avoid running the code on the robot and minimizing the damage that can be caused due to faulty coding.

A simulator was required to test the codes safely. The 3D CAD model of PKD was taken and modified to be able to convert it into a URDF (Unified Robot Description Format) file. All the joints and links were properly declared and then PKD model was converted successfully. The URDF file is then parsed through the URDF parser in ROS and a simulation model was successfully created on which codes can be tested before implementation. In the URDF, the Frubber® was not simulated, therefore the ROS package deals with the movements of the neck and eye only. A controls package was also created which has the mapping between the rotation angles of the neck and their corresponding actuator values to be sent.



Global Options	
Fixed Frame	/map
Background C...	■ 48; 48; 48
Global Status:...	
Fixed Frame	No tf data. Actual e...
Grid	✓

Type: Orbit (rviz)

Current View	Orbit (rviz)
Near Clip ...	0.01
Target Fra...	<Fixed Frame>
Distance	10
Yaw	0.785398
Pitch	0.785398
Focal Point	0; 4.7684e-07; 0

Reset **Left-Click:** Rotate. **Middle-Click:** Move X/Y. **Right-Click/Mouse Wheel:**: Zoom. **Shift:** More options.

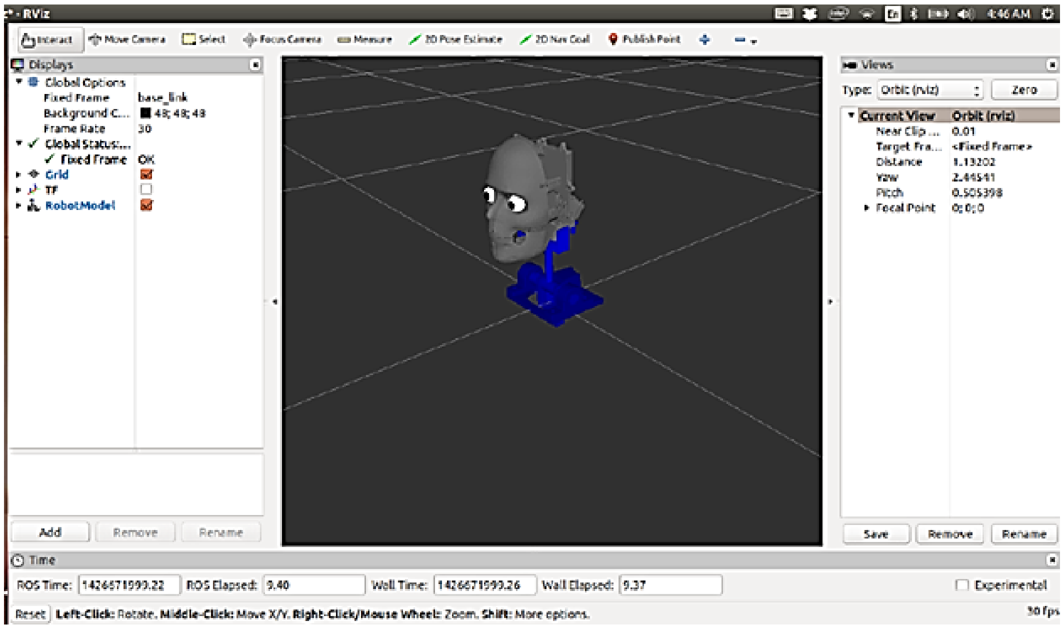
Figure 3-7 ROS Rviz [32]

Another purpose of using ROS is to develop a package specific to an android head which is yet not available as a single package bundle as is available for robots like PR2. The package will be able to cater to the needs of any android head and will include packages such as neck-eye coordination, various controls packages, object detection and gaze orientation being some of the examples.

ROS also helps in standardizing the robot via the use of URDF (Unified Robot Description Format). An URDF is a XML based file which has the information of all the joints and links of the robot. An URDF file of a robot helps in calculating the transformation matrix in real-time which reduces the calculation for the orientation of the links and the joints.

3.5.1 Simulation

The simulation of PKD is done in RViz, which is provided by the ROS platform. The URDF file is parsed and the model is created. The simulated model can now be worked on and the specified joints and links can be actuated depending on the limits provided in the URDF file.



Global Options
Fixed Frame base_link
Background C... 48; 48; 48
Frame Rate 30
✓ Global Status:...
✓ Fixed Frame OK
Grid ✓
TF ☐
RobotModel ✓

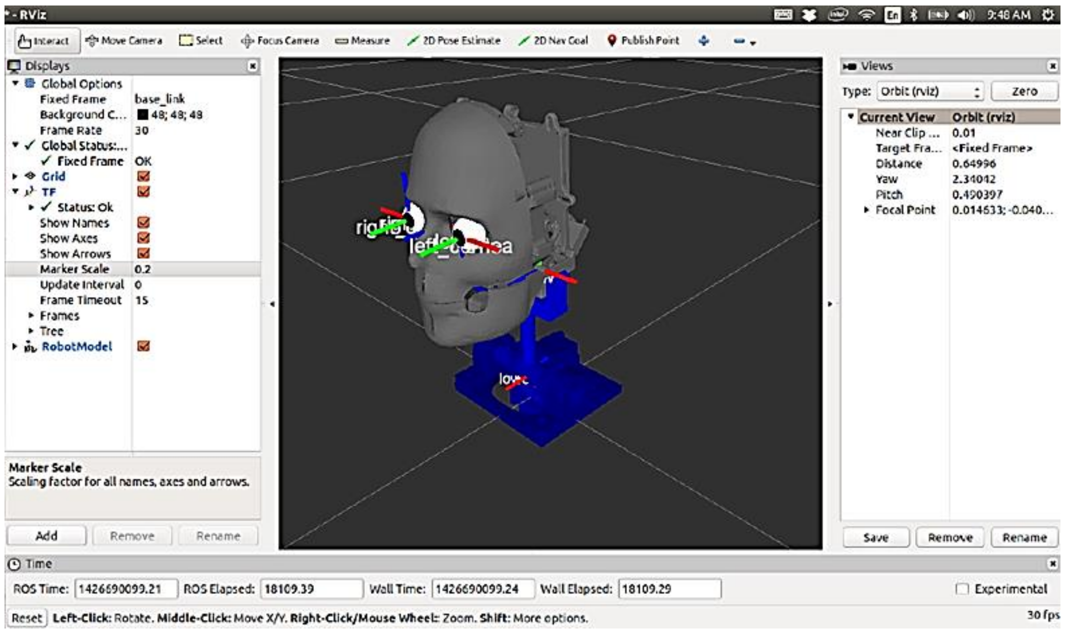
Type: Orbit (rviz)

Current View	Orbit (rviz)
Near Clip ...	0.01
Target Fra...	<Fixed Frame>
Distance	1.13202
Yaw	2.44541
Pitch	0.505398
Focal Point	0; 0; 0

ROS Time: 1426671993.22 ROS Elapsed: 9.40 Wall Time: 1426671993.26 Wall Elapsed: 9.37 ☐ Experimental

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click/Mouse Wheel:** Zoom. **Shift:** More options. 30 fps

Figure 3-8 PKD simulation in Rviz



Global Options
Fixed Frame: base_link
Background C...: 48; 48; 48
Frame Rate: 30
Global Status:...
✓ Fixed Frame: OK
Grid ✓
TF ✓
✓ Status: Ok
Show Names ✓
Show Axes ✓
Show Arrows ✓
Marker Scale: 0.2
Update Interval: 0
Frame Timeout: 15
Frames
Tree
RobotModel ✓

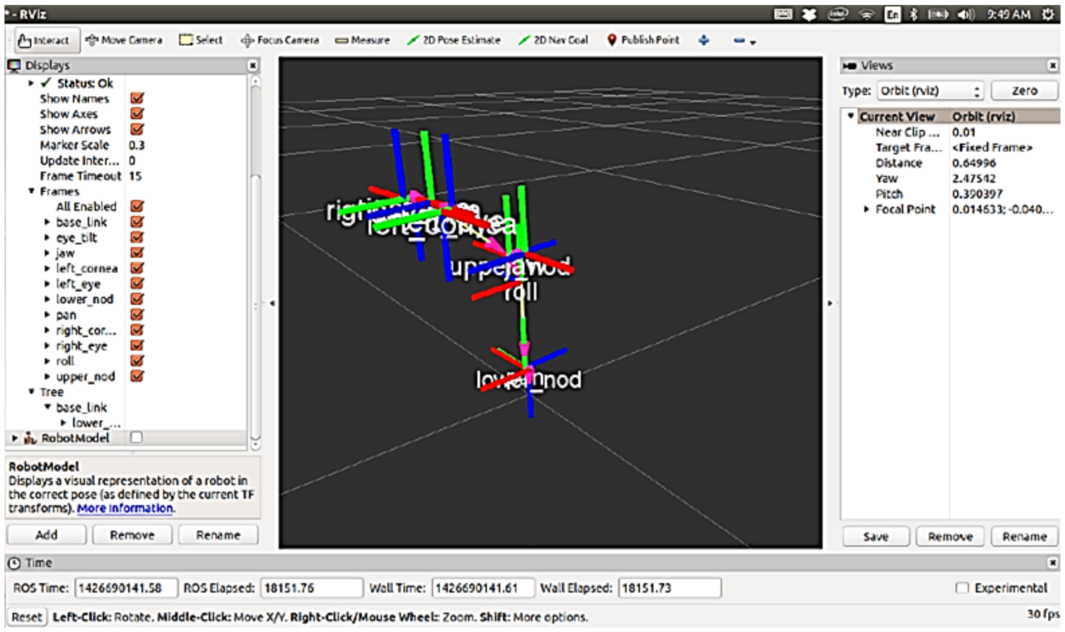
Marker Scale
Scaling factor for all names, axes and arrows.

Type: Orbit (rviz)
Current View: Orbit (rviz)
Near Clip ...: 0.01
Target Fra...: <Fixed Frame>
Distance: 0.64996
Yaw: 2.34042
Pitch: 0.490397
Focal Point: 0.014633; -0.040...

ROS Time: 1426690099.21 ROS Elapsed: 18109.39 Wall Time: 1426690099.24 Wall Elapsed: 18109.29

Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel: Zoom. Shift: More options. 30 fps

Figure 3-9 PKD's joints, links and their axis



✓ Status: Ok
Show Names ✓
Show Axes ✓
Show Arrows ✓
Marker Scale: 0.3
Update Inter...: 0
Frame Timeout: 15
Frames
All Enabled ✓
base_link ✓
eye_tilt ✓
jaw ✓
left_cornea ✓
left_eye ✓
lower_nod ✓
pan ✓
right_cor... ✓
right_eye ✓
roll ✓
upper_nod ✓
Tree
base_link
lower_...
RobotModel ☐

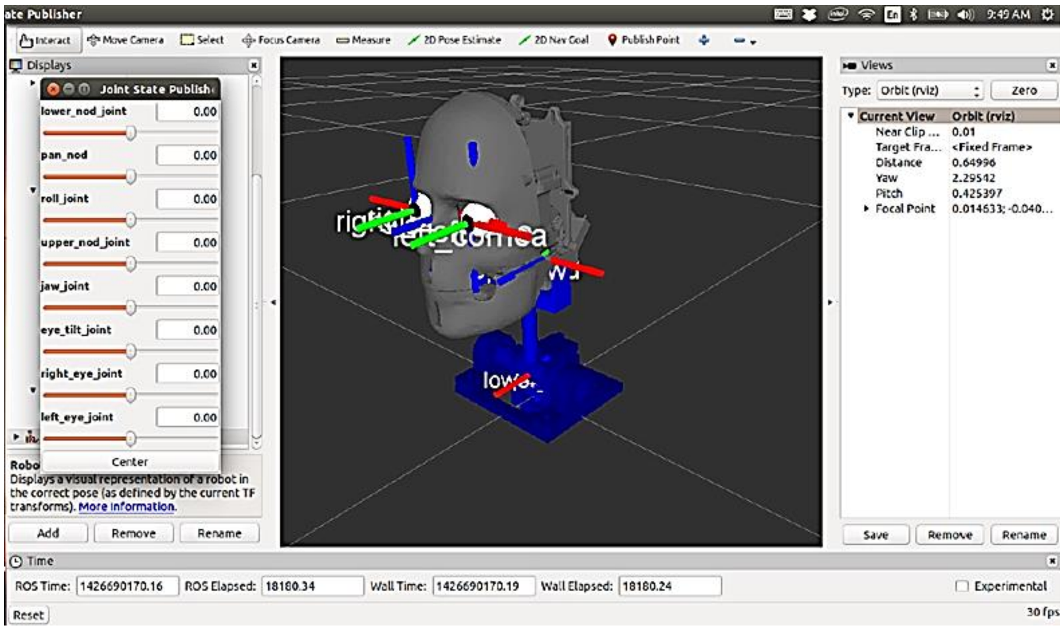
RobotModel
Displays a visual representation of a robot in the correct pose (as defined by the current TF transforms). More Information.

Type: Orbit (rviz)
Current View: Orbit (rviz)
Near Clip ...: 0.01
Target Fra...: <Fixed Frame>
Distance: 0.64996
Yaw: 2.47542
Pitch: 0.390397
Focal Point: 0.014633; -0.040...

ROS Time: 1426690141.58 ROS Elapsed: 18151.76 Wall Time: 1426690141.61 Wall Elapsed: 18151.73

Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel: Zoom. Shift: More options. 30 fps

Figure 3-10 PKD TF Tree visualization



lower_nod_joint	0.00
pan_nod	0.00
roll_joint	0.00
upper_nod_joint	0.00
jaw_joint	0.00
eye_tilt_joint	0.00
right_eye_joint	0.00
left_eye_joint	0.00

Robo

Displays a visual representation of a robot in the correct pose (as defined by the current TF transforms). More Information.

Type: Orbit (rviz)

Current View	Orbit (rviz)
Near Clip ...	0.01
Target Fra...	<Fixed Frame>
Distance	0.64996
Yaw	2.29542
Pitch	0.425397
Focal Point	0.014633; -0.040...

ROS Time: 1426690170.16 ROS Elapsed: 18180.34 Wall Time: 1426690170.19 Wall Elapsed: 18180.24

Figure 3-11 Joint state publisher to control simulations

These simulations help us in safeguarding the hardware and visualizing the robot in action. The TF tree formed by the simulation contains the real-time data of the robot configuration during its motion. Figure A - 1 shows the URDF visualization of TF tree for PKD.

Chapter 4

Facial Expressions Generation For PKD

4.1 Introduction

Facial expressions for an android head is a very important feature in order to bridge the uncanny valley. It also facilitates in human-robot interaction as users tend to be averse to any robot which has human like appearance with a robotic motion. In order to make PKD more social it is necessary to take care of the facial movements and the expressions generated. Facial expressions are also required during conversation and interaction with users to express the mood of the situation and create a perception of human as oppose to an expressionless robot.

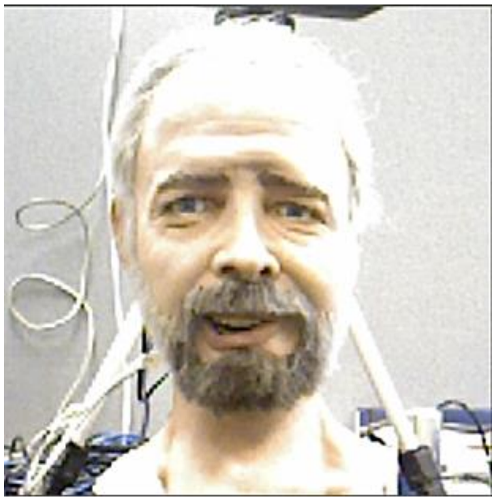


Figure 4-1 Sample facial expression by PKD

In the work presented, there were two methods used for getting the expression values of a user, namely by Faceshift and Kinect SDK. As described in section 3.4, there are 48 constituent expressions or blendshapes that can be mapped to PKD's actuators which are responsible for those expressions and regenerate the expression in PKD. The other method is by using Kinect SDK for facial Animation Units [33] for the same. The Kinect SDK provides 6 animation units (AUs), head pose angles and 11 shape units which can also be used to map the actuators. The values of the AUs range from -1 to +1 depicting the high and low of each constituent expression, whereas the head pose angles range from -90 degrees to +90 degrees. The mapping as well as the speed and acceleration of the actuators also constitute the visual perception of PKD. Correct mapping with incorrect transition can also lead to the rejection of PKD for being socially acceptable.

There are two ways of mapping the values from expression space to actuator space that has been implemented in this project, direct mapping and neural network mapping. These methods are discussed in details in the following section.

4.2 Mapping Animation Units to Actuators

4.2.1 Direct Mapping

In this method, heuristic approach is involved in regenerating the facial expressions in PKD.

The animation units or the expression values are recorded from the user and is passed on to the mapping algorithm. Before mapping, the range of all the actuators is noted down. After setting up the Pololu controller, all the actuators can be controlled by sending a value from 0 to 254 as target data. The value of facial expressions received from Faceshift ranges from 0 to 1 which is mapped to 0 to 255 linearly. The same process is followed in case of Kinect SDK where the value 0 to 255 is mapped to the animation unit range from -1 to 1 linearly.

This approach towards mapping is fast in calculation, but depends heavily on the accuracy of calibration of the actuators to find the neutral expression. The calibration involves perception of the programmer and can change from person to person.

Other aspect of this method is that it is assumed that the mapping is linear and thus PKD is considered to be a linear system which is not the case. The non-linearity also induces error in the system and as the system is open loop, it remains unchecked. Therefore it is not a reliable method of generation of expressions.

4.3 Using Neural Networks to Learn Facial Expressions

A neural network involves supervised learning of the mapping between the actuators and expressions. Using a neural network will eliminate the heuristic approach for calibration and learns the mapping at any given situation. This also helps in taking care of the non-linearity in PKD.

In this method, first random actuator values are given to PKD and the corresponding facial expressions are recorded from Faceshift. These sets of random actuator values and their resultant facial expressions become the training set for the neural network. The training algorithm used for the neural network is the Levenberg-Marquardt backpropagation algorithm. The training was done using the neural network toolbox of Matlab which constituted of a neural network which has three layers with 80 hidden units.

During the process of training, there is separate training for the mouth region and rest of the face region. The separate training is done as these are two independent regions of the face and to reduce the error induction between the two regions while training the neural network together. Also another step to reduce the mean square error (MSE) while training was to use normalized servo values, i.e. all the servo values were divided by the maximum value which is 254 in this case. This resulted in changing the range from 0 – 254 to 0 – 1. The normalized values were then used in the training set.

After the neural network is trained, facial features or expressions are recorded from the user using the Faceshift. These data are now the input to the trained neural network and the output is recorded. The output is now the resultant set with normalized servo values. These were multiplied by 254 to convert them to actual servo values and were used to regenerate the facial expressions of the users.

In this method the error is induced during the training of the neural network and bad data sets. Another error inducing agent is the neural network memorization. This causes the neural network to memorize the pattern which causes incoherent results while validation phase of the neural networks. Therefore a larger dataset with randomized values are to be used in order to ensure efficient mimicking of facial expressions by PKD.

4.3.1 Algorithm for Facial Feature Mapping

In this section, we describe the proposed data generation and learning algorithm used for facial feature mapping for PKD Android. The work presented in this section has been developed in collaboration with my colleague Ahsan Habib. The work was published at the CASE 2014 conference, titled “Learning human-like facial expressions for Android Phillip K. Dick” [7].

Neural Networks (NN) are used to learn the mapping between the facial expressions of PKD and corresponding servo configurations. The graphical description of the system used to train the neural network is shown in Figure 4-2. For the experiment we considered translating the lip motion from the user to the Android. For this we considered 9 actuators positioned around the lips and 9 relevant facial feature points extracted using face shift. Servo signals, generated by random value generation, are used to drive the servos of PKD and create facial expressions. Nine facial features, defining an expression, are obtained using Faceshift to create a training database for the neural network.

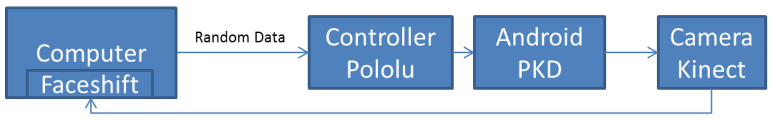


Figure 4-2 Neural Network Training Setup

In order to ensure the databases encompass a rich set of the facial features we cannot rely on the randomly generated servo signals. Therefore, we employed a genetic algorithm based control signal generator that guides PKD Android to produce an expression such that a target facial feature values maximizes (or minimizes). This ensures that the extremities of the facial feature space are explored and added to the database. A single actuator or multiple actuators may be responsible for the variability of a particular facial feature. GA through the process of evolution gradually identifies that responsible actuators and tweak them to produce the desired results. The process is described in the GA Face Search algorithm below [7]:

4.3.1.1 Genetic Algorithm for random data generation

1. [Start] Random sets of 8-bit (0 - 255) servo values were generated for the actuators (9 actuators in this case). The collection of sets of servo values can be visualized as a pool of chromosomes, where each chromosome consists of 72 genotypes (9 actuators each accepting 8 bits).
2. [Loop] The subsequent steps are followed to generate datasets for the training file.
    a. [Fitness] Each chromosome in the collection is taken and its constituent servo values are used to drive the facial expression of PKD. The facial expression is detected by Faceshift® and outputs constituent facial feature values. In order to evaluate the fitness of a chromosome, the value of a target facial feature is taken into account and a fitness score, q, is evaluated for the chromosome.
    b. [Archive] The set of servo values (constituent of a chromosome) are stored in a training file along with the facial feature values. The training file will be used later for training the neural network.

c. [Selection] For the purpose of mutation two chromosomes are selected using the “Roulette wheel selection method” [34]. This ensures greater probability of selection of chromosome having higher fitness than others.

d. [Crossover and Mutation] The selected chromosomes are crossed over at randomly selected points giving rise to a new mutated offspring.

e. [Replace] The old collection of chromosomes are replaced with the new chromosomes.

f. [Test] The end condition is defined as the maximum value (i.e. 1) for the target facial feature. When the end condition is met, the loop is terminated.

The genetic algorithm considers each facial feature value in turn and determines its maximum and minimum value. The population for all generations and their corresponding control signal sequences are recorded and added to the database generated using random signal sequence. Figure 4-3 shows the results of one such runs.

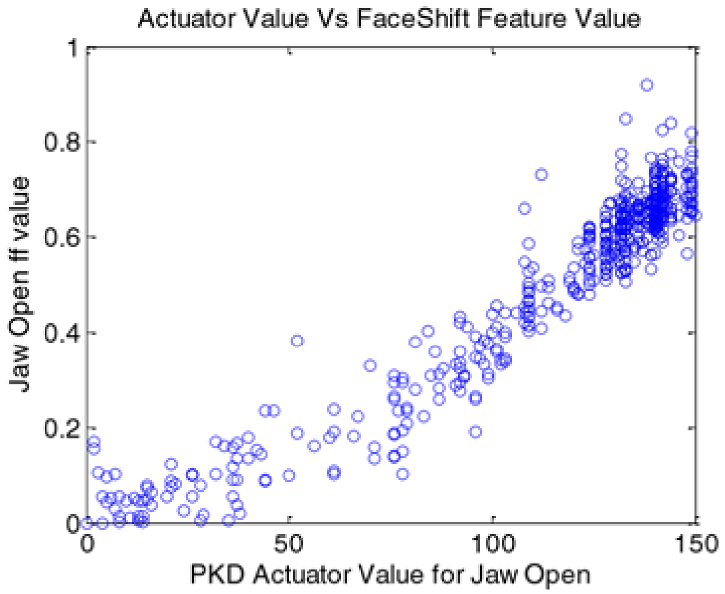


Figure 4-3 PKD facial feature value vs actuator value from data collected from GA based method

Here, the target facial feature considered is ‘jaw open’. The graph shows how this feature varies with the actuator which is primarily responsible for PKD’s jaw open. As can be seen from the graph, most of the data are collected from the extremities as expected and considerable noise is involved in the facial feature value.

4.3.1.2 Learning Algorithm

Figure 4-4 shows the plot of ‘jaw open servo values’ vs ‘jaw open ff values’ for all data collected via random generation and genetic algorithm. The relationship between jaw open servo and jaw open ff was expected to be linear but is not the case. A 3rd degree polynomial fits well with the graph as given below:

$$f(x) = -44.27x^3 - 88.72x^2 + 251x + 18.83 \quad (4.1)$$

Where, x is the facial feature value and f(x) is the corresponding servo value.

This was also seen for the case of other facial feature and servo values pair where linear relationships were expected. This is because the facial feature does not exactly coincide with the actuator. We used the graph (Figure 4-4) to see how well can translate certain facial expression from the user to PKD Android.

However, this mapping method is valid only in those facial features which can be achieved only via a single actuator, such as "jaw open". In many other cases, individual facial features depend on more than one actuator. A neural network is capable of handling this kind of one-to-many model.

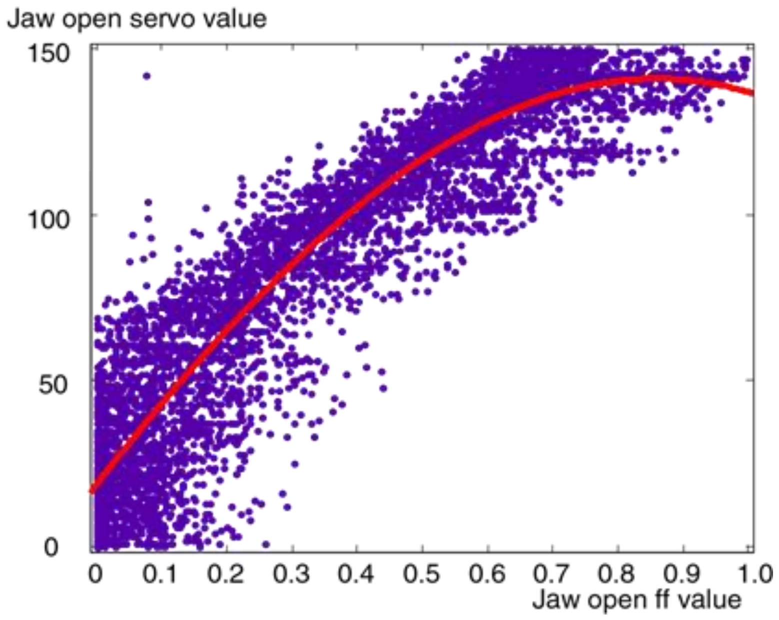


Figure 4-4 PKD Jaw Open ff values vs, Jaw open servo signal

In order to find the function that relates user's facial feature values, $O = [o1, o2, \ldots\ldots op]$, to PKD's servo signals, $S = [s1, s2, \ldots\ldots sk]$, where p and k are the number of facial feature tracked and the number of servo signals to be generated respectively. A three layer neural network with one input layer, one output layer and one hidden layer was used.

The problem statement can be formulated as:

$$S = \rho(O) \tag{4.2}$$

Where ρ is the required function to map $O => S$. The correctness of ρ can be tested by getting PKD's facial feature, $T = [t1, t2, \ldots\ldots, tn]$, as feedback and comparing it with user's facial features recorded earlier.

During training, the dataset generated earlier containing $D = \{S_{training}, T_{training}\}$, where $T_{training}$ is the facial feature values of PKD generated by signal $S_{training}$, is used. The servos, having variable operating range R ranging from [0- 255], were normalized before using them for training to transform them to the same scale.

$$S_{training,normalized} = \frac{S_{training} - Min(R)}{Max(R) - Min(R)} \tag{4.3}$$

We used a feed-forward neural network along with the Levenberg – Marquardt Algorithm (LMA) [35] to train the network weights (W_{inp}, W_{out}). The learning factor used for LMA is 0.001, the decrease ratio for learning factor was set to 0.1, the increase ratio for the learning factor was set to 10 and the maximum learning factor value was set to 1×10^{10}.

After training, facial feature values for different expressions are recorded for the user and it is passed through the neural network to generate normalized servo signal for testing:

$$S_{testing,norm} = \sum_{i=1}^{H} w_{out,i}\,\varphi\left(\sum_{j=1}^{P} w_{inp,i,j}\,o_j\right) \tag{4.4}$$

$$S_{out} = S_{testing,norm}(Max(R) - Min(R)) + Min(R) \tag{4.5}$$

Where H is the number of hidden layer nodes. The servo signals S_{out} are then used to create facial expressions on PKD and the corresponding facial features $T_{testing}$ are recorded.

After the neural network is trained, facial feature values can be used as input and the output will be servo values. These servo values are used to create the facial expression in PKD which will mimic the user's expression.

4.3.2 Experiment Setup

Figure 4-5 shows the experimental setup that we used for data collection and testing our proposed algorithms.

If experimental conditions are not controlled, the measurements are captured with considerable noise as depicted in Figure 4-3. In addition to the inherent noise of Kinect and Faceshift, the other major factors that affect the measurements are (1) change in ambient light during the experiment, (2) the presence of facial hair on PKD Andriod obstructs the view of Kinect especially in the lip section. (3) With the limited number of facial actuator on PKD it is difficult to produce satisfactory facial expression required for Faceshift training. This affects the Faceshift performance on PKD.

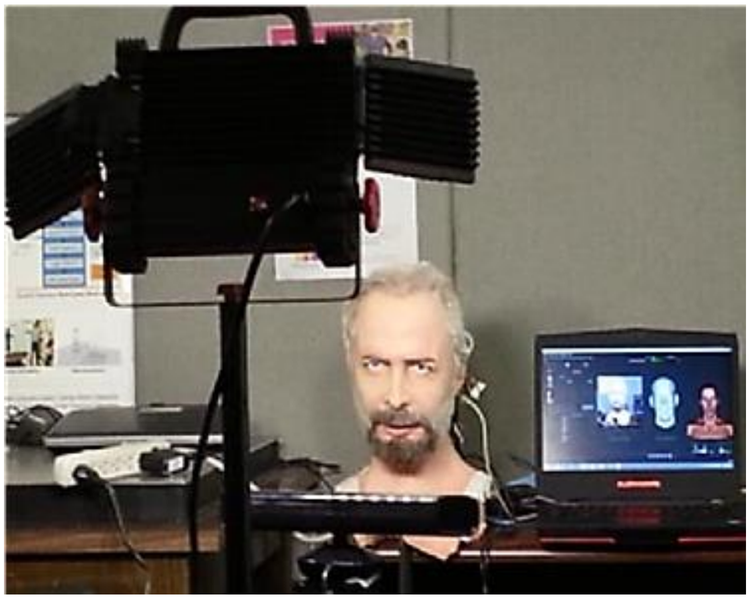


Figure 4-5 Experimental setup used for data collection in which lighting and Kinect sensor position relative to PKD are carefully controlled

In order to improve the learning process, we adjusted the experimental setup as follows: (1) The experiments were carried out in a darkened area using only a Husky

2500 Lumen Multi Directional LED Work light source which ensures the ambient light remains constant throughout the experiments (2) PKD was placed at an optimum distance (90-100cm) from Kinect for best results and tracking and it’s head was oriented directly toward the Kinect (4) the orientation was recalibrated before each tracking session (5) A moving average filter of length 5 was applied .

4.3.3 Experiment Results

4.3.3.1 Facial feature mapping

For the experiment we considered translating the lip motion from the user to the Android. For this we considered 9 actuators positioned around the lips and 9 relevant facial feature points extracted using face shift.

Using the experimental setup shown in Figure 4-6, a total of 4400 data was collected and used for training the neural network. Out of the collected 4400 data, 1400 sets were collected via random generation and the rest were generated via GA based optimization of various facial feature such as ‘jaw open’, ‘smile left’ & ‘simile right’. The input layer of the neural network contains 9 nodes, each representing a facial feature type that was tracked during the experiment. The hidden layer has 10 hidden units, and the output layer consists of 9 nodes each representing values of the servos in the mouth region. It was observed that the MSE of the neural network becomes steady if 10 or more hidden units were used in the hidden layer. Next data was collected from a human subject. The user produces a total of 17 different facial expressions and the corresponding facial features values were recorded. These facial feature values are feed into the trained neural network which produces the desired servo values. The local notebook then passes on the servo values to the controllers in PKD which will reproduce the facial expression in PKD.

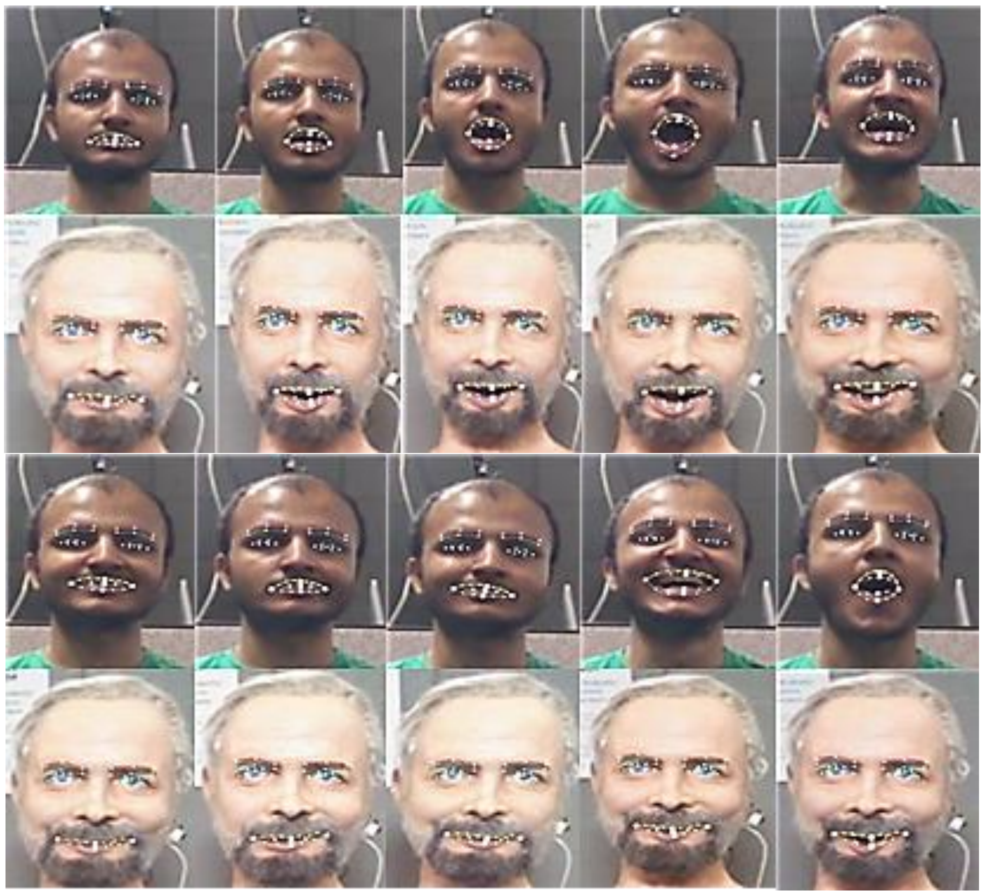


Figure 4-6 Facial expressions produced by user and Android pair

The facial features extracted via Faceshift are recorded and used for comparison with that obtained from the user. Figure 4-6 shows the expressions replicated by PKD from the user. It is evident from the figure that the neural network has learned to activate multiple servos to produce complex expressions like smile, pucker etc.

The facial features set we obtained from each expression pair was compared using the 1 and 2- norm distance metrics below, and summarized in Table 4-1:

Table 4-1 Error calculation

Distance	Facial features			
	Jaw Open	***Smile left***	***smile right***	***Lip stretch right***
2-norm	0.6423	1.0609	1.5559	1.2670
MSE	0.0242	0.0662	0.1424	0.0944

Good performance (lower MSE & norm-2) was observed for expressions that involve the lower lip region and moderate performance for the upper lip region. It was observed that error was induced during the tracking process due to facial hair (moustache) on PKD. The overall aesthetic performance for most of the mouth facial feature was excellent.

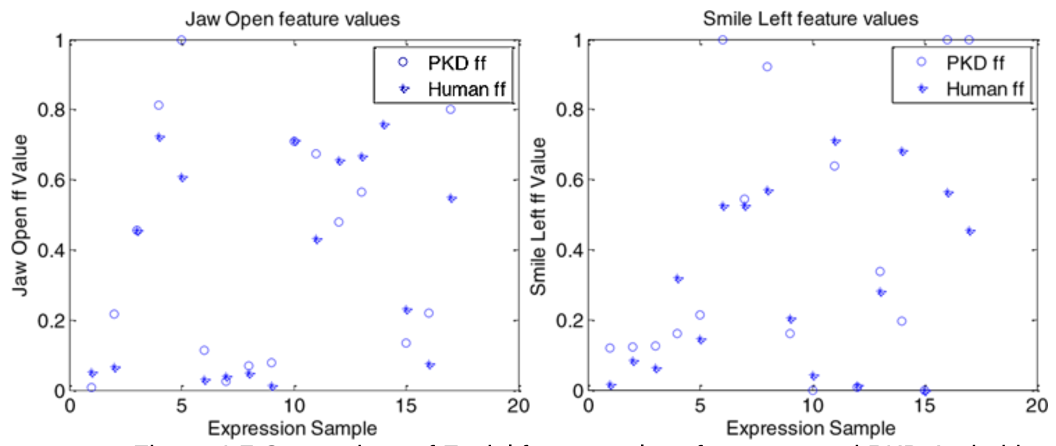


Figure 4-7 Comparison of Facial feature values for a user and PKD Android

The facial feature values for jaw open and left smile, extracted from various expressions of user and PKD, are shown in Figure 4-7. It can be seen that the error is significant when the facial feature is at its extremities. As GA explores the extremities intensively, data collected for Neural Network training under its guidance is thereby less prone to errors.

4.4 Lip Syncing Abilities for PKD

The idea of building an android head similar to a human being is incomplete without its ability to strike a conversation with users as a part of interaction. Speech is a combination of audio and visual interaction [22]. The sounds of the words combined with shape of the lip while speaking gives an idea of speech. Along with the formation of the lip structure, the motion during the transition between the visemes also plays an important role to reproduce the human lip movements during speech.

It has also been studied that with misrepresentation of the sound with that of a different lip movements also gives rise to the confusion of the syllable spoken [5]. If a person sees that the lip movement suggests one syllable is being spoken whereas another similar syllable is being heard, the person perceives a totally different syllable. This effect is popularly known as the “McGurk effect” [22] [5]. According to the experiment lip movements were recorded for similar syllables along with their voices, such as /ga/, /ba, /pa/ or /ka/ [5]. But when the video was played back to the subjects, the voice and the video of the lip movements were interchanged. It was observed that the spoken syllables were perceived differently by the subjects [5]. Table 4-2 represents the “McGurk effect”, where it is shown how the mismatched audio and video of a syllable was recognized by the audience.

Table 4-2 McGurk Effect results [5]

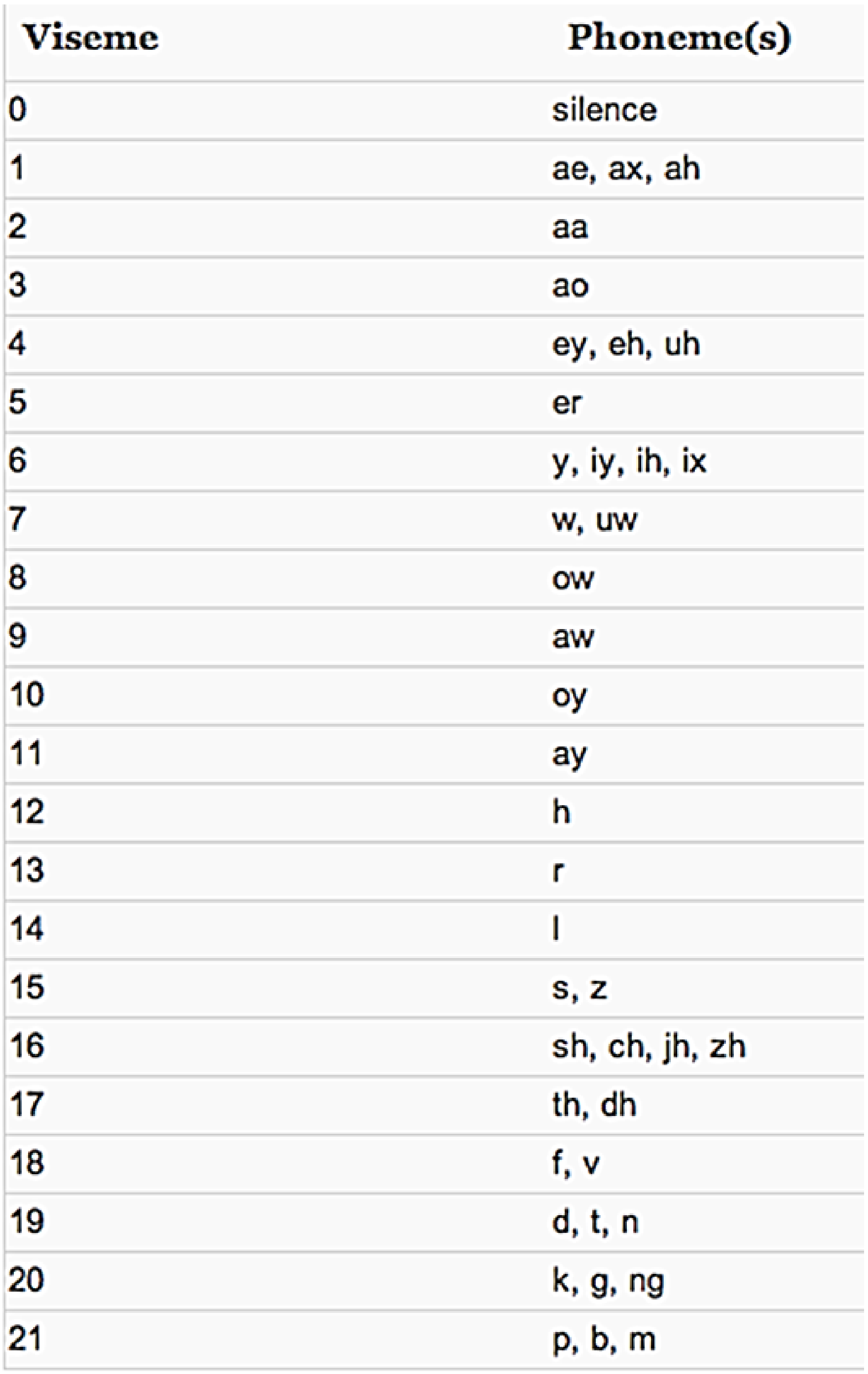
Stimuli		Response
Audio	Video	
ba-ba	ga-ga	da-da
pa-pa	ka-ka	ta-ta

According to the “McGurk effect”, if there is a loss of synchronization between the words spoken by PKD and the lip movements of PKD, it could give rise to confusion while in the midst of a conversation with a user. In order to give PKD a human like appearance, it is highly imperative that the synchronization is perfect between the lip movements and the audio of the conversation.

In order to achieve the synchronization, the shape of the lips along with duration for which it has to be maintained while speaking a sentence should be taken into account [22]. The controls aspect of controlling the lip movement should take into consideration the baud rate of data transfer, processing time for the speech and the speed of the speech.

Similar studies have been conducted with “FR-i”, a robot at KIST [22], which has 5 degrees of freedom in the mouth. Preston Blair phoneme series [23] was adopted for the experiment as it is popular for lip synchronization in animation and can be used on android robots [36]. Another study conducted by Intelligent Robotics and Automation Laboratory, National Taiwan University, on human robot interaction used the Microsoft Speech API which is based on the Disney visemes [37]. The Disney viseme has 21 types out of which 16 types were adapted in the study. The visemes to phonemes mapping can be seen in Figure 4-8 and the animated articulation of the visemes can been seen in Figure 4-9.

All of these aspects govern the issues related to the lip synchronization of PKD. PKD has a 9 degree of freedom at its mouth which can help in articulating the shapes of the visemes or lip shapes while ‘speaking’.



Viseme	Phoneme(s)
0	silence
1	ae, ax, ah
2	aa
3	ao
4	ey, eh, uh
5	er
6	y, iy, ih, ix
7	w, uw
8	ow
9	aw
10	oy
11	ay
12	h
13	r
14	l
15	s, z
16	sh, ch, jh, zh
17	th, dh
18	f, v
19	d, t, n
20	k, g, ng
21	p, b, m

Figure 4-8 Viseme - Phoneme List [38] [39]

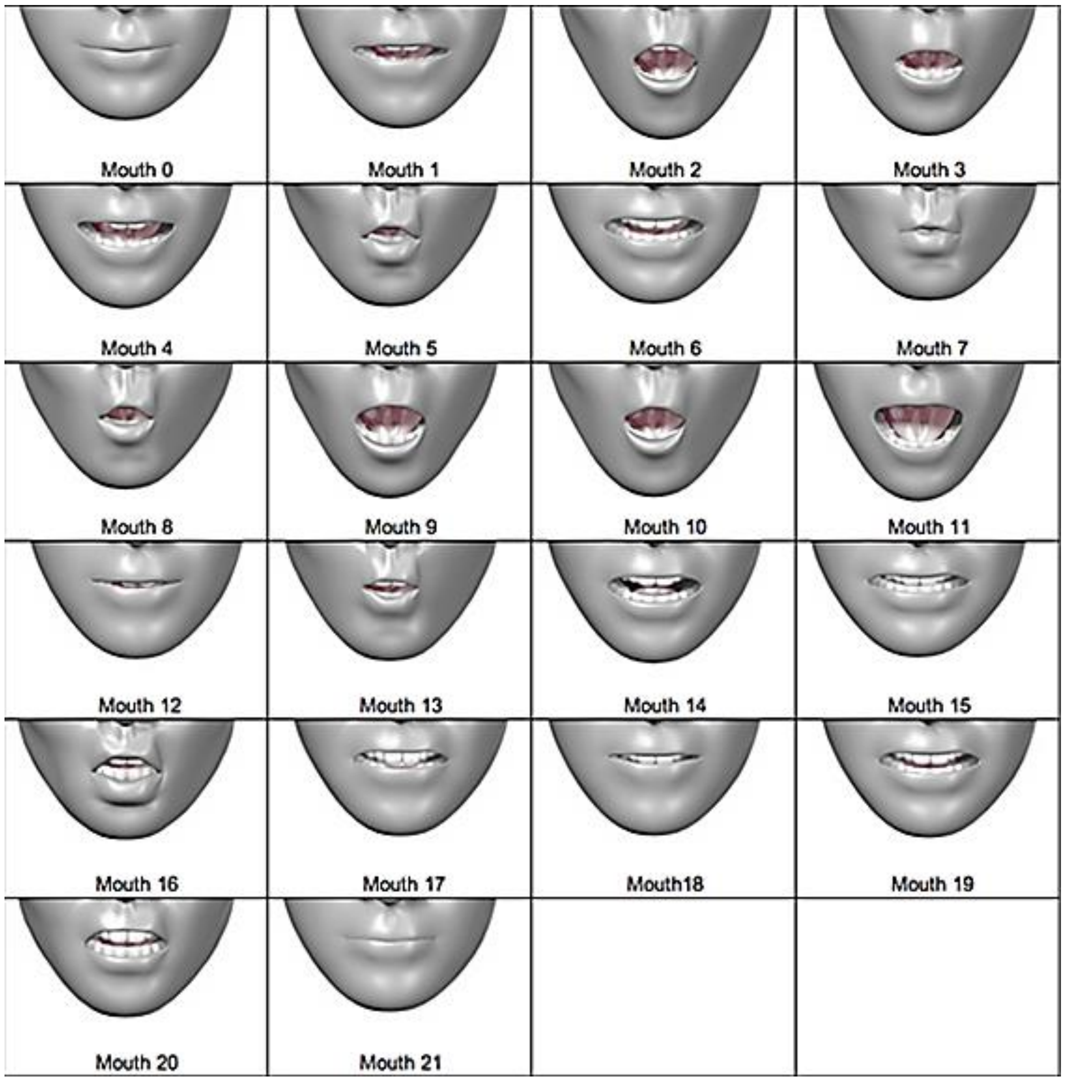


Figure 4-9 Viseme Chart [38] [39]

4.5 Mouth Control of PKD

As illustrated in Figure 4-10, there are 9 actuators controlling the movements and shape of the Lip. In order to control the lip motion, there actuators have to be controlled in such a manner that it emulates human lip movements.

The speed of the actuators has to be calibrated such that the transition between visemes should be human like. If any of the actuator is moving faster or slower than the others, even if the desired lip shape formation is achieved, the transition will not be convincing enough for it to be a human replica. It could also run into the danger of falling into the “uncanny valley” zone.

While ‘speaking’, a special control system has to be in place to correct the lagging or leading of the lip movements in comparison to the audio. While a feedback can be obtained from a camera pointed at PKD’s lip, the delay in getting feedback data has to be taken into account. In order to calculate the communication delay, the following formula is used [22]:

$$\mathrm{O} = \frac{1}{baudRate} \times [startBit(1) + data(8) + stopBit(1)] \times (No.of\ packets\ for\ lipMovement) \tag{4.6}$$

Here O is the delay induced while data is communicated to the actuators and the intended lip shape is formed. Another delay can be induced for the servo to respond to the data and move to the actual intended position. The speed of a servo is defined as x sec @ y degrees. The time required for the servo to complete the rotation is then calculated as [37]:

$$\mathrm{T} = (x)\ \times \frac{DesiredAngle}{y} \tag{4.7}$$

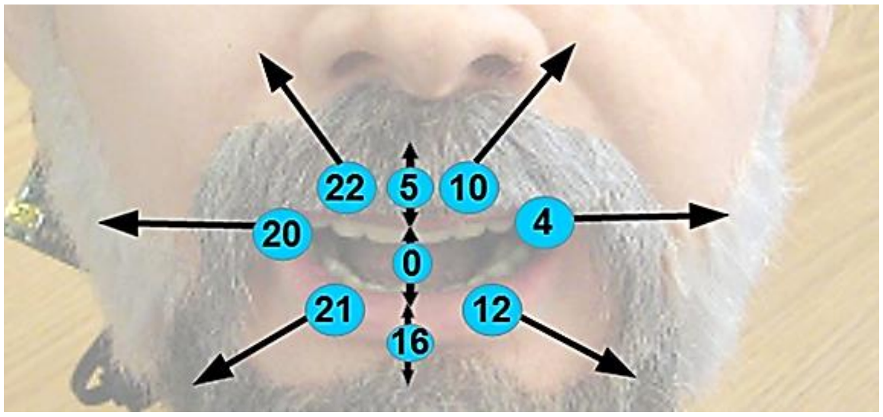


Figure 4-10 PKD Lip Actuators

The delays induced by Equation (4.6) and (5.23) are directly proportional to the number of actuators we have to communicate with in order to form the desired lip shape. These delays, if not accounted for, can cause incoherency between the audio and visual aspect of speech by PKD.

4.6 Remaining Challenges

To enable lip synchronization feature in PKD, there are many challenges that has to be addressed. The most important being the control system discussed in the previous section. Along with the communication delay and the actuator delay, there are delays induced by the compiler, debugger or the computer in use. These delays have to be measured and accounted for. Also the speed of the motor under the load exerted by the skin has to be calculated.

If a feedback to the control system can be implemented by pointing a camera at the lip of PKD and recognizing the shape of PKD. If the feedback is implemented, the noise and delay induced by it has to be considered. The visual feedback can only be provided, if the lip recognition program provides the lip shape data in real-time.

Along with the delays that disturb the synchronization of the viseme and phoneme, the burnout risk of the actuators, due to fast movements, has to be accounted for. The fast movements of the actuator can also cause damage to the Frubber skin. In the event of incoherent motion by the actuator, the stretching of the lips can cause damage as well.

There are several visemes which requires PKD to form puckered lips to be formed. Due to mechanical constraints, these visemes cannot be reproduced by PKD. In light of all these factors, the lip syncing ability for PKD has to be developed.

Chapter 5

Head-Eye Control For Target Tracking

5.1 Introduction

Target tracking by an android head involves the complex neck-eye coordination problem to be addressed. The target tracking process involves detecting the target, estimating its distance and pointing PKD's gaze direction towards it. During target tracking, if the target is moving too fast, it will get out of the frame of the camera and will cause loss in detecting and tracking. In earlier work on the tracking process, PID controller was developed which tracked faces by using the camera in the right eye of PKD. The objective of the project was to reduce the distance between the detected face or object and the center of the camera. The issue with using the camera in the eye of PKD was that the detection range is very small, which required the user to be very close to PKD in order to be detected along with perfect lighting condition. Therefore in the work presented, the use of external RGBD camera was used which boosts the target detection range during the tracking process without the need of ensuring extra lights.

In the project, there was face tracking, where PKD has to track the face of the user it is interacting with and there was object tracking, which used Emgu CV package along with Kinect's color-stream to detect objects using color and shape based detection. The 57 degree of horizontal FOV and 43 degree of vertical FOV allows a wide viewable area. Therefore rather than depending on the eye camera, the Kinect is used for tracking and estimating the distance of the target. The tracking process is described in the following section.

5.2 Tracking Aided By an External RGBD Camera

After the target is detected, its pixel index is determined by:

$$pixel\_Index = x(image_{width}) + y \tag{5.1}$$

Where, x and y are the horizontal and vertical pixel co-ordinates of the detected target in the image stream. The pixel index is then used to get the distance from the depthstream. Once we have the distance of the target, its real world co-ordinates can be calculated by using the pin-hole camera model:

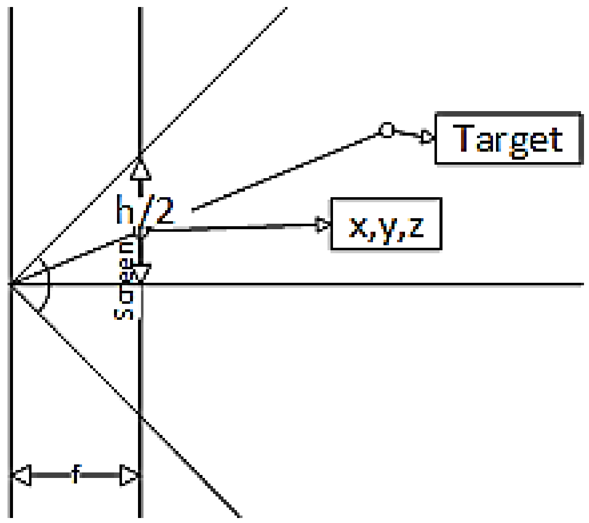


Figure 5-1 Pinhole camera model

$$\varphi_h = \frac{image_{width}}{\left(2\ \tan\frac{FOV_h}{2}\right)} \tag{5.2}$$

$$\varphi_v = \frac{image_{height}}{\left(2\ \tan\frac{FOV_v}{2}\right)} \tag{5.3}$$

$$X_{target}^{Kinect} = \frac{(x\ D)}{\varphi_h} \tag{5.4}$$

$$Y_{target}^{Kinect} = \frac{(y\ D)}{\varphi_v} \tag{5.5}$$

$$Z_{target}^{Kinect} = D \tag{5.6}$$

Where, φ_h and φ_v are the focal distances of the Kinect as calculated from the image resolution and the field of view (FOV),X_{target}^{Kinect}, Y_{target}^{Kinect} and Z_{target}^{Kinect} are the distance co-ordinates of the target in x, y and z direction respectively w.r.t. the base of the Kinect and D is the depth of the pixel, which is represented by x and y, as perceived by the depth camera.

Once the target's co-ordinates with respect to the base of the Kinect as origin is found out, the frame of reference is changed to PKD's frame of reference which can be found out by :

$$\begin{bmatrix} X_{target}^{PKD} \\ Y_{target}^{PKD} \\ Z_{target}^{PKD} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & X_{PKD}^{Kinect} \\ 0 & 1 & 0 & Y_{PKD}^{Kinect} \\ 0 & 0 & 1 & Z_{PKD}^{Kinect} \end{bmatrix} \cdot \begin{bmatrix} X_{target}^{Kinect} \\ Y_{target}^{Kinect} \\ Z_{target}^{Kinect} \\ 1 \end{bmatrix} \quad (5.7)$$

A similar transformation operation can be done to find out the location of the target w.r.t. the eye of PKD by following the below mentioned matrix multiplication:

$$\begin{bmatrix} X_{target}^{PKD_{Eye}} \\ X_{target}^{PKD_{Eye}} \\ Z_{target}^{PKD_{Eye}} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & X_{PKD_{Eye}}^{PKD} \\ 0 & 1 & 0 & Y_{PKD_{eye}}^{PKD} \\ 0 & 0 & 1 & Z_{PKD_{eye}}^{PKD} \end{bmatrix} \cdot \begin{bmatrix} X_{target}^{PKD} \\ Y_{target}^{PKD} \\ Z_{target}^{PKD} \\ 1 \end{bmatrix} \quad (5.8)$$

After determining the distance and co-ordinates of the target w.r.t. PKD's eyes, the information can be now used in different controller programs to drive the gaze direction of PKD towards the target.

5.3 Controllers Implemented in PKD

The controllers were implemented by calculating the location and velocity of the target. In the work presented, I have implemented the open-loop controller and the back-stepping controller. The back stepping controller was implemented by solving the system dynamics of the human head and a potential controller algorithm [24].

5.3.1 Target tracking by External Camera

This controller is implemented, assuming that PKD's neck and eye are on a straight and rigid line. It is also taken into account that while the neck and eyes are in motion, they move according to the motion of the line. Once this is established, the calculation of the gaze direction is done considering that the eyes never move around its axis, therefore the gaze direction is perpendicular to the line connecting the neck and the eyes. Therefore, in order to calculate the gaze direction, the actuators at the neck of PKD are actuated to achieve the desired angles. In the calculations, as only the neck actuators of PKD is taken into account, the location of the target w.r.t. the base of the neck is considered as calculated in Equation (5.7).

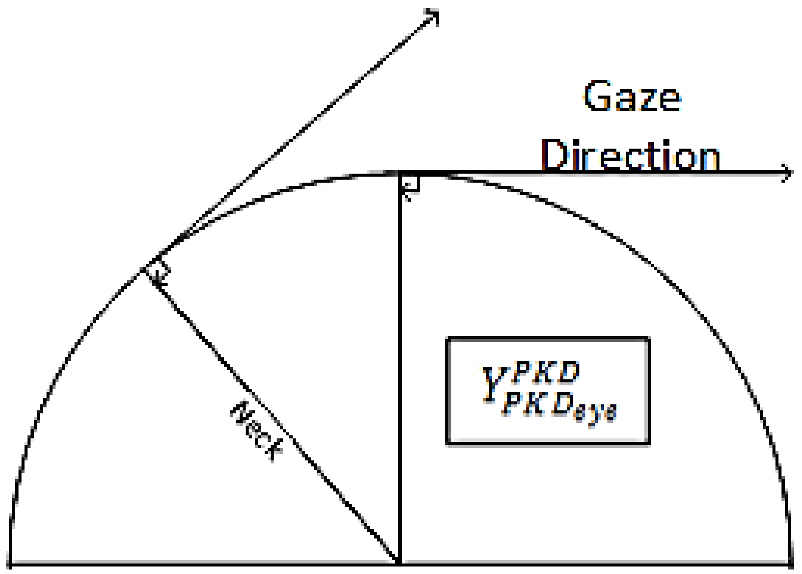


Figure 5-2 Neck and Eye orientation

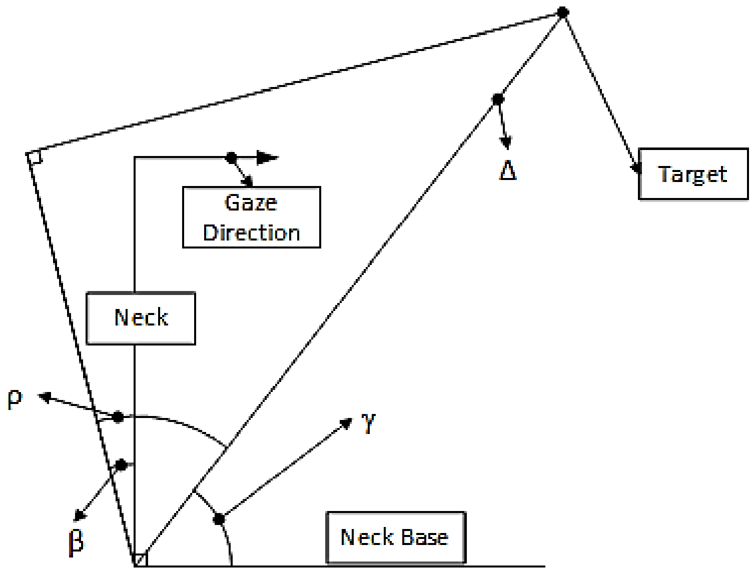


Figure 5-3 Angle calculation during tracking

Let the object distance from the neck of PKD be Δ which can be calculated by:

$$\Delta = \sqrt[2]{(X_{target}^{PKD})^2 + (Y_{target}^{PKD})^2 + (Z_{target}^{PKD})^2} \tag{5.9}$$

The calculation for the pitch angle (β) of the neck of PKD is calculated as:

$$\rho = \cos^{-1}(Y_{PKD_{eye}}^{PKD} \div \Delta) \tag{5.10}$$

$$\gamma = \tan^{-1}(Y_{target}^{PKD} \div Z_{target}^{PKD}) \tag{5.11}$$

$$\beta = \rho + \gamma - 90° \tag{5.12}$$

Similarly, the pan angle (α) of the neck is calculated as:

$$\alpha = \sin^{-1}(X_{target}^{PKD} \div \sqrt[2]{(X_{target}^{PKD})^2 + (Z_{target}^{PKD})^2}) \tag{5.13}$$

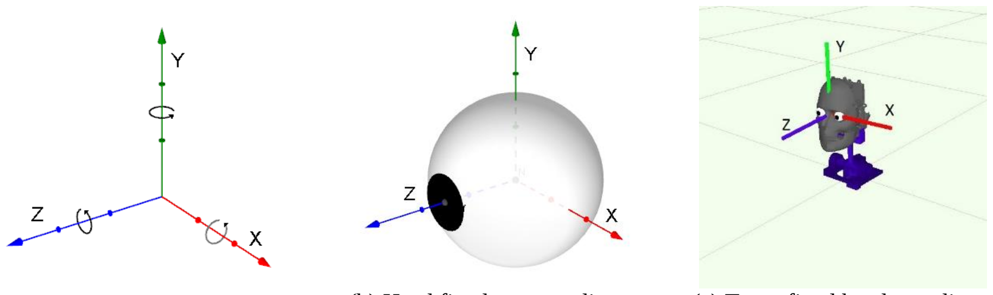
After the pan and the tilt angles are calculated, the angles are transformed to the actuator values based on the calibration of the servos.

5.3.2 Potential Controller

The essential idea behind the potential control is to exploit the fact that a system tends to minimize its total energy. A hypothetical potential function is suitably constructed such that the potential energy is minimum at the required spatial orientation. A damping term is also added to simulate dissipative friction and active damping by the muscles.

5.3.2.1 Parameterization of SO(3)

We employ the Y-X-Z Euler angles (Figure 5-4(a)) for modeling the eye and head dynamics. The head fixed eye coordinate system and the torso fixed head coordinate system are shown in Figure 5-4(b) and Figure 5-4(c) respectively with the eye and the head in their primary orientations. In the subsequent calculations, the superscripts h and e are used to denote head and eye angles $\emptyset_i^h$ and $\emptyset_i^e$, i = 1, 2, 3.



(a)Y-X-Z Euler angles (b) Head fixed eye coordinate system (c) Torso fixed head coordinate system

Figure 5-4 Coordinate systems used for the implementation. The eye and the head are in their primary orientations.

5.3.2.2 Donders' Constraint

The Donders' constraint can be seen as a perturbation in $\emptyset_3$ by expressing it as a function of $\emptyset_1$ and $\emptyset_2$. In literature, a general Donders' constraint is expressed as a quadratic expression that translates into the Euler angles [40] as

$$a(\emptyset_1^h, \emptyset_2^h)\tan^2\frac{\emptyset_3^h}{2} + b(\emptyset_1^h, \emptyset_2^h)\tan\frac{\emptyset_3^h}{2} + c(\emptyset_1^h, \emptyset_2^h) = 0 \tag{5.14}$$

It can be noted that when $\emptyset_3^h \equiv 0$, the head follows a pan-and-tilt motion which is less human-like. Unlike in the case of Listing's constraint, however, the Donders' constraint is not fixed but can vary from person to person.

For simplicity, in this work, we express the Donders' constraint as:

$$\tan\frac{\emptyset_3^h}{2} = B\tan\frac{\emptyset_1^h}{2} \tag{5.15}$$

$$\emptyset_3^h = D(\emptyset_1^h, \emptyset_2^h) = 2\arctan(B\tan\frac{\emptyset_1^h}{2}) \tag{5.16}$$

Where B is a constant. However, the method can easily make use of the full quadratic form (5.20).

5.3.2.3 Modeling the Head/Eye Movements

The goal of this section is to derive the equations of motion of the head and the eyes in their respective angle spaces. Since both the head rotations and the eye rotations are modeled using Y-X-Z Euler angles, both sets of equations of motion have the same geometry. A detail that simplified the derivation of equations of motion in this work is that PKD has position and velocity control as opposed to torque control. Due to this, the models could be abstract with respect to the moments of inertia of the rotating bodies allowing for lesser distinction between head and eye models. For both models, the moment of inertia tensor is considered to be the identity matrix. However, the modeling process with general moment of inertia tensors was considered in [41].

The equations of motion are derived through the Lagrangian approach which requires expressions for kinetic and potential energies of the system. Let $\Theta = (\emptyset_1, \emptyset_2)$ and $q(\Theta) \in S^3$ where q is the quaternion representation of the rotation. The explicit expression of q can be given [41] as

$$q(\emptyset_1, \emptyset_2) = \begin{pmatrix} \sin\frac{\emptyset_1}{2}\sin\frac{\emptyset_2}{2}\sin\frac{\emptyset_3}{2} + \cos\frac{\emptyset_1}{2}\cos\frac{\emptyset_2}{2}\cos\frac{\emptyset_3}{2} \\ \cos\frac{\emptyset_1}{2}\sin\frac{\emptyset_2}{2}\cos\frac{\emptyset_3}{2} + \sin\frac{\emptyset_1}{2}\cos\frac{\emptyset_2}{2}\sin\frac{\emptyset_3}{2} \\ \sin\frac{\emptyset_1}{2}\cos\frac{\emptyset_2}{2}\cos\frac{\emptyset_3}{2} - \cos\frac{\emptyset_1}{2}\sin\frac{\emptyset_2}{2}\sin\frac{\emptyset_3}{2} \\ \cos\frac{\emptyset_1}{2}\cos\frac{\emptyset_2}{2}\sin\frac{\emptyset_3}{2} - \sin\frac{\emptyset_1}{2}\sin\frac{\emptyset_2}{2}\cos\frac{\emptyset_3}{2} \end{pmatrix} \tag{5.17}$$

Where $\emptyset_3^h$ is calculated via Equation (5.21) and (5.28) and $\emptyset_3^h \equiv 0$. The kinetic energy with identity inertia tensor is then given by

$$KE = \frac{1}{2}G\dot{\Theta} \tag{5.18}$$

Where

$$G = J^T J \; with \; J = \frac{\delta q}{\delta \Theta} \tag{5.19}$$

G for eye movements with $\emptyset_3^e \equiv 0$ simplifies to 1/4 the identity matrix whereas the expression for head movements with the Donders' constraint is too large to state explicitly. It has been shown in [41] that Equation (5.18) is equivalent to the conventional expression $KE = \frac{1}{2}\omega^T I \omega$ up to a constant multiplication.

If the potential energy of the system is represented by V (Θ), the Lagrangian can be expressed as L = KE − V. In the Lagrangian approach, the equations of motion are obtained using the Euler-Lagrange equation and the resulting equations of motion are given by

$$\ddot{\Theta} = G^{-1}\left[\breve{\Gamma} - \dot{G}\dot{\Theta} + \frac{1}{2}\nabla(\Theta^{T}\dot{G}\dot{\Theta})\right] \tag{5.20}$$

$$\breve{\Gamma} = \Gamma - \nabla V \tag{5.21}$$

Where ∇ represents the gradient with respect to Θ. When suitable expressions are given for Γ and V, Equation (5.20) and (5.21) are called the potential control.

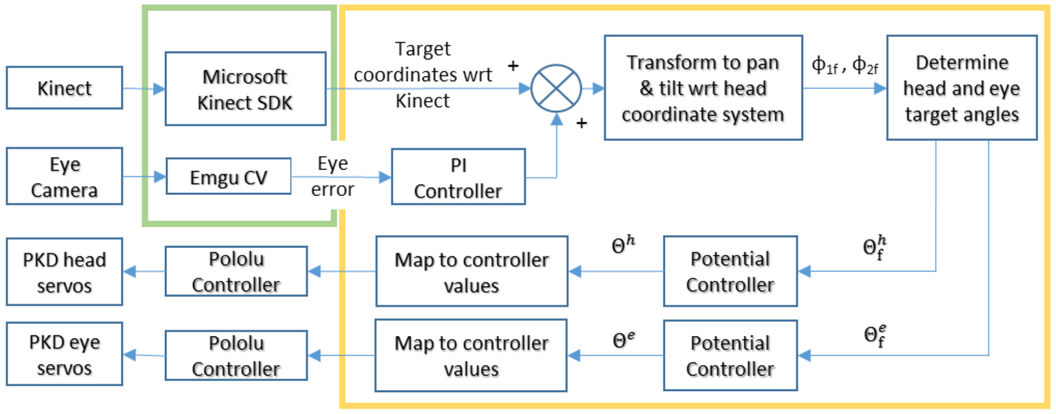


Figure 5-5 System diagram of the potential control based target tracking

In accordance with [40], we choose the potential function V and the generalized torque Γ as

$$V = A(1 - |q(\Theta).q(\Theta_f)|) \tag{5.22}$$

$$\Gamma = -cG\dot{\Theta} \tag{5.23}$$

Where A and c are scalar constants and $q(\Theta_f)$ is the quaternion representing the target orientation. In (5.22), (·) represents the vector dot product. The motivation of (5.22) is to have a potential energy profile proportional to the angle between $q(\Theta)$ and $q(\Theta_f)$ in $\mathbb{R}^4$ while identifying q with $-q$, both of which represent the same orientation. The generalized torque Γ is such that it opposes the angular velocity of the rotating body simulating active damping and friction; it can be easily shown from the relations in [41] that $\Gamma = -cG\dot{\Theta}$ is equivalent to $\mathrm{T} = -c\omega$ when T is the external torque.

5.3.2.4 Controller Implementation

The goal of this section is to describe the implementation of the potential controller in conjunction with the hardware. The basic components of the controller are shown in Figure 5-5 as a block diagram. In the diagram, Kinect, eye camera, PKD servos and the Pololu controller represent hardware components while the rest are software components implemented in Visual C#.

As shown in Figure 5-5, the software was implemented as two separate threads communicating through the Windows clipboard. This implementation separates the object recognition from the controller providing a smoother operation. The object recognition module utilizes the circle identification functions in both Visual Studio Kinect Toolkit and Emgu CV to identify the white circular object which was used as the target. The target coordinates with respect to the Kinect base frame, $P^k = (x^k, y^k, z^k)$ (mm), and the eye camera error $e = (e_x, e_y)$ (pixels) defined by

$$(e_x, e_y) = (s_x, s_y) - (c_x, c_y) \tag{5.24}$$

Where c and s represent eye camera image center and the object center respectively, are then passed to the controller module.

The controller module calculates the target coordinates $P^h = (x^h, y^h, z^h)$ with respect to the torso fixed head coordinate system and then calculates the pan and tilt angles $\Theta_f = (\emptyset_{1f}, \emptyset_{2f})$ of the target. Vectors P^h and Θ_f are given by

$$P^h = P^k + P^h_{k\,origin} \tag{5.25}$$

$$\Theta_{1f} = \arctan\left(\frac{x^h}{z^h}\right) \quad \Theta_{2f} = -\arcsin\left(\frac{y^h}{\|P^h\|_2}\right) \tag{5.26}$$

Where $P_{k\ origin}^{h}$ origin is the coordinates of the origin of Kinect base frame with respect to the head coordinate system, and $\|.\|_2$ indicates the Euclidean norm of a vector. The third angle $\emptyset_{3f}$ is not calculated as the yaw does not affect the heading direction. In this work, the pan and tilt angles of the target are divided between the head and the eyes at a ratio of 1:2 resulting in

$$\Theta_f^h = \frac{1}{3}\Theta_f, \quad \Theta_f^e = \frac{2}{3}\Theta_f \tag{5.27}$$

Since the eye movements of PKD is restricted to pan and tilt by hardware design, Θ_f^e is equated to zero. However, the yaw of the head target, Θ_{3f}^h , is determined by the Donders' law and is calculated through Equation (5.15) and Equation (5.16).

Target head and eye angle vectors Θ_f^h and Θ_f^e are then passed to the corresponding potential controllers. Each of the head and the eye controllers use a separate set of (5.20), (5.21), (5.22) and (5.23). The Runge-Kutta 4-5 method is then employed for numerical integration of (5.20) and (5.21) together with (5.22) and (5.23) that provides the trajectories of head and eye angles and their derivatives which are fed to the corresponding servo motors through the Pololu controller. Due to the complexity of the closed form expressions, only (5.15), (5.16) and (5.17) are expressed explicitly and all the derivatives and gradients are numerically calculated.

When enabled, the PI controller uses the standard form $k_p + \frac{k_i}{8}$ where k_p and k_i represent the proportional and integral gains respectively. If the eye camera fails to detect the object, the PI controller is disabled until the object is reacquired.

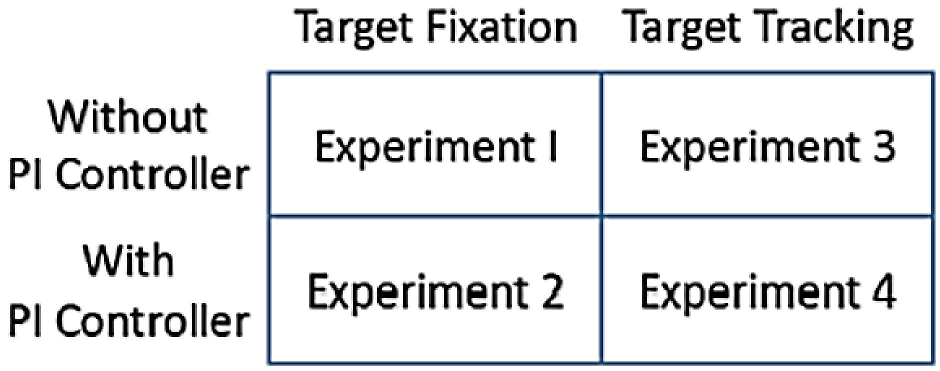
The potential coefficients A and the friction coefficients c were given the values 120 and 20 respectively whereas the proportional gain k_p and the integral gain k_i of the

PI controller were set to 0.5 and 0.0005, respectively. The time-step for the controller was set at 50 ms and the thread was awakened every 50 ms using a timer.

5.3.2.5 Experimental Design

The experiments can be classified into four types of trials based on the criteria as illustrated in Figure 5-6. Three trials were conducted for each of the four experiments. In target fixation, the object was moved from one place of the visual field of the Kinect to another abruptly. This results in the eye camera losing the target until the potential controller reacquires it. In contrast, in target tracking, the object was moved continuously and sufficiently slowly so that it stayed within the eye camera frame throughout the entire trial.

When the PI controller is not used, the systems acts as an open-loop system. When enabled, the PI controller is used on top of the potential controller to form a closed-loop system. However, even when used, the PI controller is only active when the object is detected in the eye camera. When the object is outside the eye camera frame, the PI controller is inactive and the system is controlled only by the open-loop potential controller with the data from the Kinect. When the PI controller is active, a feedback is implemented into the potential controller via the PI controller which together control the dynamics of the head-eye system.



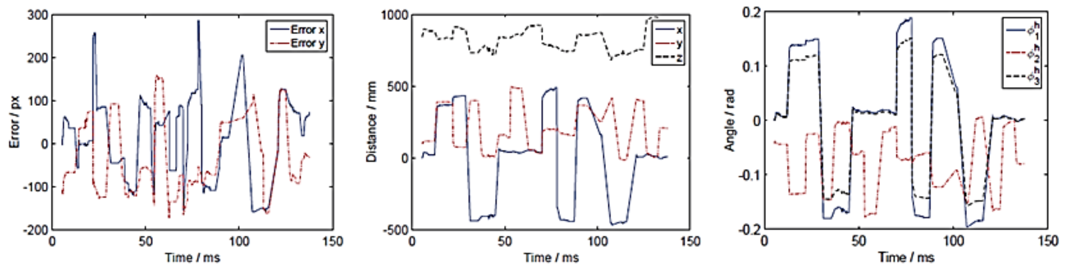
	Target Fixation	Target Tracking
Without PI Controller	Experiment I	Experiment 3
With PI Controller	Experiment 2	Experiment 4

Figure 5-6 Classification of the experiments

The accuracy of fixation and tracking were measured on the eye camera image as the distance from the image center to the object center in pixels. Due to imperfections in the object recognition through the eye camera, data were recorded only when an object was recognized in the eye camera image. For an abruptly moving target, this also means that the data is recorded only when the target is reacquired by the eye camera.

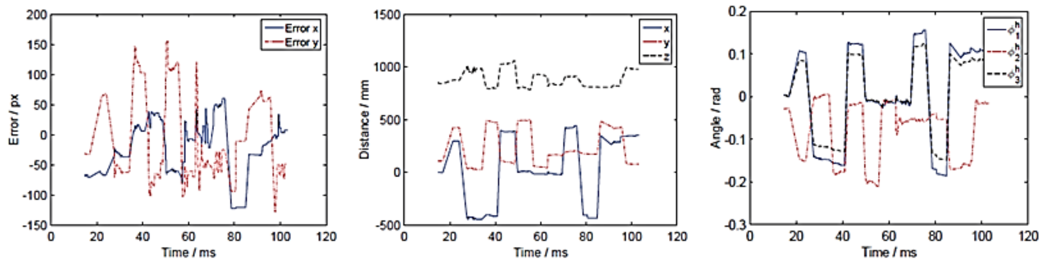
5.3.2.6 Experimental Results

In the experiments, the task of target fixation simulates the process of head and eyes moving fast to acquire a target. From a functional perspective, the process of target fixation primarily uses the Kinect and the open-loop potential controller as opposed to the eye camera and the closed-loop controller. This is due to the fact that the target mostly stays out of the eye camera frame and that the PI controller uses the eye camera information. Target tracking on the other hand simulates the smooth-pursuit of a moving target by the head-eye system. Unlike in the case of target fixation, the object is maintained within the eye camera image throughout the entire duration providing data to the PI controller. Therefore, when used, the PI controller is active throughout the entire trial. Similar to the former task, when the PI controller is not used, the dynamics of PKD is driven entirely by the Kinect data and the open-loop potential controller.



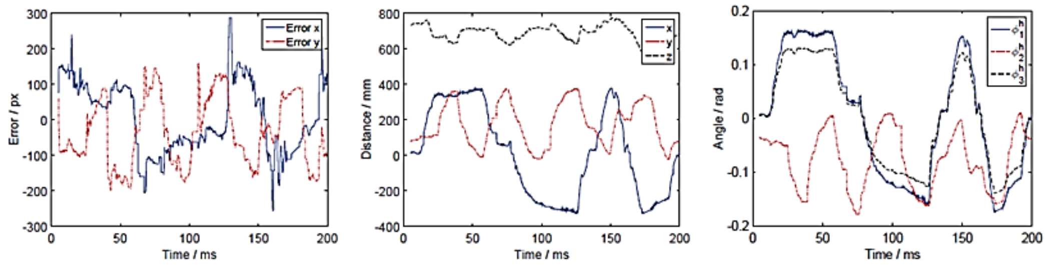
(a) Eye camera error (b) Target coordinates vs time (c) Head angles vs time

Figure 5-7 Target Fixation without PI controller



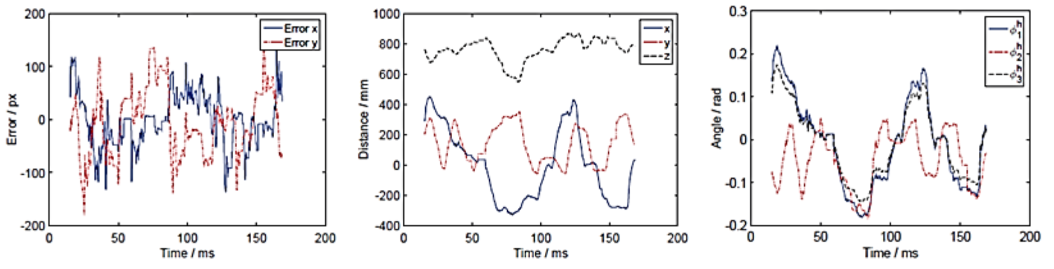
(a) Eye camera error (b) Target coordinates vs time (c) Head angles vs time

Figure 5-8 Target Fixation with PI controller



(a) Eye camera error (b) Target coordinates vs time (c) Head angles vs time

Figure 5-9 Target Tracking without PI controller



(a) Eye camera error (b) Target coordinates vs time (c) Head angles vs time

Figure 5-10 Target Tracking with PI controller

It is evident from Figure 5-7 and Figure 5-9 that the openloop potential controller can perform reasonably well on both tasks although the inclusion of feedback via a PI controller improves the performance as seen in Figure 5-8 and Figure 5-10. Table 5-1 lists the root mean square values of the eye error for all the trials conducted. These values indicate that the inclusion of the feedback with a PI controller reduces the tracking errors in all the experiments.

Table 5-1 Errors with and without PI controllers

	W/O PI controller	With PI controller
Fixating	123.32 103.12 116.01	87.70 88.49 90.46
Tracking	137.97 123.01 107.97	74.76 79.74 84.46

5.3.3 Back-Stepping Controller

The back stepping controller is a non-linear controller which stabilizes the zero dynamics of the system. While working on PKD, a lot of error is induced due to the jitter of the servos at the extremities and the absence of the dynamics of PKD to be taken into account during the calculations. The potential takes into account the dynamics of a human head. But PKD does not have the same dynamics as the human head as the placement of the actuators and their weights are not the same as a human.

Therefore, a back-stepping controller is designed based on the potential controller [24]. The dynamics of a human head is described as [24]:

$$G\ddot{X} + \dot{G}\dot{X} - \frac{1}{2}\dot{X}^T \nabla_X G\dot{X} + \nabla_X V - \lambda \nabla_X F = \Gamma \qquad (5.28)$$

Where G and F are constants which can be eliminated from the above equation, as $\dot{G}$→0 and $\nabla_X F$→0. Rewriting the above equation (Equation (5.28)):

$$G\ddot{X} + \nabla_X V = \Gamma \tag{5.29}$$

Where X represents the states of PKD i.e. position and acceleration. And V in the equation can be described as the potential field which drives the system to converge with the target. V is defined as:

$$V = A\left(1 - \left[\varphi_{gimbal}^{d\theta}\right] \times \left[\varphi_{gimbal}^{Target}\right]\right) \tag{5.30}$$

Where, $\varphi_{gimbal}^{d\theta}$ is the resultant quaternion for the gimbal angle of PKD head and $\varphi_{gimbal}^{Target}$ is the quaternion of the target. Therefore the whole system can be re-written as:

$$G\ddot{X} + (A \times u) = -cG\dot{X} \tag{5.31}$$

Where, $u = \left[\varphi_{gimbal}^{d\theta}\right] \times \left[\varphi_{gimbal}^{Target}\right]$ and$\Gamma = -cG\dot{X}$.

5.3.3.1 Back-Stepping Controller Design

This section illustrates the calculation required for deigning the back-stepping controller. The aim of the design is to replace the input u to the system with a stable controller. Let us assume that X_1 and X_2 define the states of the system such that:

$$X_1 = X \tag{5.32}$$

$$X_2 = \dot{X} \tag{5.33}$$

Taking the derivative of the Equations (5.32) and (5.33), using Equation (5.31):

$$\dot{X}_1 = X_2 \tag{5.34}$$

$$\dot{X}_2 = G^{-1}[-c \times G \times X_2 + A \times u] \equiv v_0 \tag{5.35}$$

Now Equating (5.34) and (5.35):

$$\dot{X}_1 - X_2 = 0 \tag{5.36}$$

$$\Rightarrow \dot{X}_1 - X_2 + X_{2d} - X_{2d} = 0 \tag{5.37}$$

$$\Rightarrow \dot{X}_1 - X_{2d} = -\dot{\hat{X}}_{2d} \left(assuming\ \dot{\hat{X}}_{2d} = X_{2d} - X_2\right) \tag{5.38}$$

$$\Rightarrow \dot{X}_1 + X_1 = -\dot{\hat{X}}_{2d}\ (assuming\ X_{2d} = -X_1) \tag{5.39}$$

Assuming a Lyapunov function candidate as:

$$V_L = \frac{1}{2}X_1^2 + \frac{1}{2}\hat{X}_{2d}^2 \tag{5.40}$$

Taking the derivative of the Equation (5.40):

$$\dot{V}_L = X_1\dot{X}_1 + \hat{X}_{2d}\dot{\hat{X}}_{2d} \tag{5.41}$$

$$\Rightarrow \dot{V}_L = X_1(-\hat{X}_{2d} - X_1) + \hat{X}_{2d}(-\dot{X}_1 - \dot{X}_2) \tag{5.42}$$

$$\Rightarrow \dot{V}_L = -X_1^2 - X_1\hat{X}_{2d} + \hat{X}_{2d}(-X_2 - v_0) \tag{5.43}$$

Assigning $v_0 = -X_2 - X_1 - \hat{X}_{2d}$, the Equation (5.43) becomes:

$$\Rightarrow \dot{V}_L = -X_1^2 - \hat{X}_{2d}^2 < 0 \tag{5.44}$$

Therefore from Equation (5.44), we deduce that the system is stable and the input u becomes:

$$u = \frac{1}{A}\left[G\left(-X_1 - X_2 - \hat{X}_{2d}\right) + cGX_2\right] \tag{5.45}$$

5.3.3.2 Experiment Results

Presented below are the plots of the back-stepping controller simulated in Matlab as compared to the simulation of the potential controller:

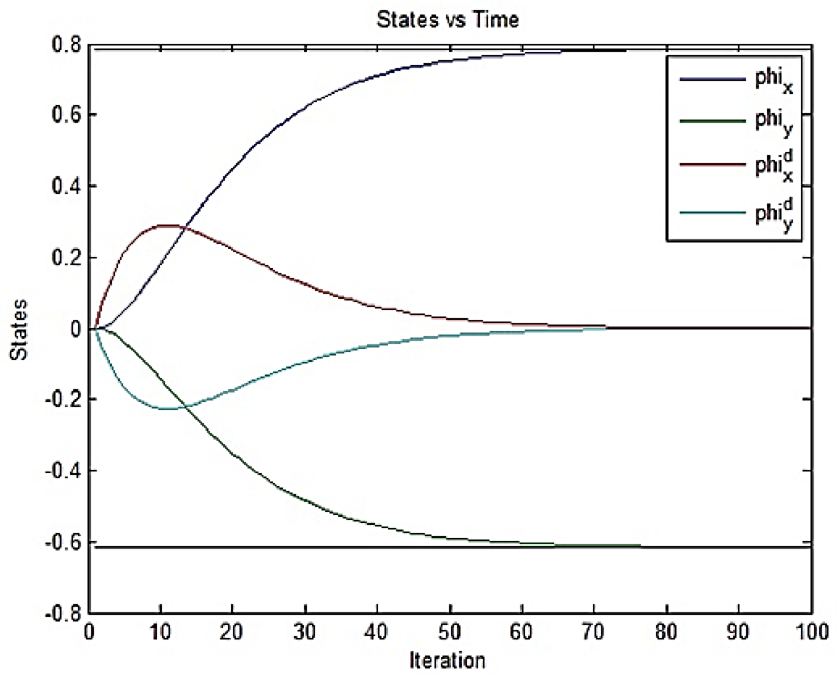


Figure 5-11 States Vs. Time plot of back-stepping controller

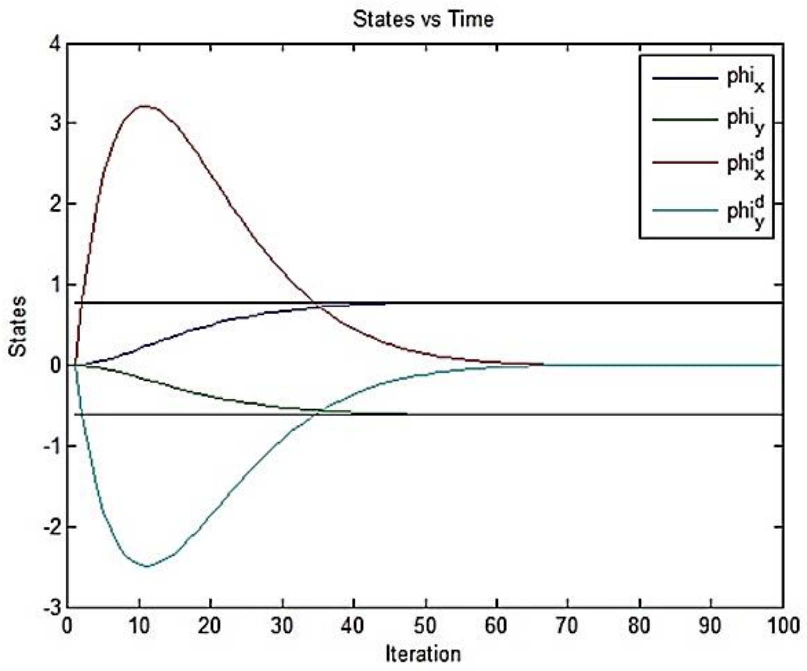


Figure 5-12 States Vs. Time plot of potential controller

The plots indicate that compared to potential controller, back-stepping controller exhibits smooth transition in position and velocity while trying to converge to required target. The potential controller exhibits sharp response during the process of tracking.

Another aspect of the analysis of the simulation result reveals the comparison of the controllers on the basis of time taken to accomplish the tracking process. Potential controller observed to be very fast at completing the tracking process as compared to the back-stepping controller.

Chapter 6

Conclusion and Future work

6.1 Conclusion

In the thesis, three aspects of social robot's ability to engage with humans and its surroundings were addressed. The aim of the work was to facilitate an android robotic head, PKD, to show human-like behavior while carrying out various conversational tasks.

6.1.1 Mimicking Facial Expression

We proposed and implemented a neural network to realistically generate facial expressions, and late mimic a user's facial expressions. Unlike past work that uses a linear mapping between actuators and expressions, the advantage of using neural networks for learning and reproducing expressions is that it can handle system non-linearities.

Our method replaces heuristic approaches for calibration of the actuators with an automated method to fine-tune appropriate facial expressions for PKD. This helps in cases where robot skins or actuators stretch and degrade over time, or if expression generation has to be applied to different hardware targets. The expressions are generated with respect to the trained neural network's output. The results indicate that the mimicking of facial expressions using neural networks has small MSE values.

6.1.2 Tracking Target

In addition to the web camera in the eye of PKD, external depth camera was implemented to increase the range of detection and robust tracking of a target. The RGB-D camera's wide FOV provides the advantage of a wider scene of the robot's surroundings.

Using the external camera an open-loop controller was developed which actuates the neck of the android head in order to point the gaze direction toward the target.

A potential controller was also implemented which is based on the human head dynamics. This controller took into consideration the laws governing the human head and actuated the android head towards the target

As the dynamics of the android head was not known, a non-linear controller, back-stepping controller, was proposed. During the simulation, it was observed that the rate of change of angles and speed required to actuate the android head was slowed down. Thus giving an impression that the executed motion is smooth and jitter-free.

Experimental results indicate that using potential controller with a PI controller increases the accuracy of the target tracking process. Simulation of back-stepping controller suggests a smooth transition of the actuators while converging to the target but the rate of convergence to the target is slow as compared to the potential controller.

6.1.3 Lip Synchronization

The problem of lip synchronization during speech was also explored in our thesis.

The mapping between the phonemes and visemes was studied and was observed that along with the shape of the lip, the motion executed while forming the shape of the lip is also as important. It was also observed that the failure to synchronize lip shapes with the word spoken can result in conveying wrong unintended messages.

6.2 Future Work

In future, the lip synchronization ability for PKD can be further developed taking into consideration the remaining challenges outlined in the thesis. A module can be developed to sync accurate visemes with the words spoken. This will allow PKD to converse with users and convey intended idea or message.

The jitters induced due to overshoot and controller instability is a major setback for PKD to be able to mimic humans. Better control systems can be put into place for reducing the jitter that is observed during the operation of PKD.

The back-stepping controller which has been proposed and simulated can be tested on PKD to validate its effectiveness and accuracy. The aim of the controller is to smooth the motion whereas simulation results suggests slower convergence rate. The trade-off between the rate of convergence and smoothness in motion during target tracking process can also be studied using the controller.

The ROS package contains simulation package for PKD for neck and eye motions only. This package can be further developed to simulate Frubber® skin on the robot, and animate facial expressions. The ROS package can be expanded to contain other research works which will contribute towards building a combined repository for android heads.

Human face has more than 40 muscles [42] to express emotions whereas PKD has only 24 actuators (excluding the neck actuators) to mimic them. A better hardware can be developed taking into account the muscle placement of a human face and its direction of motion. This will assist in better mimicking the human facial expressions and enabling PKD to look human-like.

A study can be conducted to re-evaluate the uncanny valley theory using PKD. The study should aim at exploring PKD’s interaction with users and their level of acceptance for the android. The study can compare the level of comfort a user has while interacting with PKD as compared to interacting with other robots whose appearance is not similar to that of a human. The study can also aim at evaluating the uncanny valley and the path of engagement theory to develop and improve PKD's interaction modules.

Appendix A

URDF Diagram of PKD

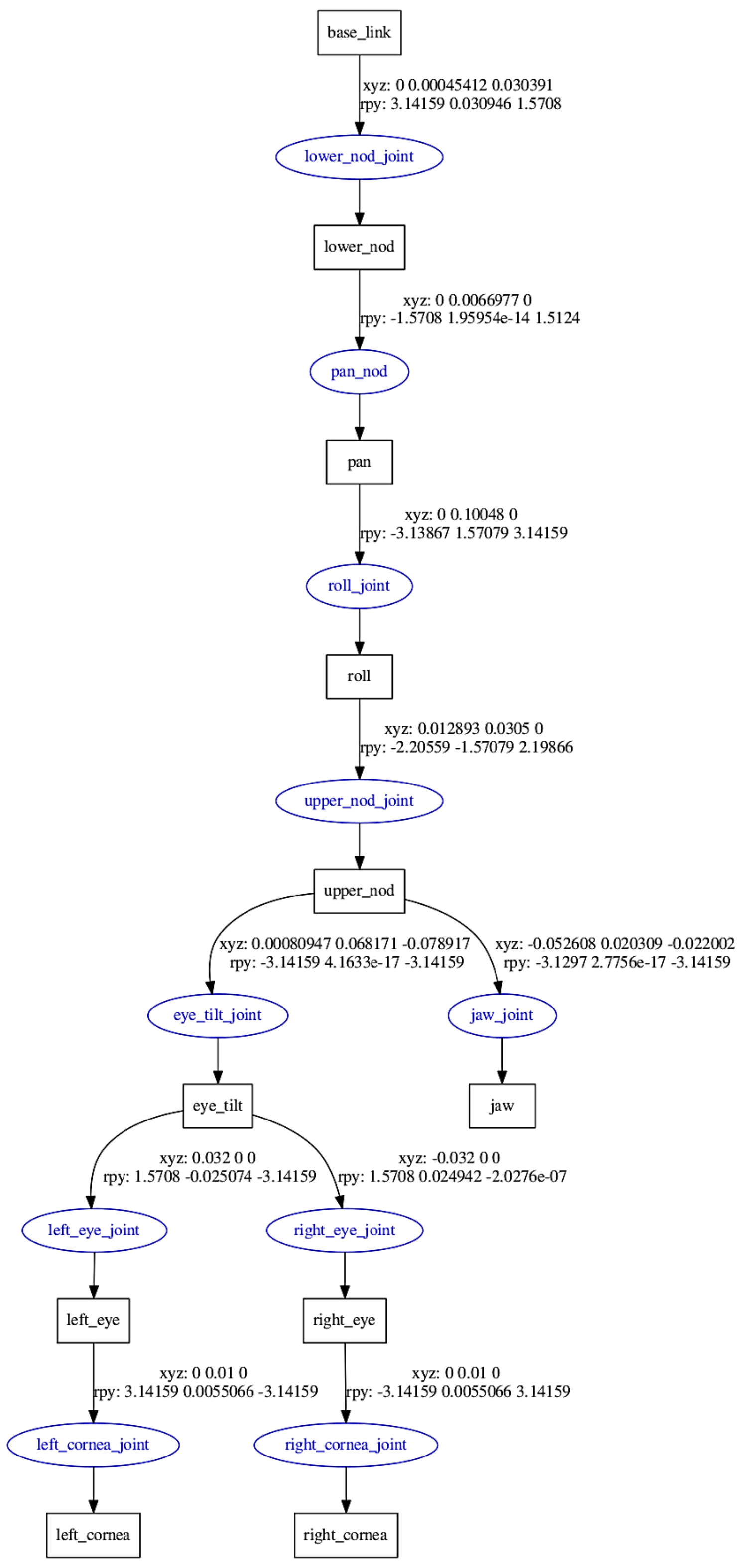

base_link
xyz: 0 0.00045412 0.030391
rpy: 3.14159 0.030946 1.5708
lower_nod_joint
lower_nod
xyz: 0 0.0066977 0
rpy: -1.5708 1.95954e-14 1.5124
pan_nod
pan
xyz: 0 0.10048 0
rpy: -3.13867 1.57079 3.14159
roll_joint
roll
xyz: 0.012893 0.0305 0
rpy: -2.20559 -1.57079 2.19866
upper_nod_joint
upper_nod
xyz: 0.00080947 0.068171 -0.078917
rpy: -3.14159 4.1633e-17 -3.14159
xyz: -0.052608 0.020309 -0.022002
rpy: -3.1297 2.7756e-17 -3.14159
eye_tilt_joint
jaw_joint
eye_tilt
jaw
xyz: 0.032 0 0
rpy: 1.5708 -0.025074 -3.14159
xyz: -0.032 0 0
rpy: 1.5708 0.024942 -2.0276e-07
left_eye_joint
right_eye_joint
left_eye
right_eye
xyz: 0 0.01 0
rpy: 3.14159 0.0055066 -3.14159
xyz: 0 0.01 0
rpy: -3.14159 0.0055066 3.14159
left_cornea_joint
right_cornea_joint
left_cornea
right_cornea

Figure A - 1URDF Visualization of PKD

References

[1] M. Mori, K. F. MacDorman and N. Kageki, "The uncanny valley [from the field]," *Robotics \& Automation Magazine, IEEE,* vol. 19, no. 2, pp. 98-100, 2012.

[2] D. Hanson, A. Olney, S. Prilliman, E. Mathews, M. Zielke, D. Hammons, R. Fernandez and H. Stephanou, "Upending the uncanny valley," in *Proceedings of the national conference on artificial intelligence*, 2005.

[3] R. B. Burns, "Verbal and Non-verbal Communication," *Essential Psychology: For Students and Professionals in the Health and Social Services,* pp. 223-235, 1991.

[4] K. F. MacDorman, "Subjective ratings of robot video clips for human likeness, familiarity, and eeriness: An exploration of the uncanny valley," in *ICCS/CogSci-2006 long symposium: Toward social mechanisms of android science*, 2006.

[5] H. McGurk and J. MacDonald, "Hearing lips and seeing voices," 1976.

[6] S. Sampathkumar, "Real Time Motion Control For Natural Human Robot Interaction," 2014.

[7] A. Habib, S. K. Das, I.-C. Bogdan, D. Hanson and D. O. Popa, "Learning human-like facial expressions for Android Phillip K. Dick," in *Automation Science and Engineering (CASE), 2014 IEEE International Conference on*, 2014.

[8] T. Minato, M. Shimada, H. Ishiguro and S. Itakura, "Development of an android robot for studying human-robot interaction," in *Innovations in applied artificial intelligence*, Springer, 2004, pp. 424-434.

[9] D. Sakamoto, T. Kanda, T. Ono, H. Ishiguro and N. Hagita, "Android as a telecommunication medium with a human-like presence," in *Human-Robot Interaction (HRI), 2007 2nd ACM/IEEE International Conference on*, 2007.

[10] I. Straub, S. Nishio and H. Ishiguro, "Incorporated identity in interaction with a teleoperated android robot: A case study," in *RO-MAN, 2010 IEEE*, 2010.

[11] T. Hashimoto, N. Kato and H. Kobayashi, "Study on educational application of android robot SAYA: field trial and evaluation at elementary school," in *Intelligent Robotics and Applications*, Springer, 2010, pp. 505-516.

[12] J. Seyama and R. S. Nagayama, "The uncanny valley: Effect of realism on the impression of artificial human faces," *Presence: Teleoperators and Virtual Environments,* vol. 16, no. 4, pp. 337-351, 2007.

[13] F. Hegel, T. Spexard, B. Wrede, G. Horstmann and T. Vogt, "Playing a different imitation game: Interaction with an Empathic Android Robot," in *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, 2006.

[14] J.-H. Oh, D. Hanson, W.-S. Kim, I. Y. Han, J.-Y. Kim and I.-W. Park, "Design of android type humanoid robot Albert HUBO," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, 2006.

[15] I. Ranatunga, J. Rajruangrabin, D. O. Popa and F. Makedon, "Enhanced therapeutic interactivity using social robot Zeno," in *Proceedings of the 4th International Conference on PErvasive Technologies Related to Assistive Environments*, 2011.

[16] C. Becker-Asano and H. Ishiguro, "Evaluating facial displays of emotion for the android robot Geminoid F," in *Affective Computational Intelligence (WACI), 2011 IEEE Workshop on*, 2011.

[17] K. Berns and J. Hirth, "Control of facial expressions of the humanoid robot head ROMAN," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, 2006.

[18] N. Endo, S. Momoki, M. Zecca, M. Saito, Y. Mizoguchi, K. Itoh and A. Takanishi, "Development of whole-body emotion expression humanoid robot," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, 2008.

[19] H. Miwa, K. Itoh, M. Matsumoto, M. Zecca, H. Takanobu, S. Rocella, M. C. Carrozza, P. Dario and A. Takanishi, "Effective emotional expressions with expression humanoid robot we-4rii: integration of humanoid robot hand rch-1," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, 2004.

[20] F. Wilbers, C. Ishi and H. Ishiguro, "A blendshape model for mapping facial motions to an android," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, 2007.

[21] T. Wu, N. J. Butko, P. Ruvulo, M. S. Bartlett and J. R. Movellan, "Learning to make facial expressions," in *Development and Learning, 2009. ICDL 2009. IEEE 8th International Conference on*, 2009.

[22] K.-G. Oh, C.-Y. Jung, Y.-G. Lee and S.-J. Kim, "Real-time lip synchronization between text-to-speech (TTS) system and robot mouth," in *RO-MAN, 2010 IEEE*, 2010.

[23] G. C. Martin, "Preston blair phoneme series," *World wide web electronic publication,* 2006.

[24] B. K. Ghosh and I. B. Wijayasinghe, "Dynamics of Human Head and Eye Rotations Under Donders' Constraint," *Automatic Control, IEEE Transactions on,* vol. 57, no. 10, pp. 2478-2489, 2012.

[25] J. Rajruangrabin and D. O. Popa, "Robot head motion control with an emphasis on realism of neck--eye coordination during object tracking," *Journal of Intelligent \& Robotic Systems,* vol. 63, no. 2, pp. 163-190, 2011.

[26] H. Simonsz and I. Den Tonkelaar, 19th century mechanical models of eye movements, Donders' law, Listing's law and Helmholtz'direction circles, Springer, 1990.

[27] "ABOUT HANSON ROBOTICS," [Online]. Available: https://hansonrobotics.wordpress.com/about/.

[28] "Pololu Maestro Servo Controller User's Guide," [Online]. Available: https://www.pololu.com/docs/0J40/all.

[29] "Kinect for Windows Sensor Components and Specifications," [Online]. Available: https://msdn.microsoft.com/en-us/library/jj131033.aspx.

[30] "Kinect for Windows Sensor," [Online]. Available: https://msdn.microsoft.com/en-us/library/hh855355.aspx.

[31] "Face Rigging - Guido Salimbeni Personal_Wiki Website," [Online]. Available: https://sites.google.com/site/salimbeniwiki/home/3d/face-rigging.

[32] "RViz User's Guide," [Online]. Available: http://docs.ros.org/hydro/api/rviz/html/user_guide/.

[33] "Face Tracking," [Online]. Available: https://msdn.microsoft.com/en-us/library/jj130970.aspx.

[34] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," *Foundations of genetic algorithms,* vol. 1, pp. 69-93, 1991.

[35] H. Demuth and M. Beale, "Neural network toolbox for use with MATLAB," 1993.

[36] I.-H. Jung, "Natural 3D Lip-sync Animation Based on Korea Phonemic Data," *Journal of Contents Society,* vol. 9, no. 2, pp. 331-339, 2008.

[37] R. C. Luo, S.-R. Chang, C.-C. Huang and Y.-P. Yang, "Human robot interactions using speech synthesis and recognition with lip synchronization," in *IECON 2011-37th Annual Conference on IEEE Industrial Electronics Society*, 2011.

[38] W. Paulus, "Lips don’t lie," 2013. [Online]. Available: http://wolfpaulus.com/jounal/software/lipsynchronization/.

[39] "Application Programming Interface (API) Developer Guide," [Online]. Available: http://www.ispeech.org/api.

[40] I. Wijayasinghe, J. Ruths, U. B{\"u}ttner, B. K. Ghosh, S. Glasauer, O. Kremmyda and J.-S. Li, "Potential and optimal control of human head movement using Tait--Bryan parametrization," *Automatica,* vol. 50, no. 2, pp. 519-529, 2014.

[41] B. K. Ghosh, I. B. Wijayasinghe and S. D. Kahagalage, "A geometric approach to head/eye control," 2014.

[42] "Human Faces Might Only Express Four Basic Emotions," [Online]. Available: http://www.smithsonianmag.com/smart-news/human-faces-might-only-express-four-basic-emotions-180949598/.

[43] Z. Zhang, "Microsoft kinect sensor and its effect," *MultiMedia, IEEE,* vol. 19, no. 2, pp. 4-10, 2012.

[44] G. P. Zhang, "Neural networks for classification: a survey," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on,* vol. 30, no. 4, pp. 451-462, 2000.

[45] O. Yetkin, J. Sanford, F. Mirza, R. Karulkar, S. K. Das and D. O. Popa, "Control of a Powered Prosthetic Hand via a Tracked Glove," *Journal of Mechanical Design,* 2015.

[46] S. Woods, K. Dautenhahn and J. Schulz, "The design space of robots: Investigating children's views," in *Robot and Human Interactive Communication, 2004. ROMAN 2004. 13th IEEE International Workshop on*, 2004.

[47] B. Robins, K. Dautenhahn and J. Dubowski, "Does appearance matter in the interaction of children with autism with a humanoid robot?," *Interaction Studies,* vol. 7, no. 3, pp. 509-542, 2006.

[48] N. J. Rinehart, J. L. Bradshaw, A. V. Brereton and B. J. Tonge, "Movement preparation in high-functioning autism and Asperger disorder: a serial choice reaction time task involving motor reprogramming," *Journal of autism and developmental disorders,* vol. 31, no. 1, pp. 79-88, 2001.

[49] D. J. Ricks and M. B. Colton, "Trends and considerations in robot-assisted autism therapy," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 2010.

[50] I. Ranatunga, M. Beltran, N. A. Torres, N. Bugnariu, R. M. Patterson, C. Garver and D. O. Popa, "Human-robot upper body gesture imitation analysis for autism spectrum disorders," in *Social Robotics*, Springer, 2013, pp. 218-228.

[51] M. B. Moussa, Z. Kasap, N. Magnenat-Thalmann and D. Hanson, "MPEG-4 FAP animation applied to humanoid robot head," *Proceeding of Summer School Engage,* 2010.

[52] D. Matsui, T. Minato, K. F. MacDorman and H. Ishiguro, "Generating natural motion in an android by mapping human motion," in *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, 2005.

[53] E. Hjelm{\aa}s and B. K. Low, "Face detection: A survey," *Computer vision and image understanding,* vol. 83, no. 3, pp. 236-274, 2001.

[54] D. E. Golberg, "Genetic algorithms in search, optimization, and machine learning," *Addion wesley,* vol. 1989, 1989.

[55] R. Gockley, R. Simmons, J. Wang, D. Busquets, C. DiSalvo, K. Caffrey, S. Rosenthal, J. Mink, S. Thomas, W. Adams and others, "Grace and George: Social robots at AAAI," in *Proceedings of AAAI*, 2004.

[56] D. Feil-Seifer and M. J. Mataric, "Defining socially assistive robotics," in *Rehabilitation Robotics, 2005. ICORR 2005. 9th International Conference on*, 2005.

[57] G.-B. D. de Boulogne and R. A. Cuthbertson, The mechanism of human facial expression, Cambridge university press, 1990.

[58] K. Dautenhahn, C. L. Nehaniv, M. L. Walters, B. Robins, H. Kose-Bagci, N. A. Mirza and M. Blow, "KASPAR--a minimally expressive humanoid robot for human--robot interaction research," *Applied Bionics and Biomechanics,* vol. 6, no. 3-4, pp. 369-397, 2009.

[59] C. Breazeal, A. Edsinger, P. Fitzpatrick and B. Scassellati, "Active vision for sociable robots," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on,* vol. 31, no. 5, pp. 443-453, 2001.

[60] S. Bouaziz and M. Pauly, "Dynamic 2d/3d registration for the kinect," in *ACM SIGGRAPH 2013 Courses*, 2013.

[61] E. T. Bekele, U. Lahiri, A. R. Swanson, J. A. Crittendon, Z. E. Warren and N. Sarkar, "A step towards developing adaptive robot-mediated intervention architecture (ARIA) for children with autism," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on,* vol. 21, no. 2, pp. 289-299, 2013.

[62] H. S. Ahn, D.-W. Lee, D. Choi, D. Y. Lee, M. H. Hur, H. Lee and W. H. Shon, "Development of an android for singing with facial expression," in *IECON 2011-37th Annual Conference on IEEE Industrial Electronics Society*, 2011.

[63] "M. Faire. (2014) Headline Maker: Hanson Robotics' Philip K. Dick Android. Maker Faire UK," [Online]. Available: http://www.makerfaireuk.com/2014/01/10/headline-maker-hanson-robotics-philip-k-dick-android/.

[64] "Open Cog," [Online]. Available: http://opencog.org/.

[65] "Phoneme," [Online]. Available: http://en.wikipedia.org/wiki/Phoneme.

Biographical Information

Sumit Kumar Das was born in Orissa, India. He got his Bachelors of Technology degree in Electronics and Telecommunication from the Biju Patanaik University of Technology, Orissa in 2010 and his Master of Science degree in Electrical Engineering from The University of Texas at Arlington in 2015. His research interests include Human-Robot interaction, social robotics and humanoid robotics.