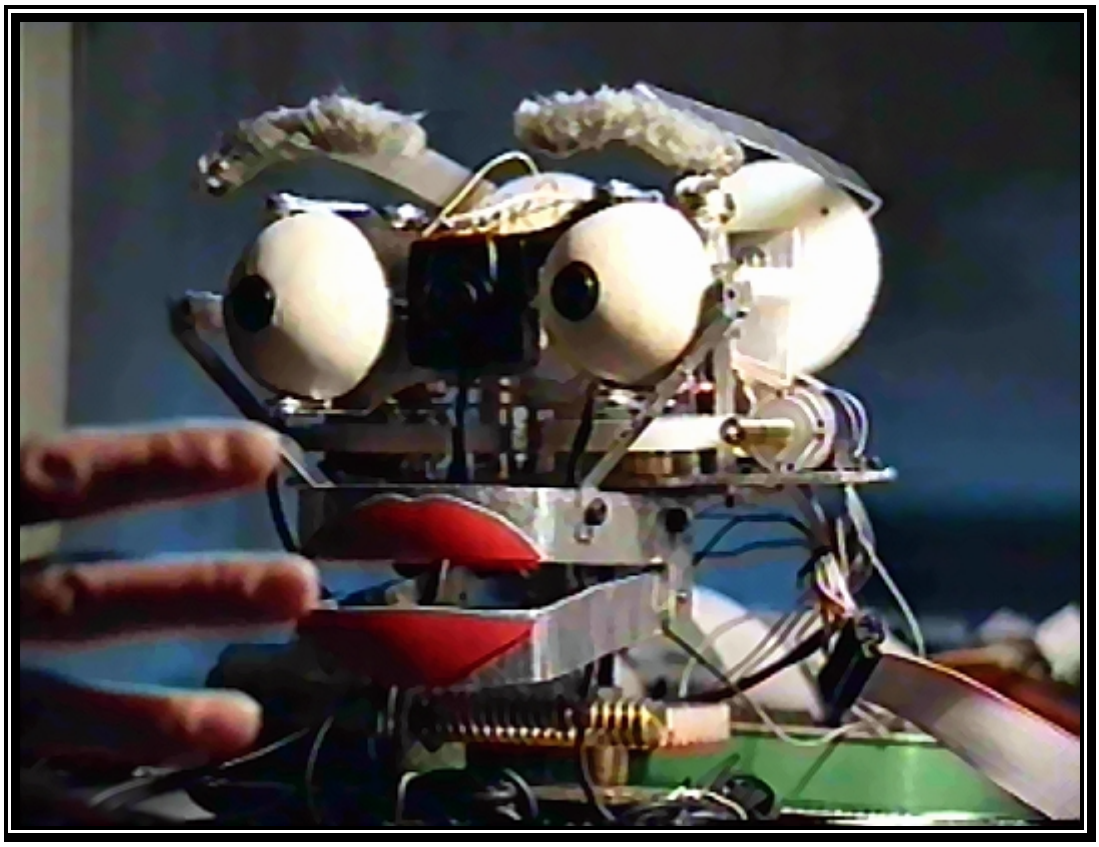
Aryan
Comes to life!



Azad University, Tehran-South Campus

Computer Hardware Engineering Group

Building an Interactive Robot Face from Scratch

By
Hossein Mobahi

Final Project Report

Submitted to the
Computer Hardware Engineering group

In partial fulfillment of the requirements for the degree of
Bachelor of Engineering in Computer Hardware Engineering

At

Azad University, South of Tehran campus
Tehran, Iran

May 2003

Author . . . . . .
Hossein Mobahi
May 2003

Supervisor . . . . . .
Massoud Katiraee (MSc.)
Project Supervisor

Distant Advisor . . . . . .
Cynthia Breazeal (PhD)
Director of Robotic Life Group, MIT Media Lab.

Jury . . . . . .
Abdol-Hossein Movahedi (PhD)
Head of Computer Hardware Engineering group

To My Dear Parents,

&

To My Lovely Mahsa

Acknowledgement

Although developing Aryan was my dissertation project, I had the advantage of having many people by my side without whom this wouldn't have been possible. Hereby, I would like to acknowledge each of their contributions in the process.

Professor Cynthia Breazeal is the first person I owe my deepest appreciation to. She had a major role to get this project up and running. When I first saw her work on her robot, Kismet, I asked her to help me and despite her busy schedule she enthusiastically agreed to kindly guide me, a complete stranger, through this difficult process. Although very detail oriented and objective, she kept me highly motivated by providing positive feedbacks as well as constructive advice.

My many thanks go to my fiancée, Mahsa Kamali, whose faith and moral support meant the world to me. She tolerated my long absences even on days when I didn't have time to call her on the phone. She believed in my project so deeply that she would overlook her own desire to be with me and instead made a priority of keeping me focused on my work. To top it all, she took a lot off my shoulders by preparing documents and photos as well as schematics, charts and diagrams. She has been my great inspiration throughout the way.

In making of the parts I needed for Aryan it was Kourosh Bayat who assisted me deliver a high quality with minimum expense. Behnam Ghiassedin took the time consuming tasks off my shoulders after I had designed the blueprints of control boards and amplifiers. I am grateful for what both these friends did. I also thank my cousin, Bita Shakoory, who accepted to edit parts of my report to minimize the grammatical mistakes. She was eager to do this boring part of the job to make the literature more sensible in English language. It is of importance to mention Ali Omumi, who gave me a great deal of support at the beginning. I also bothered my good friend, Ali Parvini and used his equipments such as printer, scanner and camera frequently.

I should also acknowledge my gratitude to Dr. Movahedi, chairman of the department of Computer Hardware Engineering at Azad University, Tehran-South Campus. During my years of undergraduate, he didn't hesitate to give me full support in coordinating and upholding seminars, workshops and competitions in addition to establishing Institute of Computer Engineering at the university. Accordingly in my dissertation project he and my supervisor, Mr. Katiraee kept faith in me and demonstrated maximum support, especially by extending the deadline when I needed extra time to finish up the job.

The last, but not the least are my parents who have especially contributed to this project in every way. Aryan would not have been born without their unconditional love and understanding. Their role is so prominent in my education that I cannot describe it with words. They made huge sacrifices so I could be a successful person and I hope someday I will be able to claim that I have made their wish come true. My ultimate appreciation goes to my father, mother and my two sisters, Haleh and Hedyeh, who provided me with all means of physical and mental comfort at home so I could concentrate on my job without any distractions.

I hope with this representation, I can make all the above mentioned individuals proud and fulfilled with their contributions.

Hossein Mobahi

Table of contents

List of Figures

List of Tables

Introduction

As robots take on an increasingly ubiquitous role in society, they must be easy for the average citizen to use and interact with. They must also appeal to persons of different age, gender, income, education and so forth. The goal is to design the human-robot interface such that untrained users can make safe and efficient use of the robot [Breazeal 99*]. It has been shown that education and entertainment are very promising applications for such interactive robots [Toschi 99, Doyle 99, Koda 96, Rizzo 99]. For instance, robotic tour guides have appeared in a few museums and are very popular with children [Burgard 98, Schulte 99].

Toward this goal, I thought it would be worthy and wonderful to start a long-term project for research and study about socially interactive robots. As the first step, an appropriate robotic platform was required for further explorations. Since such robots with the desired flexibility and sensory motor capabilities are not yet widely available, I had to develop my own robot from scratch. Therefore, I decided to choose this topic as my final project. I decided to work on my robot face, Aryan, due to the prominence of the face in social exchange. Previous work on software agents supports this by showing that people would find interaction with an agent that had a human face more appealing than an agent with no face [Koda 96, Takeuchi 95]. It is even indicated that people are more willing to cooperate with agents that have human faces [Keisler 97].

Rather than working on an animation-based synthetic face, I decided to work on a physical robot face to improve its believability. There are reasons supporting this idea. For instance, people expect that moving objects require intelligent control, while flat images likely result from the playback of a stored sequence as in film or television [King 96]. Moreover, a physical robot can be viewed from different angles, it can be touched and its approach toward people may be thought of as threat. Apparently animation does not have such strong impacts on people.

In addition to having a face, incorporating emotions in social robots can improve their operation for some reasons. First it helps the human-robot interaction to look more believable [Canamero 01, Ogata 00]. More ever, it provides feedback to the user, such as indicating the robot's internal state, goals and intentions [Bartneck 01, Breazeal 98, Kozima 01]. Lastly, it can act as a control mechanism, driving behavior and reflecting how the robot is affected by, and adapts to different factors over time [Canamero 97, Michaud 00, Velasquez 98].

Emotion expression in the face is an important biological aspect of emotion that has significant implications for how emotion is communicated in social situations [Darwin 1872]. So facial features are a key aspect for designing an interactive robot face. The mechatronical design and implementation of the face and facial features must serve a means for controlling multiple joints and features simultaneously and exhibit different facial expressions. Robot’s facial expression must be so easy to interpret that ordinary people easily recognize it.

The appearance of our robot must be such that it encourages humans to treat it as if it were a young socially aware creature. The face of the robot must reflect an amount of robotness so that the user does not develop detrimentally false expectations of the robot's capabilities [DiSalvo 02]. In fact, humans have strong implicit assumptions regards the nature of human-like interactions, and they are disturbed when interacting with a system that violates these assumptions [Cole 98].

Social robots must also possess perceptual abilities similar to humans. In particular, they need perception that is human-oriented and optimized for interacting with humans and on a human level. On the way to this goal, we should implement an anthropomorphic active vision system for our robot, capable of detecting and tracking human faces, facial features and hands. Acquired data must have high acuity for recognition tasks and for controlling precise visually guided motor movements, plus a wide field of view for search tasks, for tracking multiple objects coarsely, compensating for involuntary ego motion, etc [Breazeal 00*]. A simple brain must be developed for the robot so that it can display intelligent and emotional behaviors. Behaviors are realized by mapping visual stimulus and internal emotional state of the robot to its motor commands and facial expressions in a meaningful way. This can be achieved in a number of ways, e.g. heuristic rules, neural networks, state machines, etc.

The main objective of this project is to put all of the mentioned components together and show the possibility of developing such a complex robotic platform from scratch with very elementary and low-cost components with aid of basic tools. I think accomplishment of developing Aryan is itself the best metric for measuring success in the proposed objective, under constraints forced by instrument availability and financial issues. I had to construct most of essential building blocks of this project myself, because they were neither available in Iran[*], nor I could order them from abroad because there was no financial support for this project. And therefore all needed expenses have been paid from my personal budget.

For instance, all mechanical parts, even gears, are built at home from scratch. The electronic board is constructed from parts like resistors or transistors. The control board has been developed using widely available chips. And finally the whole software is based on my own code. Helplessly, I had to repeatedly reinvent the wheel during this project. Nevertheless, I wished this time could have been spent on other more important parts of the robot, such as its brain and intelligence. Due to the above mentioned reasons, learning, which is an important characteristic of sociable robots, is missing in this new born robot, or also more degrees of freedom are missing and are expected to be added in the future and in other parts of the long-term project.

I have put some pictures and video clips of this project on the web. Some of them show Aryan in action, tracking people and expressing emotions. Some show its perception from the world, demonstrating how it detects face or hand, superimposes a bounding box around it and recognizes it as being a hand or face. There are also pictures and clips focusing on physical implementation of Aryan, e.g. motors, gears and control boards. Interested readers can find these materials on www.digibrain.org.

---

[*] This is certainly not true in developed countries.

Chapter 1

The Face

Previous work on software agents shows that people would find interaction with agents that had a human face more appealing than an agent with no face [Koda 96, Takeuchi 95]. It is even indicated that people are more willing to cooperate with agents that have human faces [Keisler 97]. Therefore we decided to work on a robotic head. Two outstanding advantages of a physically implemented face over using a computer animation are [King96]:

- Expectation that moving objects require intelligent control, while flat moving images likely result from the playback of a stored sequence as in film or television.
- A three dimensional robot can be viewed from many angles, allowing people to see it without standing directly in front of the robot

Besides having a face, incorporating emotions in social robots can improve their operation for a number of reasons. First it helps the human-robot interaction to look more believable [Canamero 01, Ogata 00]. In addition, it provides feedback to the user, such as indicating the robot's internal state, goals and intentions [Bartneck 01, Breazeal 98, Kozima 01].

Lastly, it can act as a control mechanism, driving behavior and reflecting how the robot is affected by, and adapts to different factors over time [Canamero 97, Michaud 00, Velasquez 98].

Facial features are important and natural means for expressing emotional states. Moreover they serve several other functions, which have a significant impact in social communications such as protective behaviors (e.g. closing eyes), displaying expressive attitudes and communicative expressions (that supports language) [Breazeal 02]. Therefore, we considered facial features as a key objective of our face development.

Due to the constraints imposed by submission deadline, available components and having no financial support, our face is comprised of a few degrees of freedom so far (mouth, eyes and eyebrows). Yet, these features, particularly eyes and eyebrows, are the most essential that serve the mentioned functions. Other important facial features such as lips and eyelids can be added in the future if necessary.

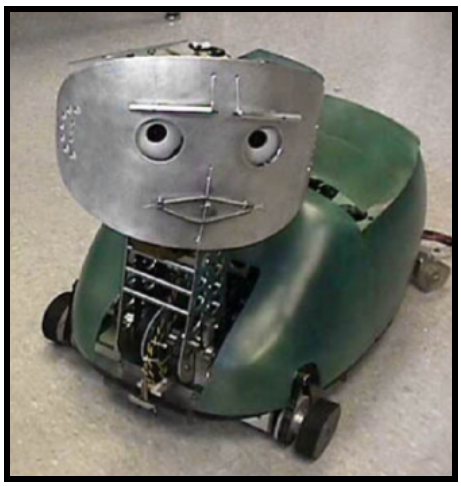


Figure 1.1 Sparky
by Scheef and colleagues

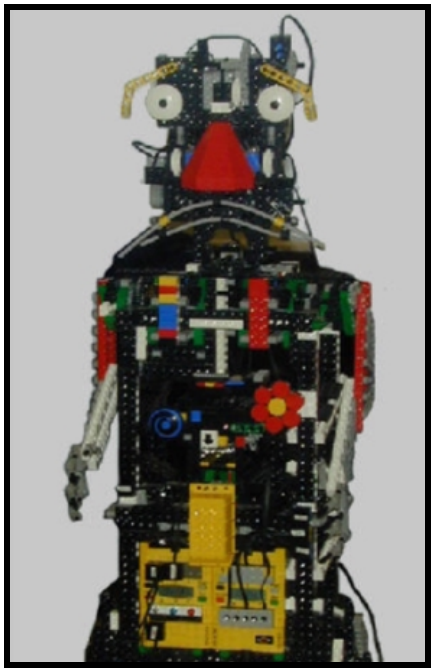


Figure 1.2 Feelix
by Canamero and Fredslund

1.1 Related Works

The use of emotions and their facial expressions in the context of human–robot interaction is receiving increasing attention. Here we give an overview of some of the most cited works in this field.

Sparky, displayed in Figure (1.1), is a robot developed by Scheeff and colleagues with the aim of exploring new ideas in human–computer interface and interactive robotics [Scheeff 00]. It uses facial expression, gesture, motion, and sound. The robot's only task is emotional expression in the context of social interaction. Unlike the other robots presented here, Sparky is not autonomous but teleoperated. Sparky's face has 4 DOF to control three expressive features (eyebrows, eyelids and lips)

Feelix, a robot built by Canamero and Fredslund [Canamero 00] is constructed from LEGO Mindstorms ™ as shown in Figure (1.2). It is a cheap and easy method to build robotic faces. However, the robot might be perceived only as a toy due its construction toy appearance and the prior experience of playing with LEGO that many people have. Feelix has been used as a research platform for human robotic interactions study. Its face has 4-DOF (two eyebrows, two lips), designed to display six facial expressions (anger, sadness, fear, happiness, surprise, neutral) plus a number of blends. It can only sense tactile stimulation.

eMuu, can be seen in Figure (1.3), is a robot developed by Bartneck to function as an interface between the user and intelligent home. The user can instruct the robot to perform a number of tasks of home [Bartneck 02]. eMuu is able to express three emotions happiness, sadness and anger with one lip and one eyebrow. eMuu's lip and eyebrow are flexible and able bend for displaying appropriate emotions. The emotion engine, which controls the emotional state and the facial expression, is based on the OCC model [Ortony 88].

Figure 1.3 eMuu developed by Bartneck

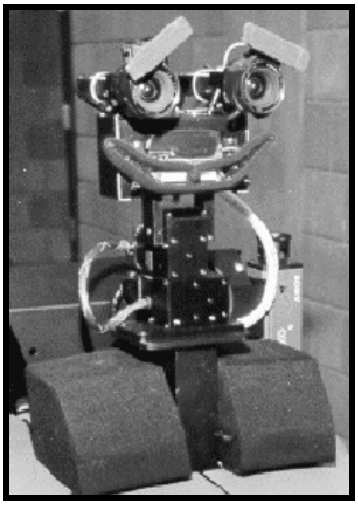


Figure 1.4 Minerva, built by Thrun *et.al*

Minerva [Schulte 99], which was developed by Thrun,*et al.* is an interactive tour-guide. Minerva can display four basic expressions neutral, happy, sad, and angry using a caricaturized face and simple speech, see Figure (1.4). Similar to Feelix, it has two expressive features and 4 DOF (one for eyebrow, two for mouth). Emotional states arise as a consequence of travel-related interaction, and their expressions aim at affecting this interaction toward achieving the robot's goals.

The Affective Tigger, is an expressive toy (Figure (1.5)), developed by [Kirsch 99] as a tool for the social and emotional awareness education of small (aged two to five) children. Affective Tigger's expressive facial features are a mouth (open or closed) and ears (pointing upwards or downwards) that allow it to express two emotions (happiness and sadness) plus a neutral face. Facial and vocal expressions reflect the emotional state of the toy as a response to the child's physical manipulation.

Fumio Hara has been developing realistic animated talking heads at the Hara Labs. He has incorporated hair, teeth, silicone skin and a large number of control points [Hara 98]. The biggest strength of his work is at the same time its biggest weakness. The high degree of realism of the heads raises high expectations in the conversational abilities of the heads that the current speech and dialogue technology cannot live up to. This problem is called "Uncanny Valley" which will be discussed in section 1.2.1.

Figure 1.5 Affective Tigger, by Kirsch

Figure 1.6 Realistic faces, by Fumio Hara

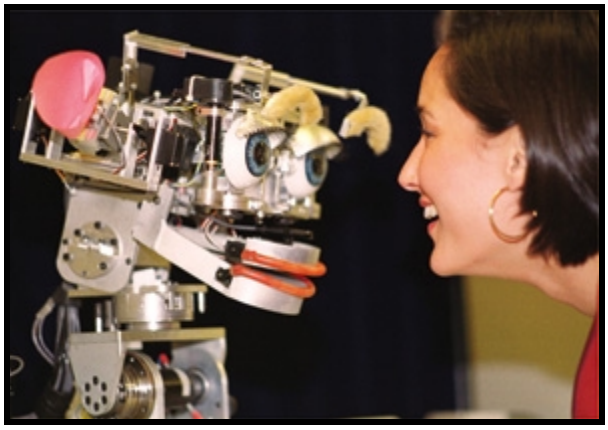


Figure 1.7 Kismet
Developed by Breazeal

Figure 1.8 Leonardo
Developed by Breazeal and
Stan Winston Studio

At the opposite end of the complexity scale is Kismet in Figure (1.7), which was developed by Breazeal [Breazeal 00] as a test bed for learning social interactions involving an infant (the robot) and her caretaker (a human). This robot is a head with active stereovision and configurable expressive features (controllable eyebrows, ears, eyeballs, eyelids, a mouth with two lips, and a neck that can pan and tilt) with 18 DOF. All these features, together with an expressive vocalization system, allow Kismet to display a wide variety of emotional expressions that can be mapped onto a three-dimensional space with dimensions arousal, valence, and stance.

At the time of writing this report, Leonardo shown in Figure (1.8), is the most expressive robot in the world. It's being developed by Breazeal in collaboration with Stan Winston Studio. Leonardo has an organic appearance. It has 61 degrees of freedom, 32 of those are in the face alone. As a result, Leonardo is capable of near-human facial expression but it is not designed to walk. It can gesture and is able to manipulate objects in simple ways. A small 16-channel motion control module is being designed for Leonardo. The motor drivers are standard FET H-bridges. The sixteen channels each support current feedback, encoder feedback, and analog feedback, and the system is controlled by a custom SoC motion controller with an embedded soft processor core implemented in a Xilinx Virtex FPGA. Breazeal and her students are now in the process of making this completely autonomous. Because Leonardo is under development, no published article is available about it at the moment. Interested readers may visit http://robotic.media.mit.edu.

1.2 Aesthetics and Physicality

The design task is to build a physical robot that encourages humans to treat it as if it were a young socially aware creature. Breazeal [Breazeal 02] suggest the following characteristics for design aesthetic:

- The robot should have an appealing appearance so that humans naturally fall into this mode of interaction. Physical appearance biases interaction.
- The robot must have a natural and intuitive interface (with respect to its inputs and outputs) so that a human can interact with it using natural communication channels. This enables the robot both read and send human-like social cues.
- The robot must have sufficient sensory, motor and computational resources for real-time performance during dynamic social interactions with people.

Due to the restrictions mentioned in the introduction chapter, particularly lack of essential building blocks for this project such as servomotors, we had to build most of them at home. Although we wouldn't have learned as much should these parts were readily available, the disadvantage was the larger size of the hand-made parts. So we had to choose a subset of facial features that were more representative in terms of displaying facial expressions.

An iconographic face consisting of two eyes with eyebrows and a mouth is almost universally recognizable, and can portray the range of simple emotions useful for interaction with humans. Therefore, we focused on these facial features for development. Currently, Aryan's mouth has rigid lips. To animate lips effectively, at least four independent motions were required that was impossible to be housed in our face because of the large size of our home-built servomotors.

1.2.1 The Uncanny Valley

Mashiro Mori developed a theory of The Uncanny Valley, which states that as a robot increases in humanness there is a point where the robot is not fully similar to humans but the balance between humanness and machine-like is uncomfortable. Mori provides and example: "If you shake an artificial hand [that you perceive to be real] you may not be able to help jumping up with a scream, having received a horrible, cold, spongy, grasp". According to Mori, there is a reasonable degree of familiarity that should be achieved and maintained in humanoid robots [Reichard 78].

According to Mori's Uncanny Valley, Aryan's face must reflect an amount of robotness. This is needed so that the user does not develop detrimentally false expectations of the robot's capabilities [DiSalvo 02]. In fact, humans have strong implicit assumptions regards the nature of human-like interactions, and they are disturbed when interacting with a system that violates these assumptions [Cole 98]. Therefore, unlike Hara's realistic robot faces [Hara 98], we did not put metal pieces in the face out of sight to mirror Aryan's robotness.

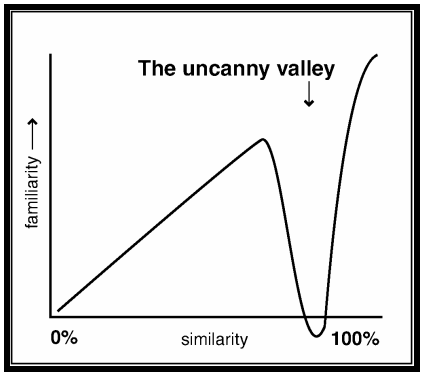


Figure 1.9 Mori's Uncanny Valley

However, the looking of Aryan must be improved for encouraging humans to interact with the robot. We augmented Aryan's facial features with colorful and suitable materials. Eyebrows were covered by a piece of dark and soft cotton. Too dark color was avoided to keep it visible even in low brightness and shady environments. Lips were chosen from a red plastic ribbon with an appropriate curvature. We also constructed anthropomorphic eyes for Aryan that will be discussed in the next section.

1.2.2 Anthropomorphic Eyes

Human visual perception plays an important role in his interaction, particularly for face-to-face interactions [Breazeal 00*]. So we employed multiple cameras to provide rich visual information from environment. We have used two different types of camera; first type is with a narrow field of view lens and the other one with a wide field of view lens. Although cameras with motorized zoom were also available, their cost exceeded our budget. More notably, they could not provide these two scales simultaneously. Therefore we preferred to use a separate camera for each scale.

The reason for this mixture of cameras is that typical visual tasks require both high acuity and a wide field of view. High acuity is needed for recognition tasks and for controlling precise visually guided motor movements. A wide field of view is needed for search tasks, for tracking multiple objects coarsely, compensating for involuntary ego motion, etc.

A common trade-off found in biological systems is to sample part of the visual field at a high enough resolution to support the first set of tasks, and to sample the rest of the field at an adequate level to support the second set. This is seen in animals with foveate vision, such as humans, where the density of photoreceptors is highest at the center and falls off dramatically towards the periphery [Breazeal 00*]. This is idea has also been used in other robots such as Cog [Scassellati98] and Kismet [Breazeal 00*].

The vision system of our robot consists of three color CMOS cameras manufactured by Proline ™ Corporation. CMOS cameras are chosen because they are less expensive than CCD cameras. Two of the cameras (Model 2155) are foveal with board lenses and a focal length of 7.8 mm and with a filed of view about 56 degrees. The other one is a wide field of view camera (Model 2154). The cameras cost 62,500 toomans or equivalently 78$.

The wide field of view camera is mounted between two eyeballs, exactly in the middle. It is used to provide a coarse image and direct the robot's attention toward people. The foveal cameras are mounted within the pupils of the eyes as shown in Figure (1.10). These are used for higher resolution post-attentional processing such as tracking human's facial features. Humans attribute a communicative value to eye movements, for instance, they use gaze direction to infer whether a person is attending to them, to an object of shared interest, or neither [Breazeal 00*].

Therefore, we surrounded Aryan's pupils (camera lenses) by bright eyeballs to ease seeing its eye movements. In order to leave room for small cameras to fit into the eyeballs, first a wooden model of the cameras was carefully constructed and a ball was split into two halves. We made a hole in the half ball so that the camera lens could pass through it and at the same time, the camera be enclosed securely within the eyeball.

Then these models were fitted into the eyeballs and using cement-like glue the required space was created for them to fit. This resulted in a structure resembling human's eye, while keeping intact the characteristics of a robot with its metal display. As an outcome we had a milky color hard frame that one could fit a small camera in, such that only the camera lens would be visible from outside. Because of the darkness of the camera lens, this resembled eye's pupil giving it a more natural look. The final appearance of the eyes can be seen in Figure (1.10).

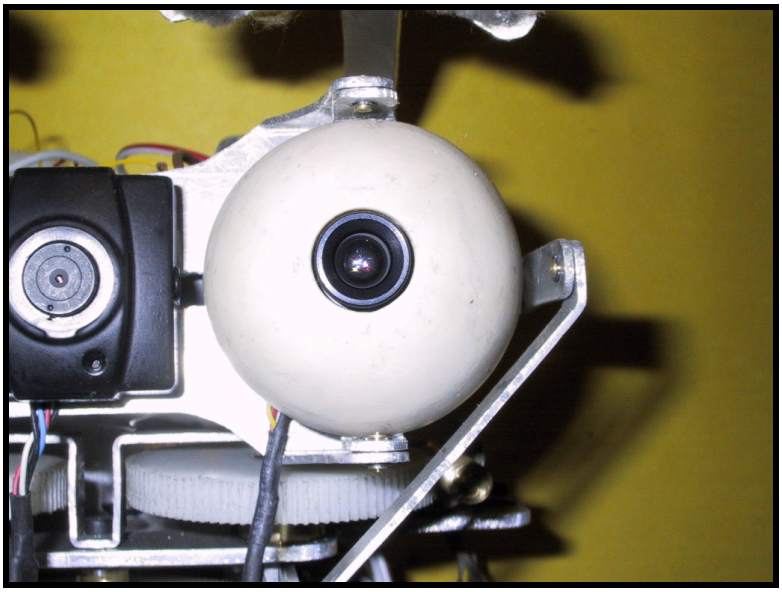


Figure 1.10 Aryan's Anthropomorphic Eyes

1.3 Mechanical Design

In our head design (facial features and neck), there are eight independent motions, so called Degrees Of Freedom (DOFs). Each DOF is actuated by a servomotor, a type of motor designed for control purpose. Briefly, it is made of a gear train, a feedback element and a control circuit. In the next chapter (Chapter 2) we will show how to build servomotors from simple DC motors. Here we just need to know that each DOF is actuated by a rotational motion.

1.3.1 Neck

The neck is comprised of two DOFs for pan and tilt motions. Since the neck must tolerate the weight of the whole face, its motors were chosen different from the rest. These motors are of lower voltage and lower RPM to give a higher torque. In addition, thicker gears were used in the neck to safely transmit large forces. The inevitable drawback is an increase in the weight of the neck. The physical and schematic pictures of the neck are shown in Figure 1.11 and 1.12 respectively.

1.3.2 Jaw

There are two aluminum pieces for resembling two halves of the mouth. The upper one is fixed and rigidly attached to the rest of the face. The lower half however plays the role of jaw and has one DOF to open and close the mouth, shown in Figure (1.13). Force transmission is achieved by a rope and drum mechanism as shown in Figure (1.14). An outstanding characteristic of rope and drum mechanism is its smooth and backlash-free motion, which helps it to look life-like.

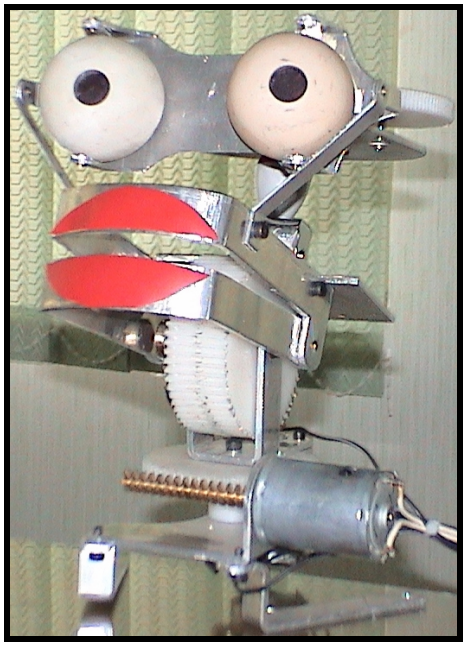


Figure 1.11 The physical neck

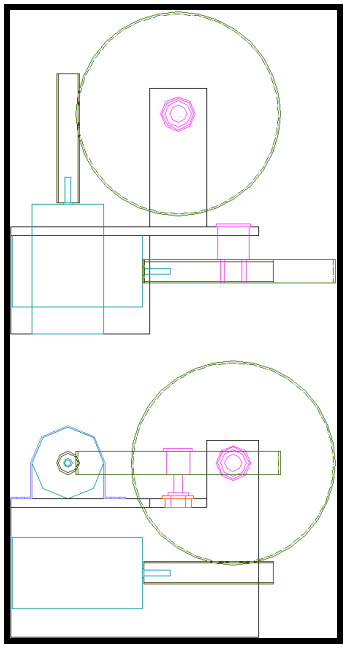


Figure 1.12 Schematic of the neck

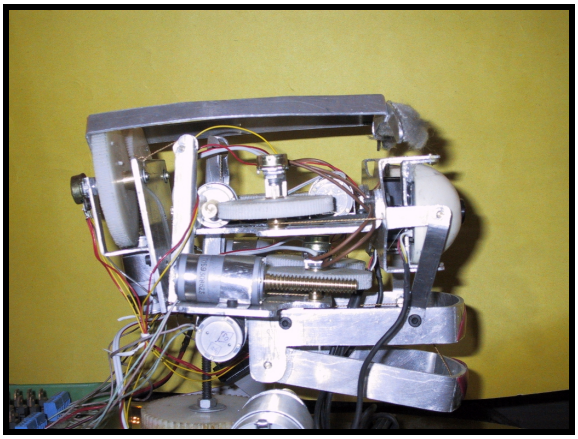


Figure 1.13 The physical jaw

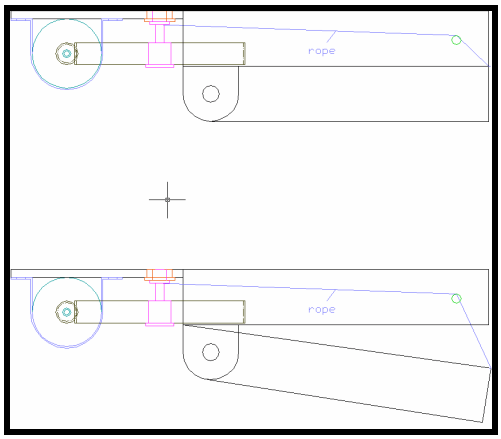


Figure 1.14 Schematic of the jaw

Such a mechanism can produce force in one direction only. Therefore, motor is able to either close or open the mouth and another force must be provided to move it in the opposite direction. We used the motor's force for closing the mouth, simply by turning the drum and twisting the rope about it. This causes the rope to be taken up. Since this is a vertical motion, the opposite force is provided naturally. In fact, we involved gravity for opening the mouth. However, due to the self-locking feature of a worm gear[*], the motor must turn back to release the rope. Releasing the rope in conjunction with gravity force opens the mouth.

Since we used potentiometer as feedback element of servomotors, the rotational range of motors is restricted, in our case to about 270 degrees. This could constrain the desired displacement range for the jaw. We solved this problem by appropriately choosing the radius of the drum.

1.3.3 Eyes

As mentioned earlier, three cameras are placed in the face. The eyes have three DOFs, two independent pans and shared tilt. The vise of each eye is constructed from aluminum pieces and hinged at the center of eyeball. The energy of motor is transmitted to this set using drum and rope mechanism. Top views of this design and its implementation are shown in Figure (1.15) and (1.16) respectively.

Since tilt motion occurs vertically, an approach similar to the one used for jaw is employed here. Although pan mechanism is same for both eyes, it differs from the mechanism used for tilt. It is so, because this time there is a horizontal motion of the eyes and gravity has no effect on it. So the opposite force should be provided in a different way. We connected one side (left or right does not matter) of each eyeball to the motor by an ordinary rope-drum mechanism and the other side was connected to chassis of the face using a polymer rope.

---

[*] We will discuss more about this in the next chapter, section 2.2

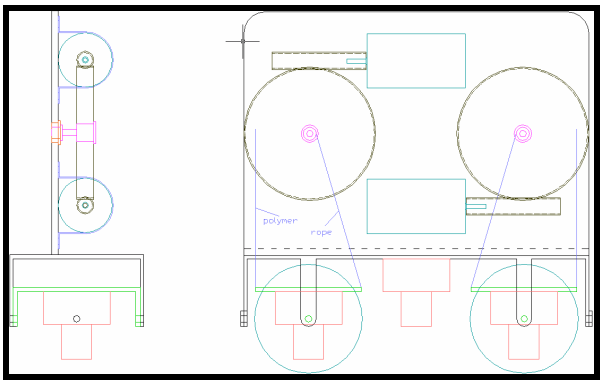


Figure 1.15 Schematic of eye mechanism

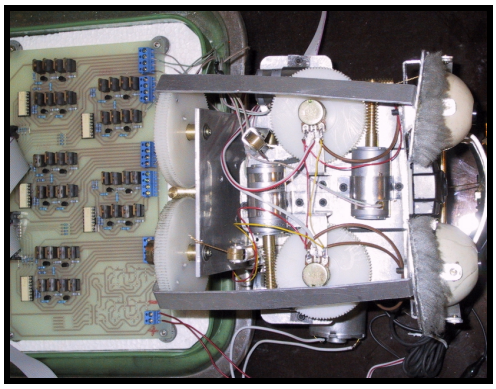


Figure 1.16 Physical eye mechanism

The elasticity property of the polymer causes storing a part of the energy coming from motor. When the energy of motor disappears (in fact, when the motor rotates in the opposite direction, because it is self-locking), the polymer rope releases its potential energy in the opposite direction. To prevent from translation in the eyes position, rotation axes must be chosen carefully. Tilt axis must connect two eyeball centers and pan axes must pass through eyeball centers and also be perpendicular to the tilt axis. Again since forces in opposite directions constantly pull the ropes, motion is backlash free.

1.3.4 Eyebrows

There are two DOFs for eyebrows, one for elevation and one for sweeping an arc. The former is shared symmetrically and the latter is shared identically between the eyebrows. To obtain elevation motion form rotational motion, we chose the rotation axis as far as possible from the location of the brows. This axis is located in the back of the head. Since only a small arc of displacement is required (with respect to the radius of rotation), the perceived motion looks linear.

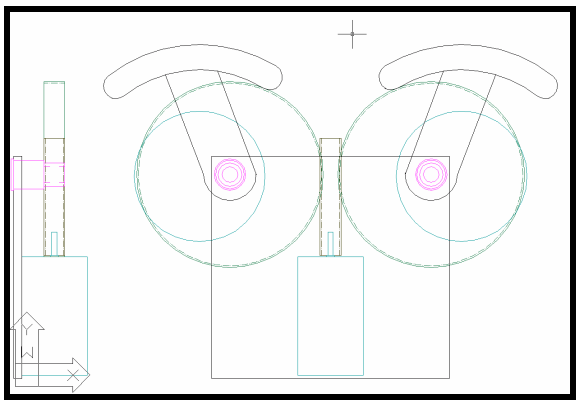


Figure 1.17 Schematic for eyebrows

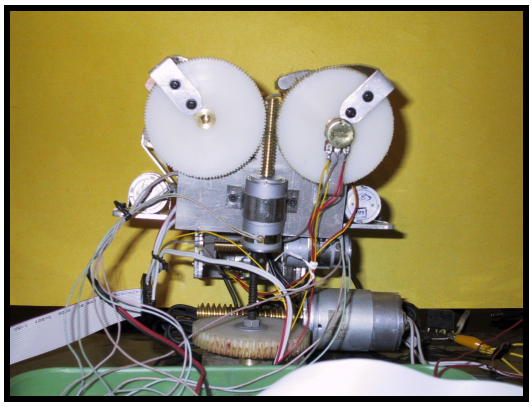


Figure 1.18 Physical eyebrows mechanism

Energy is transmitted from motor to eyebrows using a drum-rope mechanism. Since this motion is vertical, gravity can be easily used to supply the opposite force, as described earlier in 1.3.2. To sweep an arc with eyebrows, we used rotational motion about the center of the eyeballs. For obtaining two symmetrical motions, the spur gears of eyebrows enclose an actuating worm gear. A frontal view of the schematic and implemented mechanism are shown in Figures (1.17) and (1.18). The size of spur gears is the same so that they result in equal displacements for both eyebrows.

1.3.5 Summary of DOFs

Characteristics of all DOFs are summarized in table 1.1. As it can be seen, only two types of motors were used in this project. This constraint was imposed due to the very limited options that we had for buying motors in Iran. In Chapter 3 we will see that our control system can issue 256 different positioning commands. Therefore resolution of each DOF can be computed by dividing its swept angle by 256.

In any joint that only uses gears, potentiometer is coupled with the last gear to consider possible backlashes in feedback signal. However, in the DOFs that utilize rope and drum mechanism, potentiometers are coupled with the drums. However, drum and rope mechanism has no backlash because it is being pulled by two opposite forces all the time.

Table 1-1 Summary of Aryan's DOFs

DOF	Range (deg)	Resolution (deg)	Transmission	Voltage (v)	Power (w)	RPM
Neck Pan	-90 , +90	1.34	Gear (1:80)	6	10	1,800
Neck Tilt	-10 , +30	1.34	Gear (1:80)	6	10	1,800
Jaw	0 , -30	0.26	Gear (1:100) + Drum/Rope	12	5	2,800
Left Eye Pan	-15 , +15	0.19	Gear (1:100) + Drum/Rope	12	5	2,800
Right Eye Pan	-15 , +15	0.19	Gear (1:100) + Drum/Rope	12	5	2,800
Eyes Tilt	-50 , +10	0.33	Gear (1:100) + Drum/Rope	12	5	2,800
Eyebrows Elevation	-8 , +8	0.53	Gear (1:100) + Drum/Rope	12	5	2,800
Eyebrows Arcing	-15 , +15	0.50	Gear (1:100)	12	5	2,800

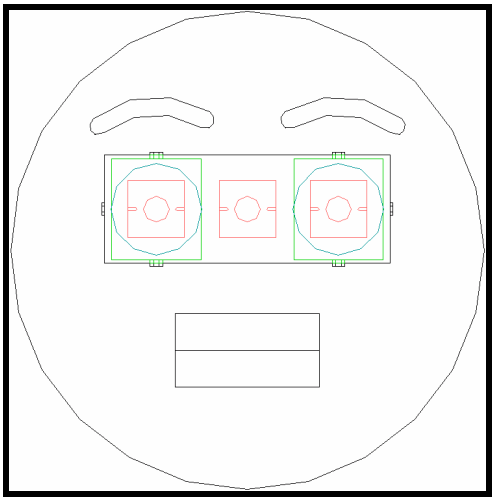


Figure 1.19 Schematic of Aryan's face

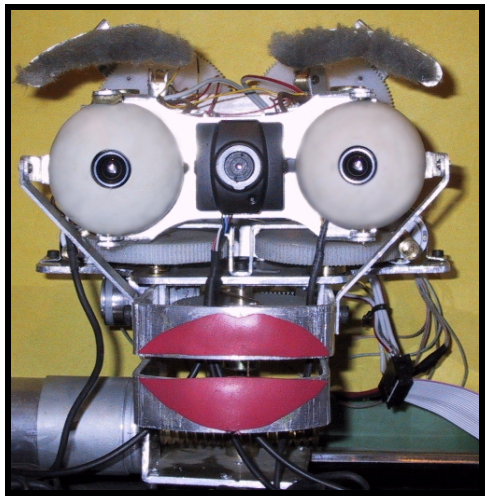


Figure 1.20 Physical appearance of Aryan's face

Frontal views of the face, schematic and real, are shown in Figures (1.19) and (1.20).

1.4 Facial Expressions

Ekman and Friesen [Ekman 82] developed a commonly used facial measurement system called FACS. The system measures the face itself as opposed to trying to infer the underlying emotion given a particular facial configuration. This is a comprehensive system that distinguishes all possible visually distinguishable facial movements. Every such facial movement is the result of muscle action. Based on a deep understanding of how much muscle contraction changes visible appearance, it is possible to decompose any facial movement into anatomically minimal action units. FACS has defined 33 distinct action units (AUs) for the human face.

Using the FACS system, Smith and Scott [Smith 97] have compiled mappings of FACS action units to the expressions corresponding to anger, fear, happiness, surprise, disgust, and sadness based on the observations of different researchers. Table 1.2 (taken from [Breazeal 02]) associates an action unit with an expression if two or more of these sources agreed on the association.

Table 1-2 Emotion Expressed vs. Facial Actions

	Happiness	Surprise	Anger	Disgust	Fear	Sadness
Eyebrow Frown			X	X	X	X
Raise Eyebrows		X			X	X
Raise Upper Eyelid		X	X		X	
Raise Lower Eyelid	X		X	X		
Up Turn Lip Corners	X					
Down Turn Lip Corners						X
Open Mouth	X	X			X	
Raise Upper Lip				X		

Each of Aryan's emotional states has an associated distinctive prototypical facial expression. Due to the physical constraints imposed by Aryan's face, e.g. lacking eyelids and flexible lips, we had to choose a subset of basic emotions (we will discuss more about this term in Chapter 5) that could be best represented by its actual face.

Looking at Tables 1.1 and 1.2 and considering that lips posture has a bigger impact than eyelids on emotion recognition, one can conclude that Aryan cannot express emotions happiness, sadness and disgust due to the dependency on lip postures. On the contrary, it can easily express emotions anger, fear and sadness plus a neutral state. These four expressions are shown in Figure (1.21).

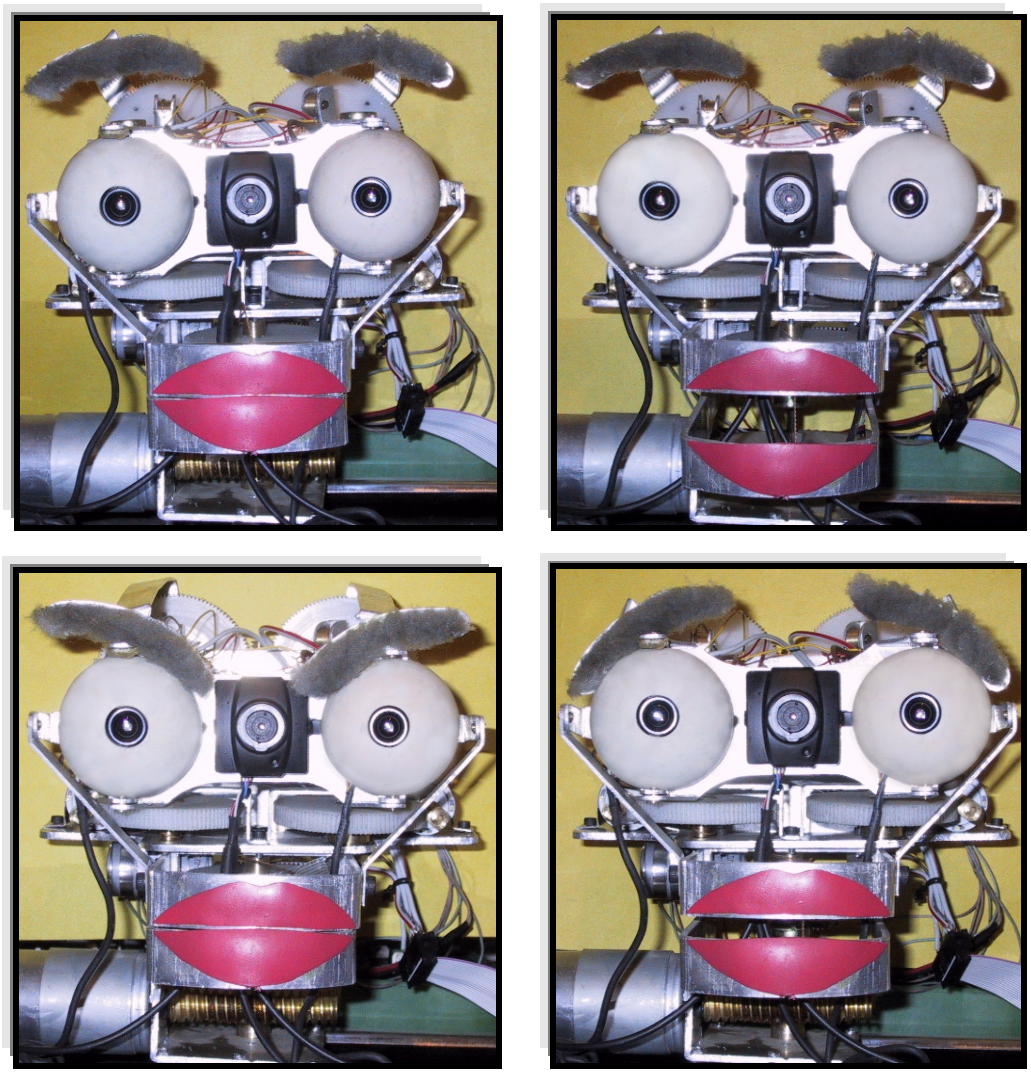


Figure 1.21 Aryan's Facial Expressions
From Left to Right and Top to Bottom : Neutral, Surprise, Anger, Fear

1.4.1 Expression Recognizability

We investigated the recognizability of Aryan's facial expressions. Emotion recognition tests were based on subjects' judgments of emotions expressed by Aryan, in a way similar to Breazeal's [Breazeal 02]. I devised a multiple-choice questionnaire where subjects were asked to label emotional expressions from pictures of Aryan expressing anger, surprise, and fear or being neutral.

21 subjects filled out the questionnaire, whose ages ranged from 12 to 64 years old. Most of subjects were students with 23 years of age. 10 of subjects were female and 11 were male. Most f the subjects had only heard about robotic but did not have any technical knowledge about it. Moreover, they did not know anything about robots with emotions.

There were four pages in the questionnaire. Each page had a large color image of Aryan displaying one of four expressions (neutral, anger, fear and surprise). The subjects could choose the best match from seven possible labels (basic emotions anger, fear, surprise, sadness, disgust and happiness plus a neutral state). In a follow-up question, they were asked to specify on a four-point scale how confident they were of their answer.

In the first evaluation, we considered answers to be correct when the subjects used the same descriptors we have employed, regardless of the confidence degree of their answers. The normalized statistics of the answers is shown in Figure (2.22). In another evaluation, we used a weighted sum of answers where the weight of each answer was its confidence degree.

The normalized statistics of weighted answers are shown in Figure (2.23). It is obvious that probability of a correct recognition is the highest for all four expressions.

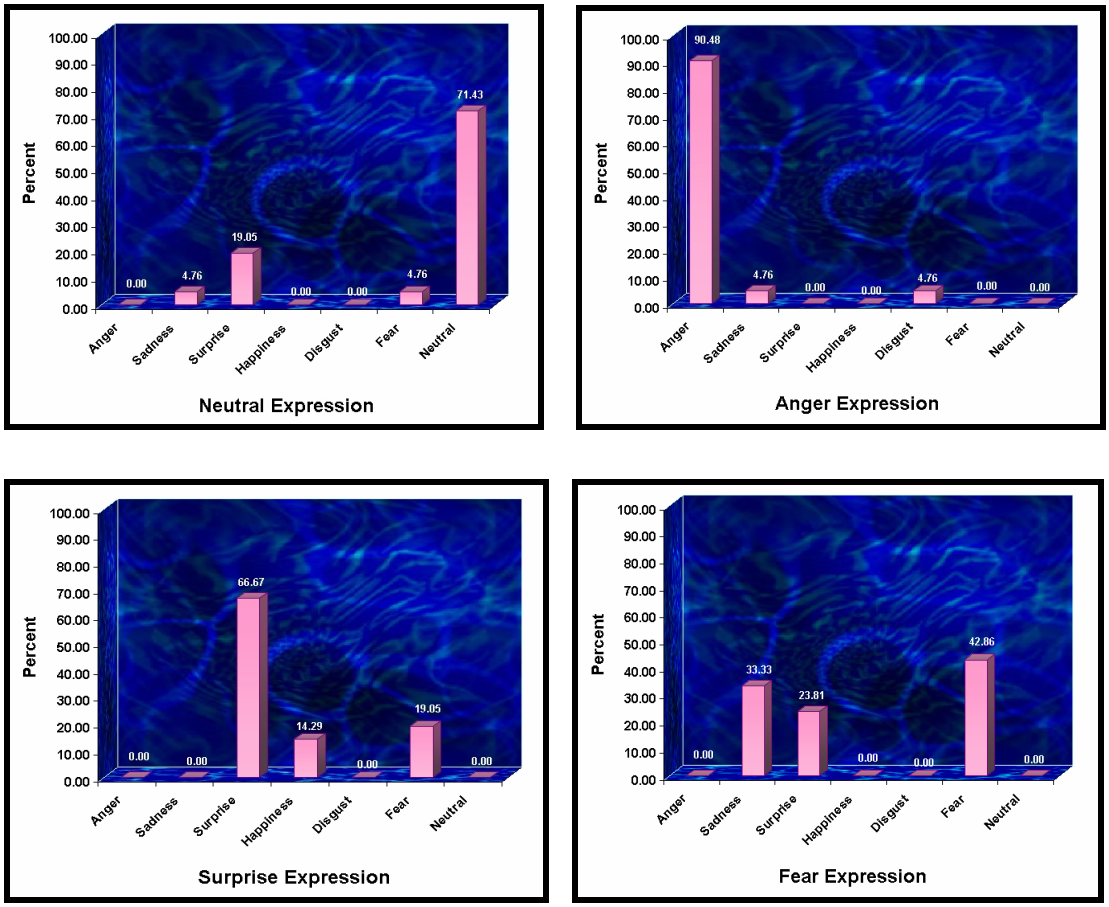


Figure 1.22 Recognizability of Aryan's facial expressions
Without considering confidence values

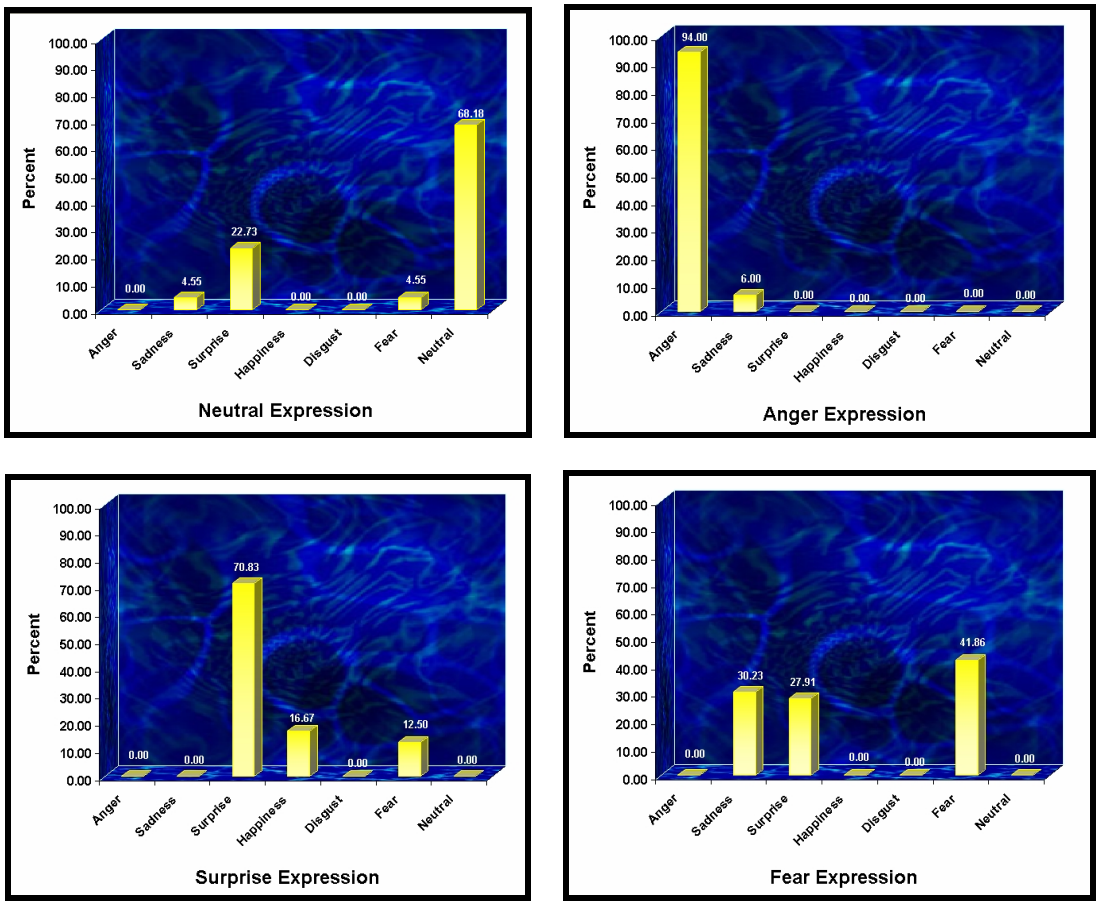


Figure 1.23 Recognizability of Aryan's facial expressions
Based on weighted answers

Other researchers have carried out similar experiments with their robots. For instance, Breazeal has evaluated recognizability of Kismet using both color images and video clips [Breazeal 02]. We took her image-based results to obtain more comparable values against ours. Canamero has performed analogous experiment on Feelix [Canamero 01]. Since she considered different statistics for children and adults, we used their average values. Table 1.3 shows correct recognitions for these robots.

Table 1-3 Expression Recognizability of different robot faces

	Aryan	Feelix	Kismet
Anger	94%	40%	76%
Disgust	---	---	71%
Fear	41%	16%	47%
Happiness	---	60%	82%
Sadness	---	70%	82%
Surprise	71%	37%	82%
Neutral	68%	---	---

One can say that comparing these three results is not fair because there were more for example choices in Kismet's questionnaire than Aryan's. In other words, if Aryan's questionnaire had such choices, perhaps some subjects would use them and this could possibly decrease its correct answers. Although it is true to some extent, but even with these noisy data it is not hard to see that recognizability of Aryan's expressions is something between two other robots.

Chapter 2

Servo Motors

Aryan utilizes a number of joints and links in its face to adjust its gaze direction and move its facial features. The actuator behind each of these joints is a simple DC motor. Depending on Aryan's perceptual data and emotional state, its brain issues appropriate motor commands for displaying facial expressions physically. This helps the robot to influence people emotionally and in a natural way.

However, controlling robot's joints is more complicated than merely issuing simple positioning commands. If the actuators were linear and static, the environment was noise-free and disturbance did not exist, it would be ok. But a DC motor is a non-linear dynamical actuator subjected to noise and disturbance

Therefore, a controller must be able to generate appropriate motor signals according to simple delivered motor commands. Usual controllers are based on a simplified model of the actual system. This is either due to our incomplete knowledge about the underlying mechanism and environment (e.g. noise) or accepting less accurate control to achieve a simpler design task.

Regardless of the source of imprecision, it causes an error between the actual output and the desired output of the controller. However, measuring this error is usually less costly than obtaining an accurate model of the system or environment. Controllers that measure the actual output of the system and feed it back to the controller are called Closed-Loop or Feedback Controllers.

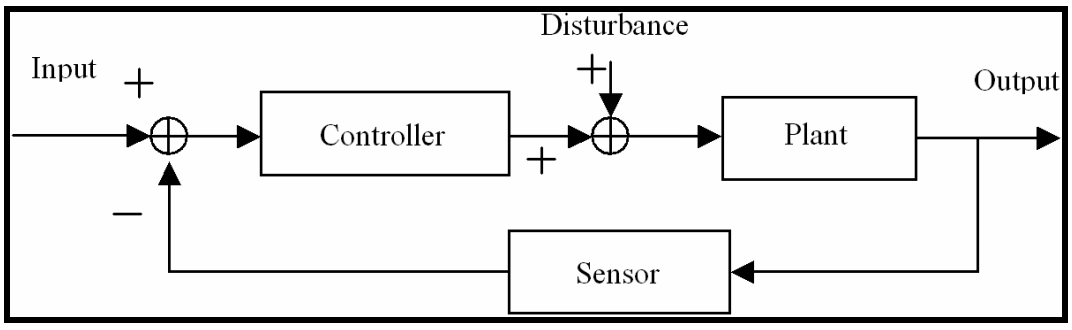


Figure 2.1 Building blocks of a servo mechanism

Figure (2.1) shows the traditional building blocks of a controller for a positioning task. Such a system is sometimes called a servo system. If the plant is replaced by a motor, the system is then called a servo motor.

In this chapter I will explain how to build a servomotor from elementary components. I will discuss about each block individually. Sophisticated blocks such as controller and amplifier will be discussed in separate chapters. In this chapter, however, we will have a brief glance at them.

2.1 DC Motors

When a wire carrying a current is placed within a magnetic field, it experiences a force normal to the plane formed by the magnetic field and the current as $F = I \times B$. If the wire is attached to a center of rotation, the resulting torque will cause it to rotate about the center of rotation. This is the basic idea of an electric motor [Sen 96].

There are many types of electric motors that are used in robotic. They include DC motors, reversible AC motors, brushless DC motors, hobby servomotors and stepper motors [Niku 01] The most adequate type of motor for building an expressive face is hobby servo motor. It consists of a DC Motor, gear train, feedback sensor and a built-in position controller, all in a compact package. Therefore, it can translate positioning commands into right signals and achieve position control with a relatively suitable torque.

Actuators in Kismet, perhaps the most complicated robotic face built so far are solely based on servo mechanism. Fourteen joints of kismet's facial features are actuated by Futaba Micro Motors that are lightweight hobby servos [Breazeal 02]. There are six other joints in its neck and eyes that are supposed to adjust its gaze direction. Since these joints need a higher precision, Maxon DC servomotors with encoder feedback are used. Although the latter set gives higher precision and torque with respect to hobby servos, they both share a common mechanism, called servo mechanism, which is the main concern of this chapter.

At the beginning of this project, Hobby Servo Motors were suggested by author's advisor, Prof. Breazeal. Unfortunately my choices for buying motors in the market of my country were too limited. From the list of motors commonly used in robotic that was mentioned earlier, I could find only DC motor and Stepper Motor in Iran. Therefore I had to choose one of these types that could best satisfy our application constraints.

2.1.1 DC Motors versus Stepper Motors

Briefly describing, a DC motor has an angular velocity proportional to the voltage across its input terminals. The direction of its rotation is determined by the polarity across input terminals.

Stepper motors get their input as a set of binary digits (on/off signals). A transition between states (by changing input bit pattern) results in a small rotation in a certain direction, which is called a step. The resolution of each step depends on the type of the motor. Common types step an angle of 0.9 degree or 1.8 degree.

By a quick study on both types, I could gather the following features helpful for our comparison purpose:

- Torque to Speed ratio is higher in DC Motors.
- Control of Steppers is much simpler than DC Motors, especially with digital controllers.

- Stepper Motors are able to lock motor's shaft whenever needed, but DC motors cannot.

- For the same torque to speed ratio, the size of a DC Motors is usually smaller than a Stepper Motor.

- Power consumption of Stepper Motors is higher than DC Motors for the same amount of useful work.

- Stepper Motors have a longer life than DC Motors, because they have no brushes to cause a friction and corrupt brushes over time.

- Stepper Motors can easily dissipate the generated heat by coils through the motor's body while in DC motors there is an air gap between the rotor and the motor's body, which acts as a heat isolator. Thus Stepper Motors are less susceptible to heat damage.

- A Stepper Motor does not necessarily need a feedback sensor as long as the torque imposed by the load is not greater than the generated torque by the motor. Therefore, given the initial position of the shaft, the angular position of the shaft after a number of steps is always known. Position control of a DC motor however requires a feedback sensor.

- Stepper Motors are more expensive than DC Motors.

For the following reasons I decided to use DC Motors:

- A DC Motor has a higher torque to speed ratio and also a smooth dynamic that yields a natural and life-like motion in the face, while motion in steppers looks very mechanical and unrealistic.

- Due to the high speed and low torque of DC Motors, they must be coupled with gear trains to increase torque. A gear train can be designed such that it prevents motions from the load side, and only allow motor side to move. Therefore, locking mechanism can be implemented through a gear train for a DC motor. An advantage of this approach is that when the motor is locked, it is off, while in steppers the lock state draws current.

- We must fit a large number of motors in a limited space of the face. So the motors must be small and lightweight. DC Motors easily satisfy this constraint.

- Sometimes a number of motors may be required to rotate simultaneously. Furthermore, at the beginning of motion, motors draw a considerable amount of current. Hence power consumption becomes an important factor and DC motors are preferred.

- It is true that Stepper Motors have a longer life than DC Motors do, because they have no brushes. But in our application, the motors do not rotate all the time, just a little after a while, and they rest again for some time. So there is no long-term friction during robot's operation and brushes will not get corrupted soon.

- Although Stepper Motors are less susceptible to heat damage, but for the same reason mentioned above, in our application DC Motors do not get so hot that heat dissipation becomes important.

- Generally, a Stepper Motor does not need a feedback sensor, but for high precision tasks even small errors are taken into the account. If for any reason the motor misses a step, over time these small errors are accumulated and produce a large error. Therefore in tasks like ours, a feedback sensor is needed for any type of motor.

- Since our face needs several motors, and my project did not have any financial support, cost was important and DC Motors were preferred.

The major drawback of a DC Motor is its sophisticated control. Indeed, there is usually a trade-off between control complexity and performance when options are restricted to these two types of motors. But due to the numerous advantages mentioned for a DC Motor in this project, I decided to build the controller myself too, but apply DC Motors in the face.

2.1.2 DC Motor Modeling

A DC motor basically works on the principle that a current carrying conductor in a magnetic field experiences a force $F = \Phi x i$, where Φ is the magnetic flux and i is the current in the conductor [Kuo 82]. The motor itself consists of a fixed stator and a movable rotor that rotates inside the stator. If the stator produces a radial magnetic field Φ and the current in the rotor (also called armature) is i then there will be a torque on the rotor causing it to rotate. The magnitude of this torque is:

$$\tau_m = K_1 \Phi I_a \tag{2.1}$$

While τ_m is the motor torque (N-m), Φ is the magnetic flux (Weber), I_a is the armature current (Amps), and K_1 is a physical constant. In addition, whenever a conductor moves in a magnetic field, a voltage V_b is generated across its terminals that is proportional to the velocity of the conductor in the field. This voltage, called the back emf, will tend to oppose the current flow in the conductor.

$$V_b = K_2 \Phi \omega_m \tag{2.2}$$

Where V_b denotes the back emf in Volts, ω_m is the angular velocity of the rotor (rad/sec) and K_2 is proportionality constant.

DC Motors can be classified according to the way in which the magnetic field is produced and the armature is designed. Here we discuss only the so-called permanent magnet motors whose stator consists of a permanent magnet. In this case, we can take the flux Φ to be constant. The torque on the rotor is then controlled by controlling the armature current I_a.

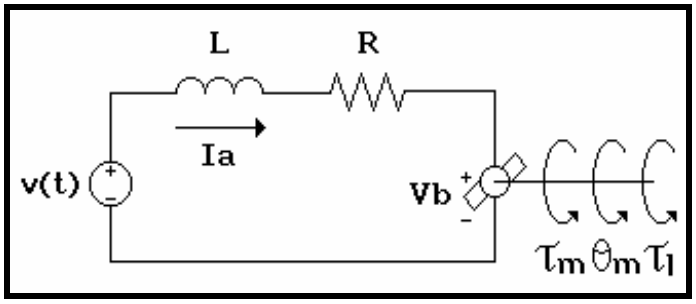


Figure 2.2 A simplified model of DC Motor

Table 2-1 DC Motor Parameters

$V(t)$	Armature Voltage
L	Armature Inductance
R	Armature Resistance
V_b	Back emf
I_a	Armature Current
Θ_m	Rotor Position (radians)
T_m	Generated Torque
T_l	Load Torque
Φ	Magnetic Flux due to the Stator

Consider Figure (2.2) and table 2.1. The differential equation of the armature current is then:

$$L\frac{dI_a}{dt} + RI_a = V - V_b \tag{2.3}$$

Since the flux Φ is constant the torque developed by the motor is

$$\tau_m = K_1 \Phi I_a = K_m I_a \tag{2.4}$$

Where K_m is the torque constant in N-m/amp. From (2.2) we have

$$V_b = K_2 \Phi \omega_m = K_b \omega_m = K_b \frac{d\theta_m}{dt} \tag{2.5}$$

Where K_b is the back emf constant. The remainder of the discussion refers to Figure (2.3) consisting of the DC Motor in series with a gear train with gear ration 1:r and connected to a link of the manipulator. We set $J_m = J_a + J_s$, the sum of the actuator and gear inertias. The equation of motion of this system is then

The remainder of the discussion refers to Figure (2.2) consisting of the DC Motor in series with a gear train with gear ration $n=1/r$ and connected to a link of the manipulator. We set $J_m = J_a + J_s$, the sum of the actuator and gear inertias. The equation of motion of this system is then

$$J_m \frac{d^2\theta_m}{dt^2} + B_m \frac{d\theta_m}{dt} = \tau_m - r\tau_l = K_m I_a - r\tau_l \tag{2.6}$$

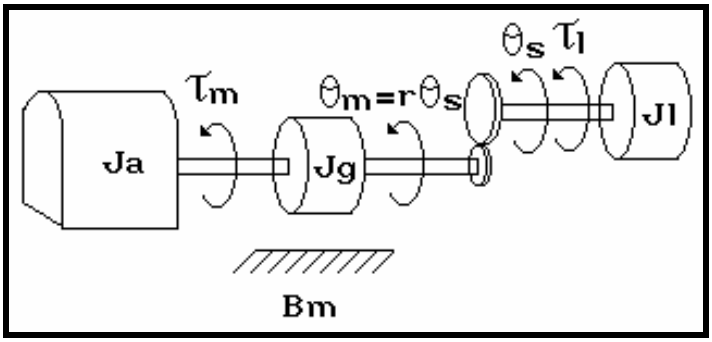


Figure 2.3 DC Motor, gear train and manipulator link

The latter equality comes from (2.4). In the Laplace domain the three equations (2.3), (2.5) and (2.6) may be combined and rewritten as

$$(Ls + R)I_a(s) = V(s) - K_b s\Theta_m(s) \tag{2.7}$$

$$(J_m s^2 + B_m s)\Theta_m(s) = K_m I_a(s) - r\tau_l(s) \tag{2.8}$$

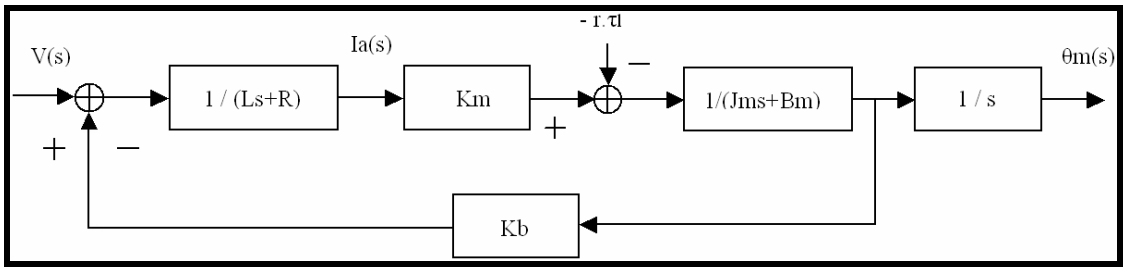


Figure 2.4 Servo system based on a DC Motor

The block diagram of the above system is shown in Figure (2.4). The transfer function from *V(s)* to $\Theta_m(s)$ with $\tau_l = 0$ is then given by:

$$\frac{\Theta_m(s)}{V(s)} = \frac{K_m}{s[(Ls + R)(J_m S + B_m) + K_b K_m]} \tag{2.9}$$

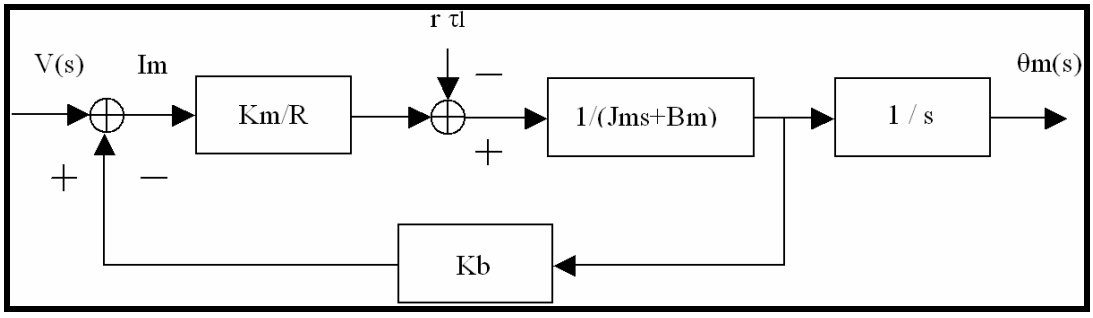


Figure 2.5 Servo system using a simplified model of DC Motor

Frequently it is assumed that the "electrical time constant" L/R is much smaller than the "mechanical time constant" J_m/B_m. This is a reasonable assumption for many electro-mechanical systems and leads to a reduced order model of the actuator dynamics.

Now if we divide numerator and denominator of (2.7) by R and neglect the electrical time constant by setting L/R equal to zero, the transfer function between $\Theta_m(s)$ and V becomes (again, with $\tau_l = 0$ for linearity).

$$\frac{\Theta_m(s)}{V(s)} = \frac{K_m / R}{s(J_m s + B_m + K_b K_m / R)} \tag{2.10}$$

The block diagram corresponding to the reduced order system (2.10) is shown in Figure (2.5).

2.2 Gears

Electric motors rotate at high speeds (up to many thousands of revolutions per minute) and generate very low torques (motor's shaft can be easily stopped by pressing it with a little force by two fingers). So they must be used in conjunction with reduction gears to increase their torque and to decrease their speed.

This, of course, increases the cost, number of parts, backlash, inertia of rotating body, etc., but also increases the resolution of the system, as it is possible to rotate the link very small angle [Niku 01].

Options in Iran's market for choosing among available geared DC motors were very limited. In addition, they were so heavy and large that we could not fit all of them in robot's face. At last, they were very expensive. So we not only had to build a servo mechanism, but also the gears! Fortunately an old gear-cutting machine was available in the cellar of the house∗ by which we built the gears.

We connected the shaft of each motor to a worm gear, which is itself connected to a spur gear for obtaining the desired gear ratio. A Spur gear is characterized by its teeth, which are perpendicular to the face of the gear. They are the most commonly available and least expensive gears to manufacture [Shigley 01]. A worm gear looks like a screw.

It is mounted horizontally, and as it turns, the "threads" of the worm gear appear to advance. As the "threads" advance, the vertically mounted spur gear is rotated [Shigley 01].

One advantage of a worm gear is that when it is rotated, the mating gear will move, but when the mating gear is attempting to turn, the worm gear will not, therefore, worm gear systems are self-locking. As we discussed in section 2.1.1, this characteristic is useful when the motor is idle. This feature allows the motor to turn off without worrying about the load to cause any unwanted change in the current state.

Another benefit of worm gears is very high ratios that they provide. This feature was very significant in our work because our gears were not manufactured with professional instruments, but were cut at home with an old machine. Therefore it was really hard to construct a gear train with several gears. We had to keep the number of gears as few as possible. So we adopted the simplest case that only two gears were used. High ratio of the worm gear helps to use only a few gears for each train.

The drawback of using a worm gear in a gear train is its high friction coefficient that makes it very difficult to turn by hand. This causes the power of the motor to be wasted on overcoming the resistance of the gears. We tried to reduce this friction by lubricating gears with grease oil.

∗ My friend Kourosh Bayat who helped me in building mechanical parts of the face was responsible for producing the gears

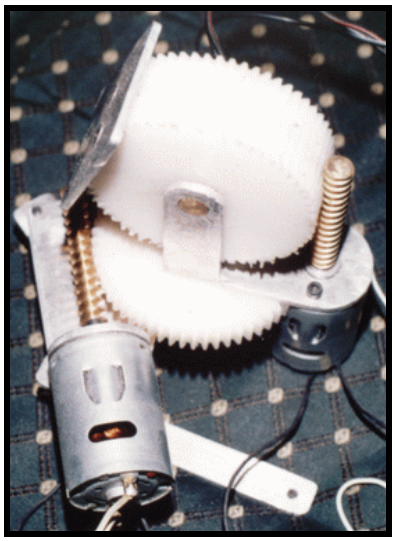


Figure 2.6 Combination of spur and worm gears in our motors

In Figure (2.6) pan/tilt neck of our robot is shown. As you probably saw in Table 1.1 of chapter one, only two sets of gear ratios were used in our project (1:80 and 1:100). It is so because building servomotors at home and from elementary tools prevented us from choosing any arbitrary ratio and consequently desired torque. We tried among a few gears that we could produce aiming to achieve a working system, not necessarily optimal.

2.3 Feedback Sensor

In a servo system, feedback sensor is used to report the actual output to the controller. In section 2.4 will see how this measurement can be applied in a control task. There are various quantities in a robotic system whose measurement may be useful, such as position, velocity, acceleration, force, torque, distance, etc. In a robotic face, the main concern is the position and orientation of facial features and gaze direction. These factors themselves depend on position of joints. Hence in the rest of this section we will focus on position sensors.

2.3.1 Potentiometers versus Encoders

There are different position sensors such as potentiometers, encoders, resolvers, LVDTs, etc. [Niku 01]. One may even use a sensor of a different quantity that is convertible to position. For instance, tachometer is a velocity sensor. But displacement can be computed from velocity, by integrating it over a time interval.

The commonest sensors in robotics used for position measurement are potentiometers and encoders. A quick study was done on each of them to gather necessary information for a comparison between these two types and finally selecting the adequate one for my project:

- Encoders have a higher precision, in terms of both linearity and resolution.
- Common encoders are incremental, that is they determine displacement, not the absolute position. But potentiometers determine the absolute position.
- Potentiometers have a shorter life, because they have some mechanical parts that produce friction, while encoders are optical and friction-free.
- Encoders are more expensive.

Since it is the absolute position of the joints that determines the appearance of the face, potentiometers are preferred. It is possible to measure displacement by an encoder and add it to the initial position for obtaining the actual position, but determining the initial position with only an incremental encoder is impossible. Some tricks may be used such as adjusting the joints manually for the first time, and whenever shutting down the robot, controller returns joints their origin again.

Nevertheless, a potentiometer is a compact and low cost package that can always work without human interference. The only problems of a potentiometer are its short life and low precision. Since in a robot face, joints do not move all the time, and they are often idle, there is no continuous friction. On the other hand, to simplify hardware design of the controller, we decided to represent position variables by 8-bit values. Although this resolution is low for the eyes of the robot, especially when the target of interest is far from the robot, but it is sufficient for facial features representation and their interpretation.

As always there is a trade-off between complexity and performance in the system. Since bigger potentiometers are usually more precise, I used large potentiometers. In addition, precision of numbers in the controller was chosen to be 8-bit so that the balance between performance and complexity is preserved.

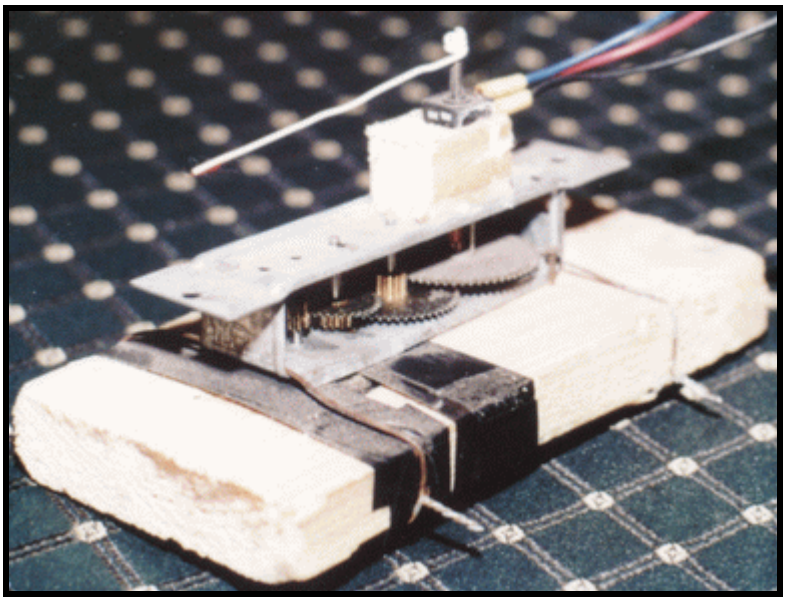


Figure 2.7 My first experimental home-built servo motor

Figure (2.7) shows my home-built servo motor during feasibility study of this project. I wanted to know if I could build a servomotor with a simple position controller to work properly regardless of its performance. A wooden board was chosen and a hole was created inside it to emplace the motor there. A simple DC motor was fitted inside the hole and got fixed by a black tape stick.

This experiment was done before producing our own gears, so a gearbox was taken out from a broken toy and its input was connected to the motor's shaft. You can see a small potentiometer coupled with the output shaft of the gearbox and a narrow wire showing the direction of the output motion.

This figure was brought here to justify our claim about building Aryan really from zero with very elementary tools and resources. Of course, we could bypass this stage by buying hobby servomotors if they were available in our country, and spend more time on other parts of the robot. Obviously, better results could be achieved.

2.4 Control

In this section we will review a classical and widely used closed-loop control method called PID controller. In this section we will study PID controllers theoretically. In the next chapter (chapter three) we will show how to build a PI controller by off-the-shelf digital components step by step.

2.4.1 Control Theory

In a closed-loop control system, the feedback signal is usually compared with the desired output to generate the appropriate control signal. A simple and common comparison criterion is the difference of these quantities, which is called error. In such controllers, they must then generate outputs that will decrease the absolute value of the error.

A classical example in control theory for such an error-based controller is a PID controller [Kuo 82]. It has been successfully applied to motor control problem, both in theory and practice. A PID controller feeds to the plant a signal equal to a linear combination of the error, its integral and its derivative.

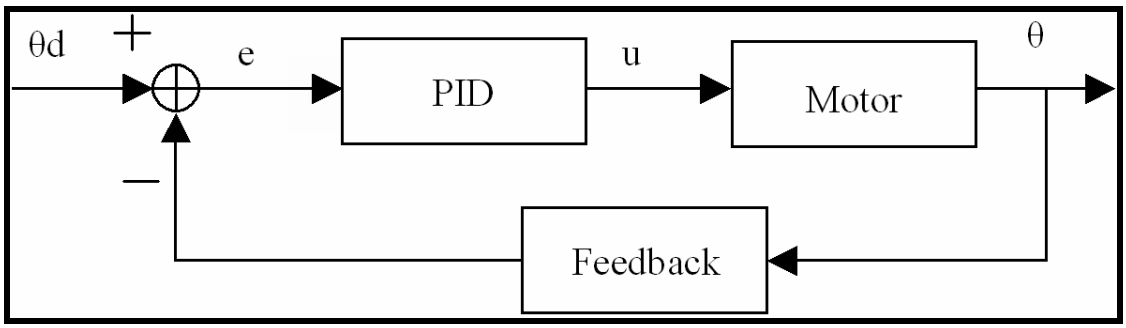


Figure 2.8 A typical PID controller

The linear coefficients are called proportional gain, integral gain and derivative gain respectively and dented by K_p, K_d, and K_i. The output of a PID controller is computed as follows:

$$u(t) = K_p e(t) + K_d \frac{de(t)}{dt} + K_i \int_0^t e(\tau) d\tau \tag{2.11}$$

$$U(s) = K_p E(s) + K_d E(s) s + K_i E(s) / s \tag{2.12}$$

For sake of simplicity, let's assume there is no disturbance (τ_l=0) so that we can use the transfer function of DC motor obtained in (2.10). In addition, let $B_m+K_bK_m/R$ be called *Beff* and K_m/R be *Keff*. We also assume that the feedback sensor is ideal, that is what it measures is equal to the real quantity. Therefore its transfer function is equal to unity. Realizing Figure (2.8) with these functions yields the system shown in Figure (2.9)

The overall transfer function of this system is of 3rd order. To ease analysis, let's consider the simplest case that K_d and K_i are equal to zero. This will reduce the order of the transfer function to two, see equation (2.13).

$$\frac{\Theta(s)}{\Theta_d(s)} = \frac{K_e K_p}{J_m s^2 + B_{eff} s + K_e K_p} \tag{2.13}$$

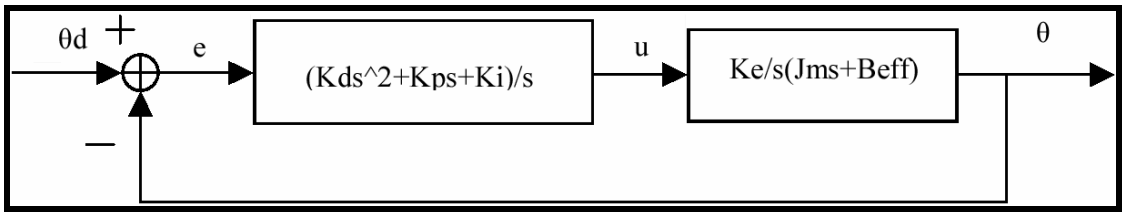


Figure 2.9 PID controller connected to a DC Motor.

The general behavior of system is determined from its characteristic equation. In our special case, a second order system. This equation is written as below:

$$s^2 + (B_{eff}/J_m)s + K_e K_p / J_m = s^2 + 2\varsigma\omega s + \omega^2 = 0 \tag{2.14}$$

Where the second equation is an equivalent representation with a physical meaning in terms of natural frequency and damping ratio, denoted by ω and ζ respectively.

$$\omega = \sqrt{\frac{K_e K_p}{J_m}}; \varsigma = \frac{B_{eff}}{2\sqrt{J_m K_e K_p}} \tag{2.15}$$

If ζ is greater than one, it is said to be an overdamped system. The response in an overdamped system has no oscillation or overshoot but a long settling time. If ζ is less than one, the system is underdamped, that is having both overshoot and oscillation.

It is clear that the fastest non-oscillatory response is when ζ is equal to one, which is called critically damped system. Natural frequency determines the frequency of oscillation when system is oscillatory.

In a second order system, natural frequency and damping ratio are dependent; when one increases, the other one decreases and vice versa. It's worth to mention that negative damping ratios are in fact amplifying ratios and will cause the system to become unstable. In our case since damping ratio is non-negative (due to the real positivity of system's parameters), stability is guaranteed.

Using (2.13) and final value theorem, the final output of the system is computed as follows:

$$\Theta(s) = \lim_{s\to 0}(sT(s)) = \lim_{s\to 0}(s\frac{K_e K_p \Theta_d(s)}{J_m s^2 + B_{eff} s + K_e K_p}) \tag{2.16}$$

Now consider an example where the desired response is a step function with an amplitude of θ_d to the controller, that is $\Theta_d(s) = \theta_d$.

$$\Theta(s) = \lim_{s\to 0}(s\frac{K_e K_p \theta_d / s}{J_m s^2 + B_{eff} s + K_e K_p}) = \theta_d \qquad (2.17)$$

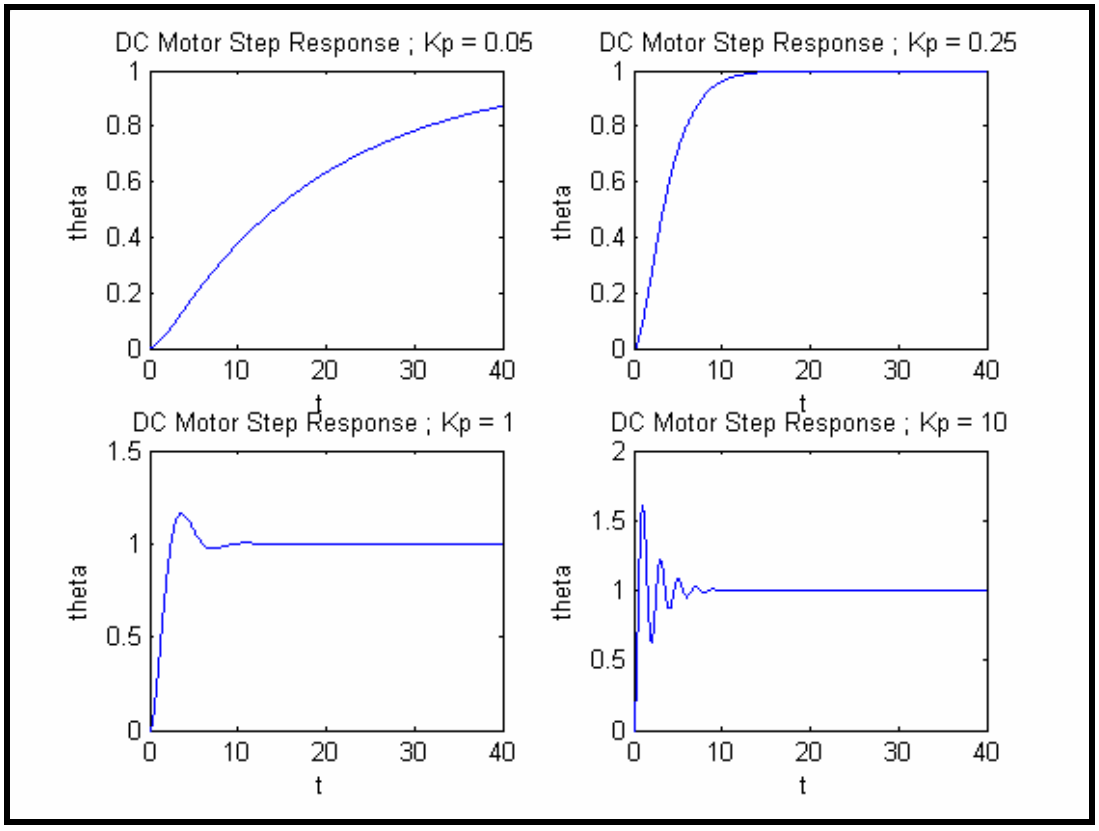


Figure 2.10 Response of DC Motor to P Controller

Therefore, our servo system is theoretically able to eventually reach the desired value with no error. To have a better understanding of what has been discussed so far, consider a DC motor whose parameters K_e, J_m and *Beff* are all equal to unity.

Now suppose that we are going to control this motor by a proportional controller. It's easy to compute damping ratio of critically damped situation for this system which will yield $K_p=0.25$. Three different behaviors of this system are examined by appropriate K_p's as shown in Figure (2.10).

Remember that we ignored the existence of any noise and disturbance when obtaining these results. Even when there is no external interference in the system, the effect of gravity on the load acts as disturbance. In addition, we assumed the motor is linear, while in practice there is a dead-zone∗ in its transfer function, which causes non-linearity. Therefore, achieving zero error by a proportional controller is not feasible in practice.

∗ When connecting a DC motor to a variable voltage source and gradually increasing the voltage from zero, the motor will not rotate until reaching a threshold voltage. This interval is usually called dead-zone. After passing the dead-zone, the motor will just start rotating such that increasing the voltage will increase motor's velocity.

Adding the integral of error to the control signal (PI controller) will highly improve the performance of a real system with disturbance. Consider the case where because of dead-zone or due to a disturbance, control signal is not able to rotate the motor. The integrator will accumulate these errors over time and therefore control signal is enlarged gradually until it overcomes the disturbance or dead-zone threshold and decrease error again.

Adding an integrator to has however two major drawbacks. First, the integrator will increase the lag of system's response. In addition, system will be of third order and our previous assumptions about stability will be no longer valid. In fact, there are gains, which can make the system unstable. Therefore contribution of integrator must be much less than proportional part. This is achieved by selecting Ki much less than K_p.

In most applications of servomotor control, using a P controller suffices. Those applications that require a high precision may adopt a PI controller. PID controller is not common for servomotors, particularly for position control. As discussed in section 2.3.1, since the resolution of position variables is limited to 8 bits, which is a medium resolution for our robot, even small errors in this scale may mean large misplacements. So we employed a PI controller to obtain almost zero error.

Although in our simulation shown in Figure (2.10) we assumed the motor parameters to be known in advance, say by on hand parameter estimation methods for a linear system, it is more convenient to tune a PID (or its simplified variants such as P, PD or PI) controller by changing controller's gains until attaining a satisfactory response.

Chapter 3

Controller

In the previous chapter I discussed how I had to build servomotors at home from off-the-shelf components. Unfortunately, I could not find an adequate controller chip in Iran for my motors either. Only a few DC servomotor controller/driver chips were available in the market but none of them could meet project's requirements because:

- Available chips were low power with maximum current of 1 Amp.
- They were based on Proportional Control law, which does not afford the desired precision.

Therefore I decided to build my own PI servo controller from available parts in our country. This reinvention of the wheel was present in all parts of my work and took a lot of my time. If these components were available, I would spend my time on other parts of the project.

Controller is a low power circuit, while motors draw a considerable amount of current. Therefore an intermediate module is also required to amplify the output of controller before being used by any motor as shown in Figure (3.1). Amplifier∗ is the topic of the next chapter. In this chapter we restrict our talk to controller only.

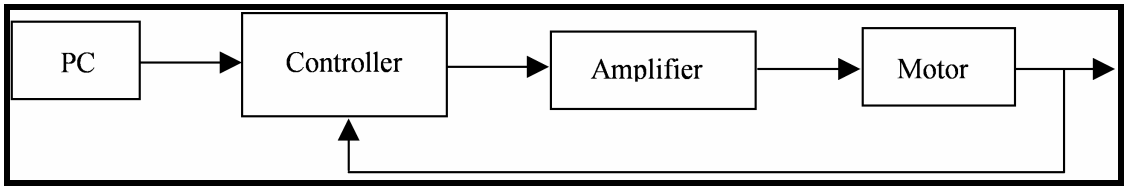


Figure 3.1 A practical motor controller

∗ Although amplifier is meaningful in electrical level, its elimination has not effect in control level. It is so because in my system, current is amplified without affecting voltage, while control signals are represented by voltages. So it was ignored in previous chapter for simplicity.

3.1 Analog versus Digital controller

For my home-built servo motor Figure (2.6) I implemented an analog proportional controller. It was an analog computer consisting of op-amps and resistors as shown in Figure (3.2). Its input came from parallel port of PC through a Digital to Analog Converter (DAC), as shown in Figure (3.3). Output was amplified using a pair of transistors in push-pull arrangement.

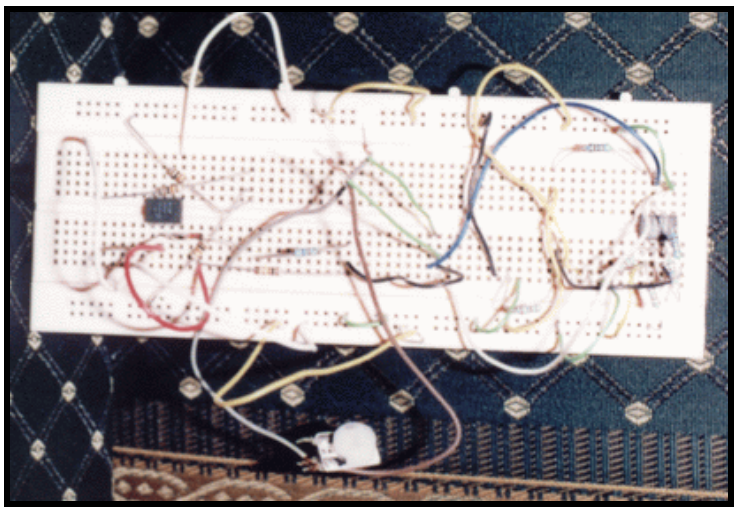


Figure 3.2 Home-built analog controller, analog computer part

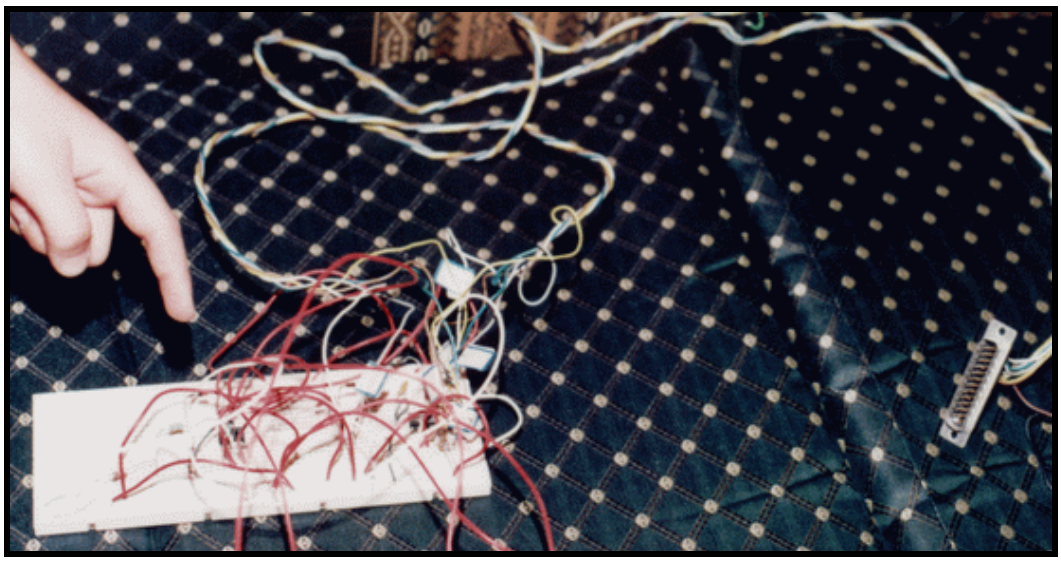


Figure 3.3 Home-built analog controller, Digital to Analog part

By accomplishing this experiment, advantages and disadvantages of an analog controller were understood practically. They were written down to be compared against a digital controller. Let me mention that since Aryan was to be comprised of a number of independent joints, an important factor was extendibility of a controller for multi motor case. The result of my comparison can be summarized as follows:

- A digital multi-channel controller is compacter than its analog counterpart because in analog case, each motor needs a dedicated pair of a Digital to Analog Converter (DAC) and a latch. These will increase the amount of wires on the circuit board. On the contrary, a digital controller only requires a multi-channel Analog to Digital Converter (ADC).

- Since the rate of issuing control commands is bounded to the speed of brain software which is itself limited to video capture's frame rate, commands look like step functions to the controller. Placing a low-pass filter between PC and controller smoothes these commands. Digital implementation of filters is easier than analog[*].

- The output of an analog controller is a continuous signal. So amplifying transistors (will be discussed in chapter four) work in active mode. Keeping power transistors in active mode may damage them due to the excessive heat. A digital controller deals with pulses that switch transistors either off or saturated. In these states transistor does not heat up so much as in active mode.

- There exists a dead-zone for transistors in small voltage regions. Since analog controller has a continuous control signal, it might be trapped into transistors' dead-zone, which causes distortion in the amplified control signal. On the other hand, pulses in digital controller are either zero or equal to the source voltage. Thus, the control signal always jumps over dead-zone.

- Using a combination of a Microcontroller and FPGA[ψ] chips, one can build a very compact controller. An analog controller, however, requires a large amount of small and discrete electronic components.

Therefore we concluded that the only disadvantage of a digital controller is its quantization error, that is the error caused by representing continuous data with a set of finite discrete numbers. It, however, does not seem so bad when taking tolerance of resistors and offset of op-amps into the account for an analog controller. There is no doubt that a digital controller is more adequate for our project.

∗ Despite of the fact that the operation point of an analog filter may change by changes in temperature, frequency, etc.), it also needs more components than just a single Resistor and Capacitor for high precision applications. Using op-amps or transistors can improve the performance of such filters but increases cost and also occupies more space on the circuit board.

ψ Field Programmable Gate Array (FPGA) is a compact programmable chip able to store a mid-size logic circuit.

3.2 Pulse Width Modulation (PWM)

Pulse With Modulation (PWM) is a method of actuating devices normally driven by varying voltages, with a series of high frequency on-off switching signals. This feature makes PWM very suitable for controlling analog devices by a digital device. The theory of this type of actuation is that the behavior of the actuators, in our case the DC motors, is based on the average power delivered to the device [Niku 01].

This can be understood by imagining a 12 volt source attached to a DC motor, but switched on and off (at a very high frequency) such that the power is on for half the time, and off for half the time. If the frequency is high enough (say 100 times per second) then the motor behaves in the same manner as if a constant 6 volts had been applied. Similarly, by varying the timing continuously, a variable voltage can be created which can be used in DC motors (see Figure (3.4)).

The amount of time that a PWM signal stays high in a period is called its duty cycle. PWM frequency is fixed and independent of its duty cycle, and it must be determined empirically for each motor. Too low or too high frequency results in a jerky motion for that motor. Higher frequencies are also more sensitive to noise.

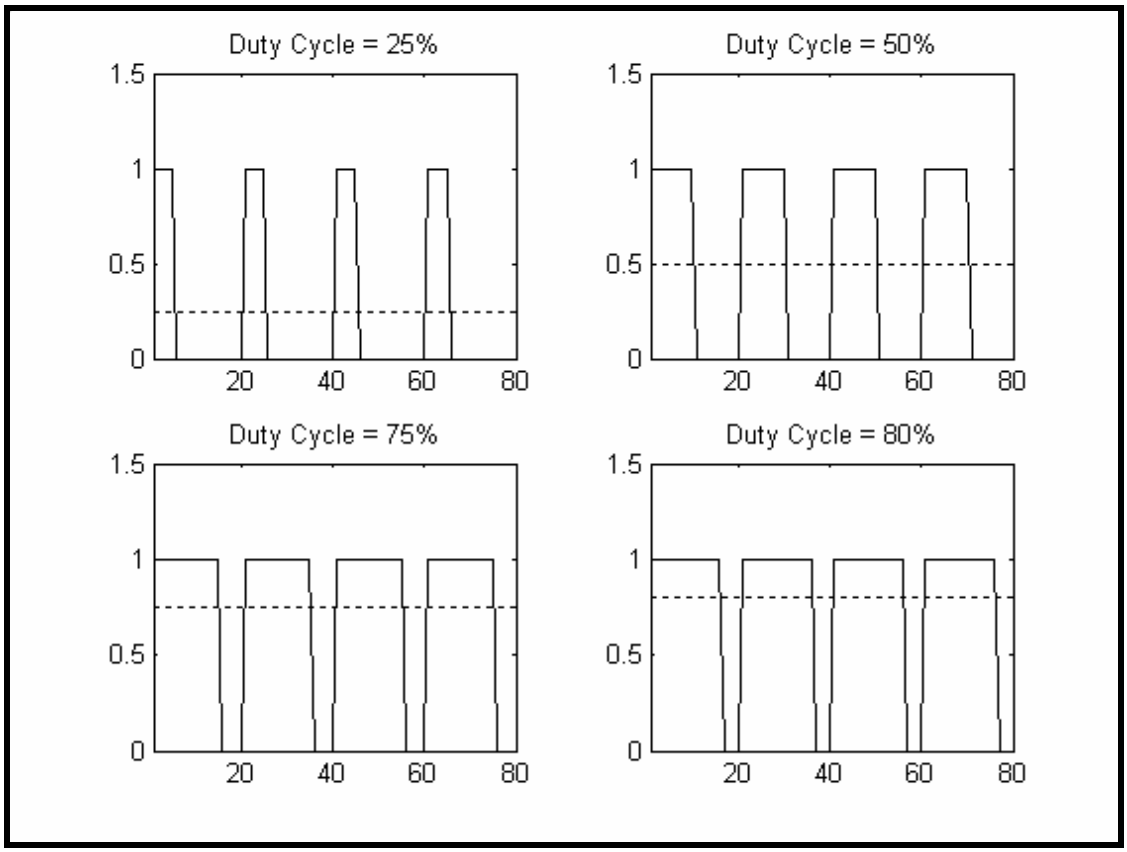


Figure 3.4 Different duty cycles of PWM

3.3 Controller Design

A digital controller requires a processing unit not only for computation of control signal, but also for managing subsystems in the controller. A microcontroller* is a processor with on-chip memory, I/O ports and interrupt controller designed and optimized for control applications. So a microcontroller is the heart of our design.

The algorithm that we used for computing control signal is based on PI control law (see section 2.3 for more details). So as the first step, the microcontroller must acquire desired and actual output values. Assuming a PC is issuing desired position of motors, microcontroller must be able to communicate with PC. In our design this communication is established through parallel port of a PC.

Since feedback signals are generated by potentiometers, they are analog quantities. An Analog to Digital Converter (ADC) is used to make these quantities usable for the microcontroller. Thus, having a communication mechanism between microcontroller and ADC is also inevitable.

In 3.2 we saw that PWM is an easy and appropriate method for actuating DC motors. It is possible implement PWM in software using microcontroller, but it will eat up some valuable resources of the microcontroller such as CPU time and free pins that could be used for other purposes. Therefore, we decided to build a hardwired PWM generator.

There are advanced microcontrollers in industrial countries, with built-in Analog to Digital Converter and PWM generator. What we could find in our market was a simple microcontroller (AT89C51). So we had to add ADC and PWM generator ourselves. PWM generator itself was also built from scratch by us using simple logic components programmed into a programmable logic device called CPLD.

The clock frequency of microcontroller and CPLD were chosen to be 12 MHz. ADC however receives a reduced clock frequency (500 KHz) from CPLD due to its speed limitation. Figure (3.5) depicts a general overview of our design. In this section we will discuss about each module in a separate subsection.

3.3.1 Common Bus

Two common buses interconnect modules of our controller. The key module, the microcontroller, dictated the structure of our bus. AT89C51 can handle an 8-bit data bus and a 16-bit address bus. The lower 8 bits of the address bus (*A0-A7*) are multiplexed with data bus (D0-D7). A line termed *Address Latch Enable (ALE)* indicates what is on the multiplexed bus at any moment, address or data. High order address lines (*A8-A15*) are dedicated.

---

* Controller should not be confused with Microcontroller. By controller we mean the whole system in a control task. Microcontroller is a chip and just a component of the controller.

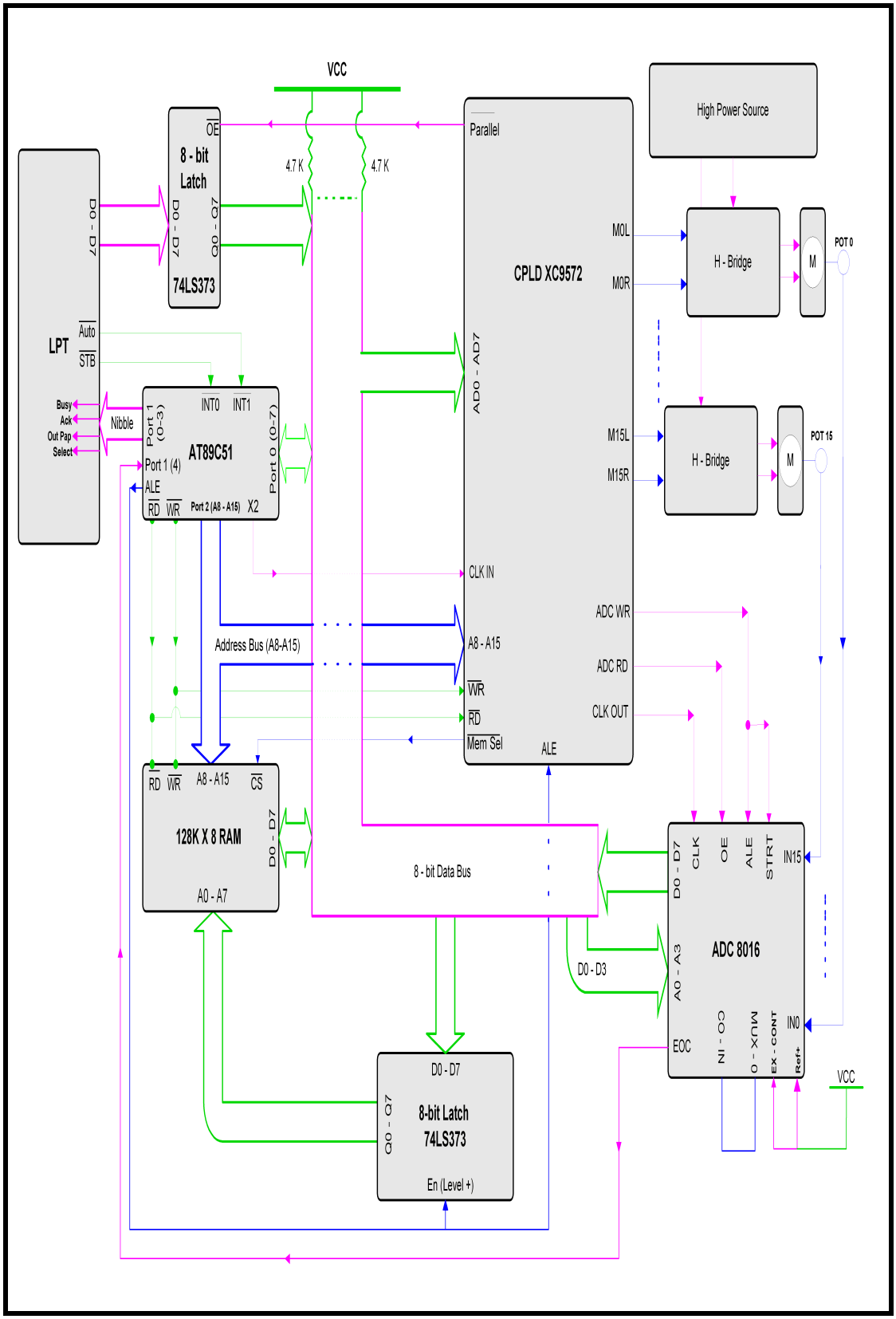


Figure 3.5 Schematic of our controller

When a read/write operation from/to RAM wants to take place, RAM module requires both address and data simultaneously on its lines. Since (*A0-A7*) of the bus are

multiplexed, a latch is used between the multiplexed lines of bus and (*A0-A7*) of the RAM. The following scenario occurs for a read/write operation from/to RAM:

- Microcontroller puts (*A0-A7*) on the multiplexed bus and sets *ALE* high. Thus contents on the bus are latched into 74LS374.
- Now (*A0-A7*) lines of RAM see the right signals from the latch.
- Microcontroller clears *ALE* and puts data on the multiplexed bus, which is connected to (*D0-D7*) of RAM too.
- Now (*D0-D7*) lines of RAM see the right signals from the bus.
- Since (*A8-A15*) were dedicated, now all signals are ready for RAM.
- Microcontroller determines type of operation using *RD/WR* lines.

Since data bus is shared among several modules and they are all able to write there, collision may occur and corrupt contents on the bus. To avoid this, an address decoder is employed to disconnect all modules from the bus (high-impedance state) except the one whose address is given to the address decoder. Such a decoder has been built from simple logic components inside of CPLD. Address decoder controls the following modules to prevent from collision:

- RAM by its *Chip Select (CS)* line.
- Analog to Digital Converter (ADC 8016) through its *Output Enable (OE)* line.
- The latch (74LS73) used to isolate (output lines of) parallel port from data bus. This latch can go to high-impedance state by controlling *Output Enable (OE)* line.

Port 0 of the microcontroller has open drain outputs. That is when 0 is written to one of the bits in port 0, that pin is pulled low (0). But, when 1 is written to one of the bits in port 0, the pin goes into a high-impedance state. To be able to get 1 as output, an external pull-up resistor must pull up the port (to 1) when the port is in its high-impedance state. Typical value for such pull-ups is 4.7KΩ.

3.3.2 Parallel Port

A bi-directional communication channel between PC and controller is required. The path from PC to controller is used to transmit the desired position of motors. Another path from controller to PC must exist to report the current position of motors upon request. In fact, feedback signals are not only used by the position controller itself, but also by PC for high level tasks such as computation of visual-motor mapping.

Such an Input/Output channel could be external bus slots or I/O ports. The bandwidth and hardware capabilities of an external bus are much more than I/O ports, so devices that require a high-speed and sophisticated communication utilize these slots. It's obvious that building such devices is not and easy task.

I/O ports are used by devices that do not need very high bandwidth and complicated communication protocol. Some common PC ports are Parallel, Serial and USB. Among them, parallel port has the easiest communication scheme. A major reason might be the ability of sending 8-bit data that is consistent with byte definition, the smallest accessible unit by microprocessors.

Since the rate of issuing motor commands is bounded by processing time of visual information in the robot, say 30 frames per second, a low bandwidth channel suffices for our application. In addition, messages that are passed between PC and motor are very simple. So using an I/O port is reasonable. Parallel port was preferred because of its simplicity.

Unfortunately a standard parallel port can transfer byte-oriented data only from PC to a device not vice versa. This is because there are only five input lines on parallel port, called status lines. Using four of them, one can send two nibbles of a byte separately from an external device to achieve a complete byte. This however requires two operation cycles that slows down the communication. But it is still ok for our low-bandwidth application∗.

Table 3-1 Parallel Port Lines Description

Pin #	Type	Register	Name	Role
2	Output	Data	D0	Output Byte (bit 0)
3	Output	Data	D1	Output Byte (bit 1)
4	Output	Data	D2	Output Byte (bit 2)
5	Output	Data	D3	Output Byte (bit 3)
6	Output	Data	D4	Output Byte (bit 4)
7	Output	Data	D5	Output Byte (bit 5)
8	Output	Data	D6	Output Byte (bit 6)
9	Output	Data	D7	Output Byte (bit 7)
14	Output	Control	Auto Feed	Get Position of a Motor
1	Output	Control	Strobe	Set Position of a Motor
13	Input	Status	Select	Input Nibble (bit 0)
12	Input	Status	Out of Paper	Input Nibble (bit 1)
10	Input	Status	Acknowledge	Input Nibble (bit 2)
11	Input	Status	Busy	Input Nibble (bit 3)

∗ The bandwidth required in the input channel of controller should not be confused with its output channel. Outputs are PWM signals used by motors and must be updated with a high enough rate to resemble a continuous motion.

Recently new standards such as Extended Capability Port (ECP) and Enhanced Parallel Port (EPP) [Buchanan 99] have been proposed. These standards allow bi-directional byte-oriented use of parallel port. We however preferred compatibility and simplicity of a standard parallel port, especially when there is no problem with low speed of nibble-mode data transmission.

Parallel port lines are accessible through three registers called Data Register, Control Register and Status Register, the latter one being input and others being output. Pin out description of the port is available in Appendix B. Each line has a standard name* regardless of its role in its application. Each application requires its own lines. So the actual role of lines is determined by application. Table 3.1 summarizes parallel port lines and their role in our design.

3.3.3 Complex Programmable Logic Device (CPLD)

Some parts in our design such as divider, PWM generator and address decoder could be built from simple logic circuits. To make the final circuit as compact as possible, we decided to program these logic circuits into a programmable logic device. Another advantage of this method is that the whole task (design, simulation and optimization) can be accomplished in a software environment. When everything is ready, the design is transferred to the programmable device using a programming device.

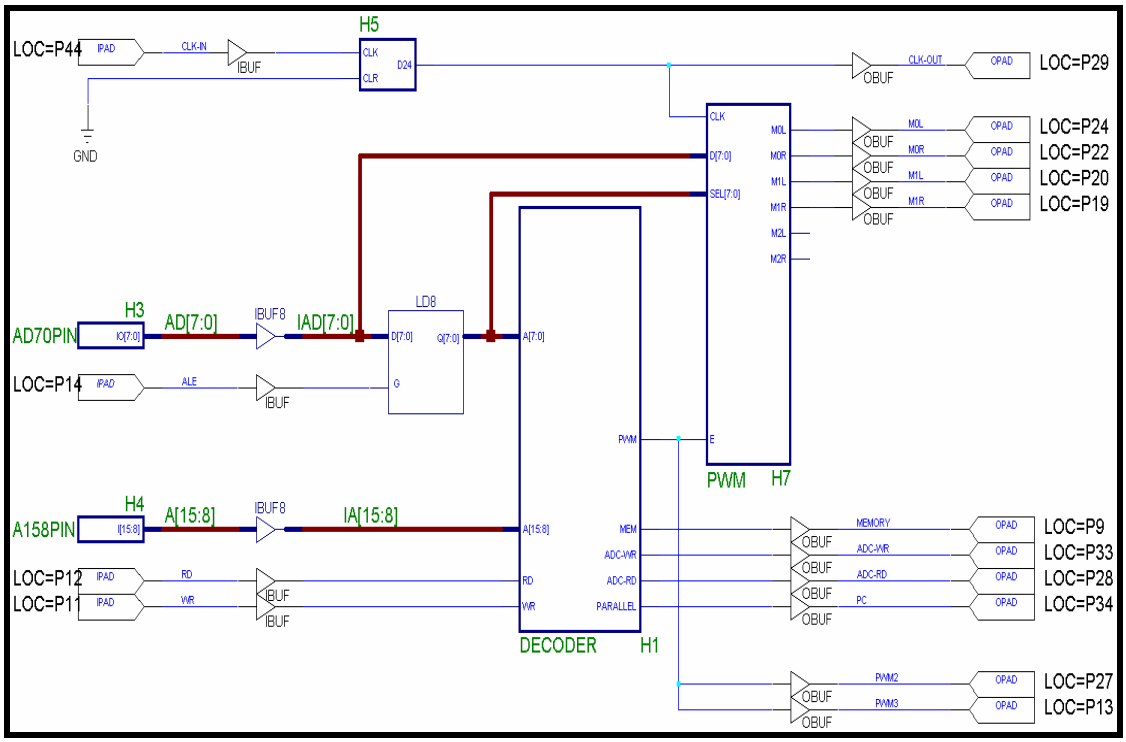


Figure 3.6 building blocks of the first CPLD

* Originally taken from the main application of parallel ports, that is connection to a printer.

Two common programmable devices are Complex Programmable Logic Device (CPLD) and Field Programmable Gate Array (FPGA). We used CPLD because it was less expensive than FPGA. Xlinx XC9572 was what we could easily find in our market. It contains 72 macrocells with 1600 usable gates and can work with a clock frequency up to 83.3 MHz. We used XC9572's with 44-pin PLCC package. Each of these chips costs 3,100 toomans or equivalently 3.87$.

In our design we employed three CPLD's because each one had a limited capacity and could not house the whole circuit. We placed an address decoder, a divider by 24 and two PWM channels in the first CPLD. The rest of CPLD's were the same, each one containing 5 PWM channels. It is possible to duplicate the latter CPLD many times to obtain 16 PWM channels without need to touch the rest of circuit.

The reason of not putting two latches of Figure (3.5) inside of CPLD's is the large number of occupied pins by them. Since a latch has parallel input and parallel output, bearing in mind 8-bit data, each latch occupies at least 16 pins, which is too much with respect to the limited number of available pins in a CPLD.

The block-diagram of the first CPLD is shown in Figure (3.6). Other CPLD's are all the same and similar to the first CPLD. The only difference is that other CPLD's only contain PWM generators. Since decoder and divider are eliminated in other CPLD's, there is more room for PWM generators. We could fit up to 4 PWM generators in each CPLD.

The first CPLD receives the original clock generated by crystal, which is 12MHz.It then divides this frequency by 24 to obtain a reduced clock frequency (500 KHz) appropriate for PWM and ADC modules. The reduction is required due to the frequency limitation of motors and ADC. The reduced frequency is used by internal PWM and also sent to ADC and PWM's in other CPLD's using *CLK_OUT* line.

Multiplexed bus is demultiplexed in the CPLD by *ALE* line and an 8-bit latch (LD8). Complete 16-bit address is then sent to address decoder. Data bits, low order bits of address and an enable line (from address decoder) are sent to PWM submodule (for updating motors' velocities).

Now we explain each of CPLD submodules in details.

3.3.3.1 Address Decoder

To simplify and unify memory and I/O addressing, we used memory mapped I/O method. That is reading from or writing to an I/O device is achieved by reading/writing from/to certain addresses of memory. An address decoder is responsible to notify the device whose address has been detected on the address bus.

Although addresses were assigned to modules arbitrarily, they were such that their decoding was easy. The resultant memory map is shown in Table 3.2

Table 3-2 Memory Map

From Address	To Address	Device	Access
0000	EFFF	Memory	Read/Write
F000	F000	Parallel Port	Read
F100	F100	ADC	Read/Write
F200	F21F	PWM Channels	Write

3.3.3.2 Divider by 24

The divider is comprised of five flip-flops. The first two flip-flops are of JK type and divide the input frequency by three. The rest are Toggle flip-flops, each of which divides its input frequency by two (because after a complete clock period, the state of a T flip-flop is toggled). This has been shown in Figure (3.7).

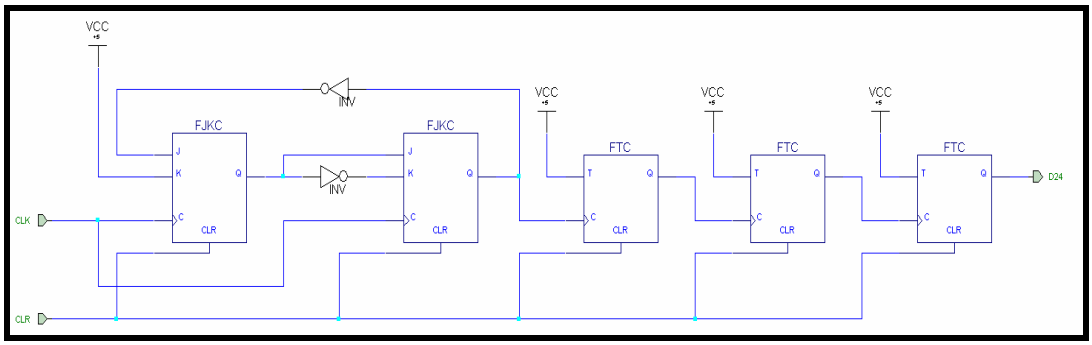


Figure 3.7 Division by 24

3.3.3.3 PWM Generator

Based on PI control law, microcontroller computes voltages to be applied to motors in form of PWM signals. Since 16 motors are potentially supported in our design, 16 PWM channels are required simultaneously, which needs 32 pins of the microcontroller (each PWM channel has two wires, to represent magnitude and direction). But this amount of pins is too high for our microcontroller.

Multiplexing might be a solution but yields a poor result because each PWM channel is inaccessible for sometime until it is selected again. During this time, the channel might be either high or low, it does not matter. What is important is that this period affects the average value of the PWM signal.

A more reasonable but also more costly approach is to use 16 PWM generators, each one with its own input latch. Then microcontroller just needs to update the content of a latch whenever needed. This method is more efficient not only in terms of the required number of pins, but also reducing computational load of the microcontroller.

Inputs to PWM module are as follows:

- Reduced clock
- Enable line controlled by address decoder
- Demultiplexed *data*
- Demultiplexed low order address (*A0-A7*).

Two output lines exist per motor. Each pair must be eventually connected to two inputs of an H-Bridge (H-Bridge is the mechanism we used of our amplifier and for driving motors. It will be discussed in chapter four). At any moment, only one of these lines is carrying pulse, and the other one is low. Consider a motor, the width of pulses determines the velocity of the motor. Also depending on which line is carrying these pulses, the direction of rotation is determined. This is how an H-Bridge works.

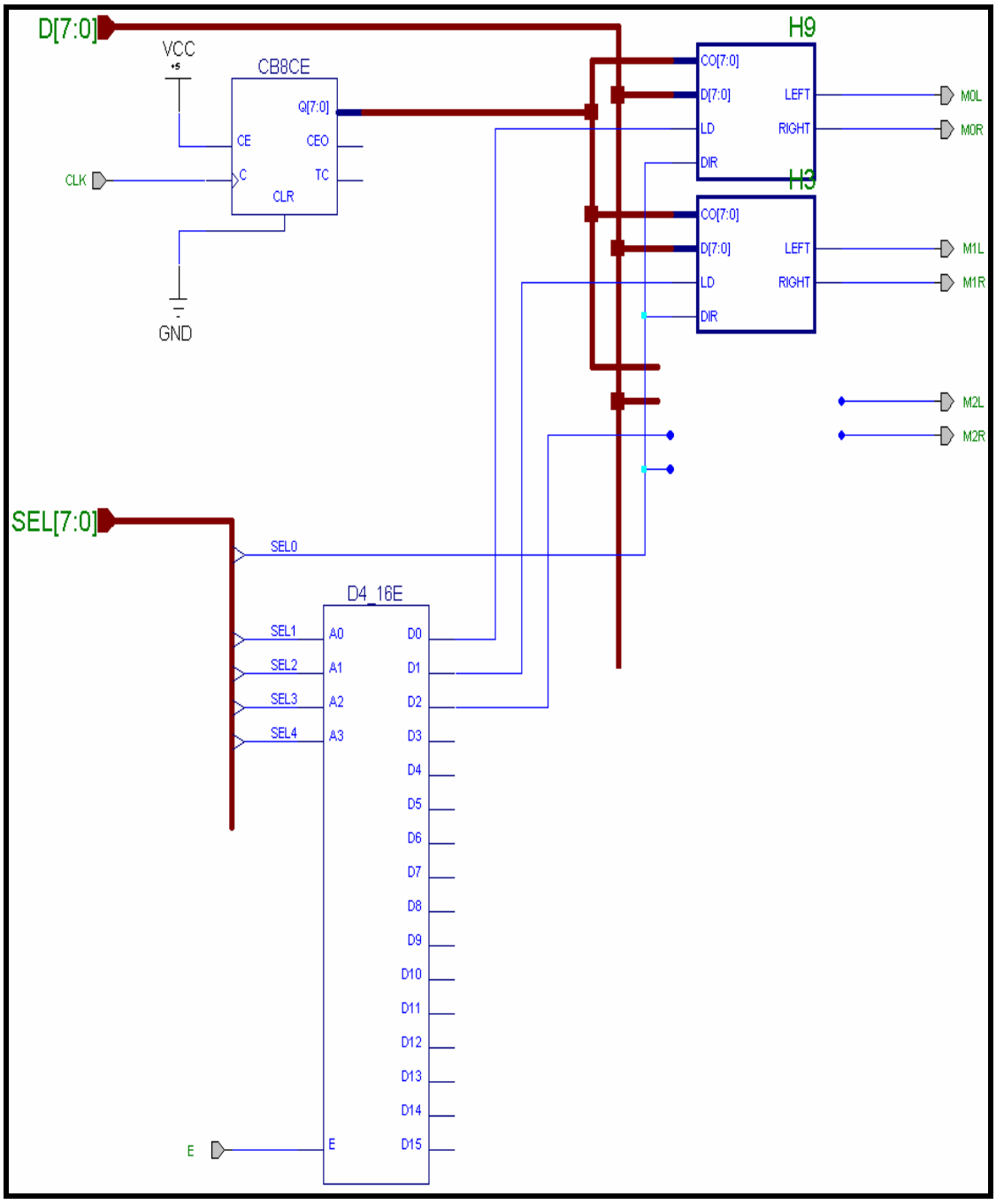


Figure 3.8 Decoder and PWM generators

To change the velocity of a motor, PC writes the desired value to the memory mapped address of that motor. We wanted the resolution of PWM pulses to be 8-bit, that is having 256 distinct widths. One more bit was required to indicate direction of rotation. So it seems two memory write operations must be done, one for setting pulse width and the other for choosing direction.

We rather used an elegant approach to choose both direction and pulse width by one memory access. Two successive bytes were assigned to each motor. In our approach, writing pulse width byte to the even address of that motor would set its velocity to the new value, whereas writing to its odd address had the same effect but in the opposite direction. In our design *A0* determines direction of rotation and (*A1-A4*) determine what motor to be chosen using a 4 to 16 decoder. This can be seen in Figure (3.8)

To generate a pulse with an adjustable width, we used a counter (up or down does not matter), a latch and a comparator for each PWM channel. The 8-bit counter counts all the time. The 8-bit latch also keeps the desired value of the pulse width. They are compared and if the desired value is greater than counter's value, output goes high, otherwise stays low.

To understand how this combination yields a train of pulses with adjustable width, without loss of generality, let's consider the desired width to be constant at the moment and denoted by *A*. In addition, let's denote counter's state by *B*.

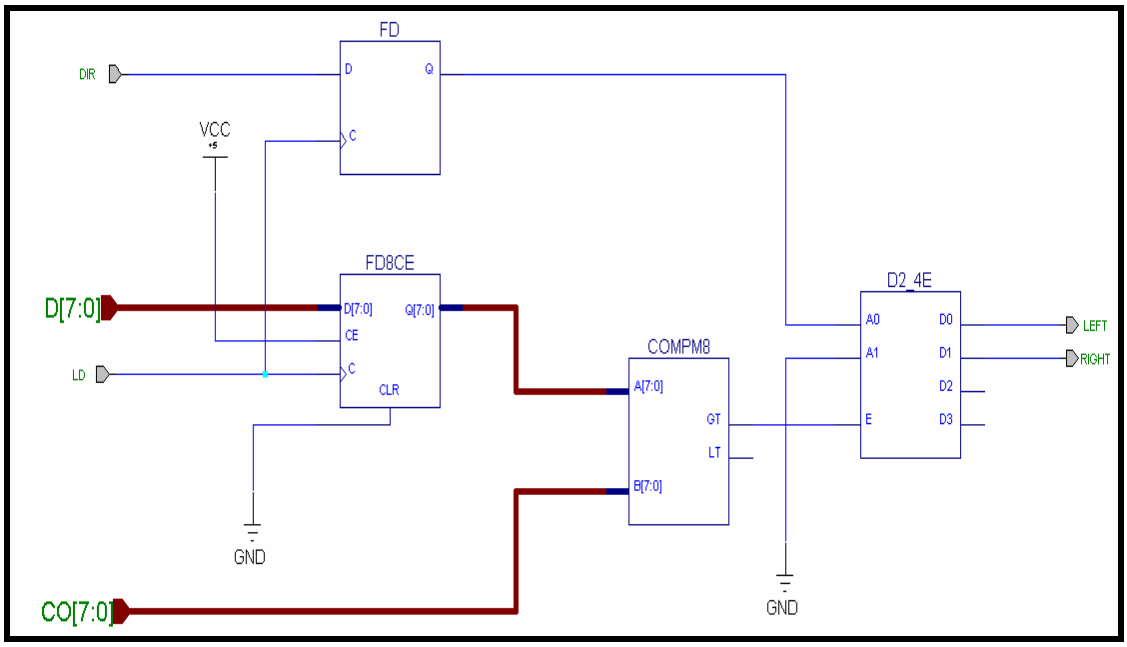


Figure 3.9 Inside of a PWM generator

From Figure (3.9), when *B* is less than *A*, output goes high. Considering a complete loop, that is starting *B* from zero and ending with 255, there are exactly *A* number of values for *B*, which are less than *A*, i.e. 0, 1, … *A*-1. So the ratio of high states to total states in a loop is computed as *A*/256. When *B* reaches 255, in the next clock it will return to zero and everything will be repeated as before. So high to total ratio computed from *A*/256 is anytime valid.

The generated pulses are combined with direction bit using a decoder as shown in Figure (3.9). Pulse state enables or disables the decoder. Depending on the state of direction bit, one of two output channels goes high and the other goes low. Totally, at any moment, only one of the outputs is carrying pulses and the other channel is silent. This is what we will need in H-Bridge to drive motors (We will discuss about H-Bridges in chapter 4).

Since counter is not affected by any signal except clock, its state is independent of the rest of circuit and one counter can be shared among all comparators. The shared counter is shown in Figure (3.8).

We mentioned in 3.3.3.1 that the clock frequency that PWM receives is 500 KHz. So there are 500K/256=1869 pulse cycles per second.

3.3.4 Analog to Digital Converter (ADC)

Actual position of each joint is determined using a potentiometer and measuring the analog voltage dropped across its terminals. Analog signals must be converted to digital to be usable for digital controller. We employed ADC8016, a product of National Semiconductors™. It is a low power (15 mW) 8-bit ADC, with a 16-channel multiplexer and microprocessor compatible logic.

Some characteristics of this device are as follows:

- A convenient interface to microprocessors by latched and decoded multiplexer address input and latched TTL tri-state outputs.
- An easy cascading capability if more than 16 channels are required.
- The device does not need any zero or full-scale adjustment externally.

I bought this chip with 4,500 toomans or equivalently 5.6$.

According to typical application schematic of ADC8016 datasheet, *Comparator In* and *Common Out* pins must be tied to each other and expansion line must be connected to V_{cc}. A clock frequency of 500KHz is recommended. Such a frequency has become available after reduction, in the *Clock Out* pin of CPLD.

V_{ref+} and V_{ref-} are voltages corresponding to zero and full-scale digitized signals. The maximum value for V_{ref+} is 5 volts. Since our power supply was 5-volt too, we connected V_{ref+} and V_{ref-} to V_{cc} and ground respectively. Accordingly, two ends of each feedback potentiometer were also connected to V_{cc} and ground.

Conversion register of ADC is reset on the positive edge of *Start Conversion (SC)* pulse and conversion is begun with its falling edge. ADC8016 has 16 multiplexed input channels. So four address lines (*A0-A3*) are required to select a channel. An *ALE* line also exists on this chip to handle multiplexed buses. When *ALE* is high, it means contents on (*A0-A3*) are valid (addresses).

Conversion takes a short time to complete, in ADC8016 about 100 ns. To notify microcontroller about conversion completion, a line termed *End of Conversion (EOC)* goes low during conversion and when conversion is finished it goes high. In addition, ADC's output (*D0-D7*) can be connected/disconnected to/from bus by *Output Enable (OE)* line of the ADC.

When address decoder detects ADC's address, it enables either *ADC RD* or *ADC WR*, depending on the state of *RD* and *WR* lines of microcontroller. To read a feedback channel, microcontroller first writes the channel number to the memory-mapped address of ADC that makes *ADC WR* high.

Connecting *ADC WR* to *ALE* and *Start* lines of ADC makes it start conversion. Note that *ALE* of ADC is different from *ALE* of microcontroller. When *ALE* of microcontroller goes high, the memory-mapped address is decoded to generate *ADC WR* signal. Next, *ALE* of ADC is obtained from this line. So when ADC thinks there in an address on the bus, there is really data there, the channel number of the ADC to be read.

ADC signals CPU when converted data is ready, using *End of Conversion (EOC)* line. When conversion is started *EOC* goes low, and it comes back to high when conversion is over. Converted data are stored in an internal latch inside of ADC. Microcontroller polls *EOC* line to see when ADC is ready for reading the converted data.

When this time arrives, microcontroller executes a read instruction from the memory mapped-address of ADC. This time address decoder enables *ADC RD* signal, which is connected to *Output Enable (OE)* line of ADC. So data will be available on the bus and will be read by Microcontroller.

Pin-outs of ADC8016 is available in Appendix B.

3.3.5 Microcontroller

AT89C51 is a product of Atmel™ It is a general-purpose 8-bit microcontroller with 4K bytes flash memory and 128 bytes of RAM. It has four ports each of which has its own capabilities. Port 0 is an 8-bit port. It was configured to be a multiplexed data/address bus (low order address (*AD0-AD7*) lines are meant). *ALE* determines if microcontroller has put address or data on the multiplexed bus at the moment. More details about multiplexed bus can be found in section 3.2.2.

Bit 5 (starting from 0) of this port is used as an input to receive *End of Conversion* signal from Analog to Digital Converter. As discussed in 3.2.4 this line is necessary because conversion takes a short time and the output of ADC is not valid during this period.

When this signal arrives, microcontroller reads output of the ADC. Port 1 is a bi-directional 8-bit port. We used its four lower bits as output and connected them to status lines of parallel port. AT89C51 sends data to PC in nibbles using these pins. More details about nibble-based data transfer can be found in section 3.3.2.

Port 2 was configured to emit high-order address lines (*A8-A15*). In this mode, it uses strong internal pull-ups when emitting 1's. Port 3 is an 8-bit bi-directional that also serves the functions of special features of AT89C51. The features that we used were two interrupt lines *INT0* and *INT1* and memory read/write strobes *RD* and *WR*. Bit numbers of these signals in port 3 are 2,3,6 and 7 (starting from 0) respectively. All of these lines are low active.

Pin description of AT89C51 with PDID package, the one that we used, is available in Appendix B.

3.3.6 RAM

An approach for a digital implementation of the integrator in a PI controller could be based on a sliding window. It is a fixed-size window that holds a history of recent error values. Details of this method will be discussed in section 3.4. Here, what we worry about the memory to keep this history.

In our software, error is represented by 4-byte floating points. Assuming the size of each window to be 256 samples and having 16 motors to be controlled, 16K bytes of RAM is needed. This is larger than the internal 128-byte RAM of AT89C51. Obviously, an external RAM chip is required.

We used UPD431000ACZ70LL; a product of NEC™ which is a 128Kx8 SRAM. It is a low power CMOS 32-pin chip with a maximum access time of 90 ns and maximum current of 70 mA. I bought this chip with by 2,700 toomans or equivalently 3.5$.

SRAM requires the complete 16-bit address and 8-bit data (if data is to be written to memory) simultaneously. So a latch (74LS374) is used to demultiplex *AD0-AD7* using *ALE* signal, as described in section 3.2.2

RAM pins are connected to the rest of system as follows:

- Address lines of RAM are connected to *A0-A15*.
- Data lines of RAM are connected to data/address bus.
- *Output Enable (OE)* line of RAM is connected to *RD* of microcontroller.
- *Write Enable (WE)* line of RAM is connected to *WR* of microcontroller.
- *Chip Select (CS)* line of RAM is controlled by address decoder to disconnected RAM from bus, when the address is not in the addressable range of this memory as shown in memory map in table 3.2.

Pin outs information of this chip can be found in Appendix B.

3.4 Controller Software

Our software is comprised of two interrupt service routines (ISR) and a main control loop. The two interrupts are either for updating the position of a motor, or for reading the actual position of a motor, activated by *Strobe* and *Auto* lines respectively (See section 3.3.2). The main loop computes pulse widths of PWM signals based on a PI control law to compensate error.

The algorithm proposed in this section is implemented in c and used to control Aryan's joints. The complete source code is available in Appendix C.

3.4.1 Writing to controller

If system is not initialized yet, update interrupt uploads control gain parameters for motor control from PC to microcontroller. Generally, this ISR transfers data from PC to Microcontroller. After initialization, PC uses this ISR to update the desired position of a motor in the memory of microcontroller.

When gain constants are to be transferred, they are broken into 8-bit data due to the limitation of parallel port sending out data. We used 4-byte floating points for representing the gains. An internal counter is used by this routine and increased by one unit each time this routine is called. So it always knows to which byte of which constant is being transferred.

When this ISR is used to update the desired position of a motor, it must read from PC twice, first for determining which motor to choose, and next for selecting the new desired position. An internal flag is used inside of this ISR to point out in which of these two states it is. The flag is always toggled at the end of the ISR.

The whole description of this ISR can be summarized in the following pseudo code:

```
Static int I = 0
Static int count = 0
Static Boolean flag = false
Static Boolean gain_p = false
Static char motor

If init_done is false then
{
    float_data.byte[I] = [parallel_port]
    I = I+1
    If I = size_of_float then
    {
        If gain_p = true then
        {
            kp[count] = fdata
            gain_p = false
        }
        else
        {
            ki[count] = fdata
            gain_type = true
            count = count + 1
```

```
        }
        I = 0
    }
    if count = number_of_motors then init_done = true
}
else
{
    if flag is false then motor = [parallel_port]
    else set_point[motor] = [parallel_port]
    toggle flag
}
```

3.4.2 Reading from controller

The other ISR is used to report the actual position of a motor to PC. This routine must be called twice too. In the first entrance, first it reads parallel port's data to determine which channel to read from. Next, It reads position of that motor from ADC channel and stores the result (to be used in the second entrance).

This position value is broken into two nibbles. The first half is also sent in the first entrance. In the second entrance, only one thing is left, sending the second half to PC. An internal flag inside of this routine determines in which of these two states it is. The flag is toggled at the end of the ISR.

The whole description of this ISR can be summarized in the following pseudo code:

```
Static char nibtemp
Static Boolean flag = false
Static char motor

If flag is false then
{
    motor = [parallel_port]
    nibtemp = actual_position[motor]
    write_low_order_nibble_of_nibtemp_using_individual_status_pins
}
else
{
    shift nibtemp four bits to right
    write_low_order_nibble_of_nibtemp_using_individual_status_pins
}
toggle flag
```

3.4.3 Main control loop

The main control loop is executed continuously regardless of incoming interrupts. Its inputs are desired and actual position of a motor, and its output is the control voltage that must be applied to that motor. A flowchart of this loop is depicted in Figure (3.10).

It is worth to mention that interrupts must be disabled when reading from ADC, otherwise an interrupt might interfere with reading feedback in the control loop. This may happen when the second request arrives while ADC is in the middle of conversion for the first request. This causes the conversion to be restarted and the first request to be lost.

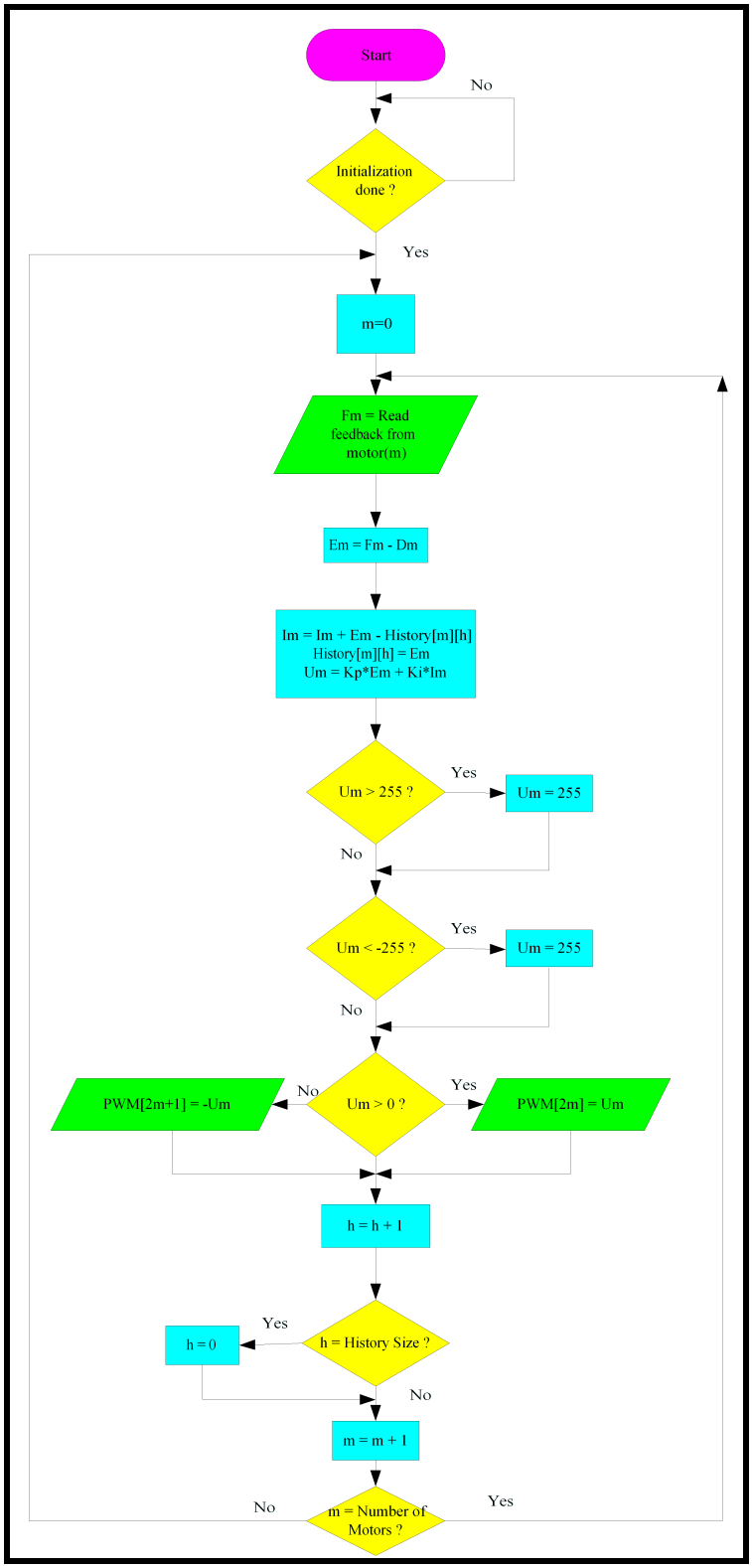


Figure 3.10 Flow-chart of control algorithm

The following pseudo code can be used to overcome this difficulty:

```
Disable interrupts
Read from the m'th channel of ADC
Busy Wait here as long as high End of Conversion is high
Enable interrupts
Busy Wait here as long as End of Conversion is low
```

Since ADC sets *EOC* low during conversion, we must wait by the first while until ADC gets our command and starts conversion. When ADC becomes idle again, is sets *EOC* high, so in the second loop we will wait until it finishes the conversion.

The rest of algorithm can be explained from the flowchart in Figure (3.10). It can be seen that there are two loops in the algorithm. The inner loop updates the control signal of motors. The outer loop is endless and merely repeats updating process forever.

History is a two-dimensional array that stores the most recent error values for each motor. So motor number and time step index it. The size of history for motors is arbitrary; we chose 256 error values (each error value is represented by a byte in our software) for each motor.

History ensembles a circular array for error values. At the bottom of the inner loops, variable *h*, the index of history array is increased by one and compared with the size of history. If equal, *h* is reset to zero. The overall procedure keeps the most recent errors, from index (*h*-1) to (*h*-1+*history_size* modulo *history_size*) in the history array.

The control rule is based on PI law that was discussed in chapter 2. Integral is approximated by a finite summation over the error values in the history. This approximation could be achieved by summing the elements in the history every time, but it would require another loop to select these elements for summation.

A more efficient solution is to accumulate errors by successively adding the current error to the previous integral value. So what is the use of history? Accumulating errors over time results in a phenomenon called wind up. It means the weight of previous errors becomes so large in the summation that they remain the dominant value for some time. This increases latency time in the system.

By subtracting the oldest element of the history from the approximated integral in each update, integral will always keep the sum of the most recent errors without need for a new loop. So an efficient integral approximation is achieved.

In our design, a control signal is represented by a byte, but the result obtained from PI control law may exceed this range, depending on values of K_p, K_i and recent errors. This can give rise to an undesired control signal when values are out of range. A saturation operation prevents from this problem.

Finally, the value of the control signal is written to the memory-mapped address of the m'th PWM generator. Remember from 3.3.3.3 that writing to an even address determines direction of rotation for the corresponding motor.

3.4.4 Actuator Calibration

The role of feedback signal measured by each potentiometer is to help for finding the actual situation of the corresponding revolute or prismatic joint. This requires combining various equations to rule out intermediate variables and obtaining a relation in terms of the desired variables only.

In our case the desired variables are the situation (position or angle) of a facial feature and the corresponding measured feedback signal represented by a byte value. Here is the chain of equations relating these two variables:

- Situation of a facial feature and its force transmitter.
- Situation of a force transmitter and angle of potentiometer's knob.
- Potentiometer's knob and corresponding resistor value.
- Resistor value and the voltage the voltage of its middle terminal (with respect to ground).
- The voltage of potentiometer's middle terminal and its byte value generated by Analog to Digital Converter.

As we saw in chapter 1, Aryan can only express discrete emotions. So all we need is a look-up table associating each emotional state to the corresponding byte values for facial feature joints. These values are called calibration values and can be obtained empirically. We can manually change the set point of each joint (in terms of byte values) and write down the desired values for each emotion.

However, for eyes and neck, discrete positions are insufficient because they do not depend on discrete emotions, but on the location of person of interest in space. Therefore, we require an equation relating their byte values and angular situations. Fortunately all intermediate equations mentioned above follow an almost linear model. So combining and rewriting them in terms of desired variables also yields a linear model:

$$v = a\theta + b \tag{3.1}$$

where v and θ denote byte value and angular situation of a joint respectively. a and b are called calibration parameters. Since these parameters may differ for each joint they must be estimated separately.

For each joint, we collected a few (θ,v) pairs and fitted them to a linear model by least squares method. This provides the optimal estimation for calibration parameters in terms of sum of squared errors criterion. Assuming a collection of N (θ,v) pairs, the solution of linear regression is computed as follows:

$$\begin{cases} a = \dfrac{\sum_{i=1}^{N}\theta_i v_i - \dfrac{\sum_{i=1}^{N}\theta_i \sum_{i=1}^{N} v_i}{N}}{\sum_{i=1}^{N}\theta_i^2 - \dfrac{(\sum_{i=1}^{N}\theta_i)^2}{N}} \\ b = \dfrac{\sum_{i=1}^{N}\theta_i^2 \sum_{i=1}^{N} v_i - \sum_{i=1}^{N}\theta_i \sum_{i=1}^{N}\theta_i v_i}{N\sum_{i=1}^{N}\theta_i^2 - (\sum_{i=1}^{N}\theta_i)^2} \end{cases} \quad (3.2)$$

3.4.5 Smoothing filter

An important factor that helps the Aryan to look natural and life-like is continuity and smoothness of its motions. A necessary but not sufficient condition is to detach motor control system from the main processing unit. Since motor control task is much simpler than processing done by the main processor, such as image processing task, this split allows the controller to loop with a much higher rate than the processor.

In our system, building a separate controller to control joint positions could satisfy this condition. The main processing unit, which is called Brain in our system, only issues position commands. Speed of Brain's control loop is however bounded to the rate of visual information acquisition; say 30 frames per second using common video capture cards.

So although the controller is capable of producing a continuous motion, motor commands are themselves discrete and step-like. Thus, if the controller was perfect and it could track the desired input commands with zero or a very small error, it would produce a discrete motion too.

To overcome this difficulty, a filter can be added in the controller to smooth the flow of desired positions[*]. Since our controller is digital, designing such a filter is a very straightforward. When writing this document, this feature has not been implemented yet, but it has been studied and evaluated and planed to be added to controller's program soon.

The most common filter responses are Butterworth, Chebyshev and Bessel types [Chen 86]. Butterworth ensures a flat response in the pass band and an adequate rate of rolloff. A good "all rounder," the Butterworth filter is simple to understand and suitable for applications without sharp transitions. Step inputs, however, result in overshoot with a little ringing.

---

[*] It is true that the controller is digital and it can not produce a continuous output. But it can convert sharp discrete transitions to smooth discrete transitions. For example, time series (1,4,4,4) may be replaced by (1,2,3,4).

The Chebyshev gives a much steeper roll-off, but pass-band ripple. It is superior for applications in which the pass-band includes only one frequency of interest (e.g., the derivation of a sine wave from a square wave, by filtering out the harmonics).

The Bessel filter gives a constant propagation delay across the input frequency spectrum. Therefore, applying a square wave (consisting of a fundamental and many harmonics) to the input of a Bessel filter yields an output square wave with no overshoot (all of the frequencies are delayed by the same amount). Other filters delay the harmonics by different amounts, resulting in an overshoot on the output waveform. Since our position commands are discrete and look like step functions, Bessel filter is the best choice.

Any of the mentioned filters can be characterized by its transfer function. The order of a filter is the order of its transfer function's denominator represented in complex frequency (Laplace) domain. A higher roll-off rate is achieved with higher order filters. The drawback is higher cost. A second order filter is satisfactory for our application.

$$H(s) = \frac{K}{(a/fc^2)s^2 + (b/fc)s + 1} \tag{3.3}$$

Equation 3.3 denotes the general form of second order filters. We will assume constant gain k is equal to unity, because no amplification or attenuation in steady state response is needed. Depending on values for coefficients *a* and *b*, different filters are obtained. Table 3.3 contains coefficients required for creating the mentioned second order low-pass filters [Karki 00].

Table 3-3 Coefficients of Second Order Filters

	a	**b**
Butterworth	1	1.414
Bessel	0.616	1.360
Chebyshev	1.4124	0.9107

Equation (3.3) represents a continuous-time transfer function, but our controller is digital and restricted to discrete-time functions. Thus, (3.3) must be approximated by a discrete-time model. There are well-known transforms for such an approximation. The simplest one is called backward difference shown in equation (3.4)

$$s = \frac{1 - z^{-1}}{T} \tag{3.4}$$

Where T denotes sampling period. A more efficient transform called bilinear transform is written in (3.5).

$$s = \frac{2(1 - z^{-1})}{T(1 + z^{-1})} \tag{3.5}$$

Using (3.5) and assuming and then (3.3) $B = b/fc$ can be rewritten as:

$$y[n] = \frac{(8A - 2T^2)y[n-1] + (2BT - T^2 - 4A)y[n-2] + KT^2(x[n] + 2x[n-1] + x[n-2])}{T^2 + 2BT + 4A} \quad (3.6)$$

Figure (3.11) depicts the simulated response of three second-order low-pass filters to a step-based input. In this figure f_c and T are assumed to be 1 and 0.01 respectively.

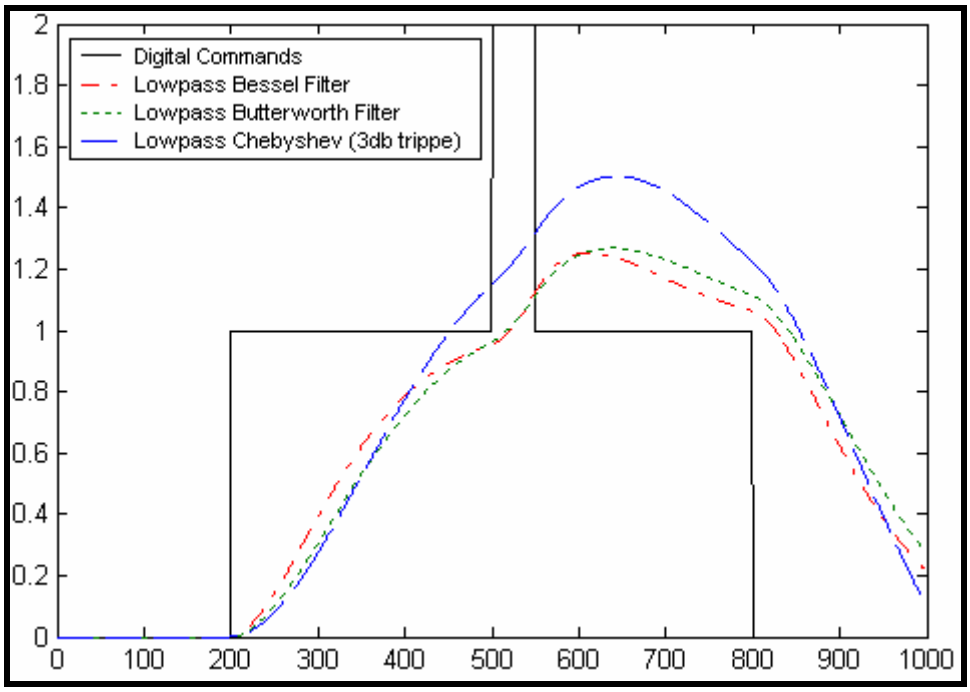


Figure 3.11 Response of second order filters to step-like input

3.5 Implementation Notes

The control program was written in c language and compiled to 89C51 machine language using a cross compiler running on PC. The resultant code was then programmed into the flash ROM memory of the microcontroller.

Frequency divider, address decoder and PWM generators were designed in a software environment that not only allows simulation of design, but also its optimization. The final design was converted to a netlist and eventually to an object code to be used by CPLD programming device.

The schematic of the whole design with details required for implementation is available in Appendix A. By placing a crystal between pins *XTAL1* and *XTAL2* of the microcontroller, an on-chip oscillator is obtained. We used a 12MHz crystal to provide the clock frequency of the whole circuit except ADC, which uses a reduced clock frequency.

The reset pin of microcontroller is connected to a resistor and a capacitor (see the schematic) to provide a power-on reset mechanism. In fact, when the controller is switched on, capacitor has no charge, so its terminals are short and Reset gets connected to V_{cc}.

After a short time, the capacitor gets charged and becomes equivalent to an open-circuit. Thus, Reset gets disconnected from V_{cc} this time. So a mechanism for activating Reset line when powering-on the system has been obtained. A Reset key may also be added for manual resets.

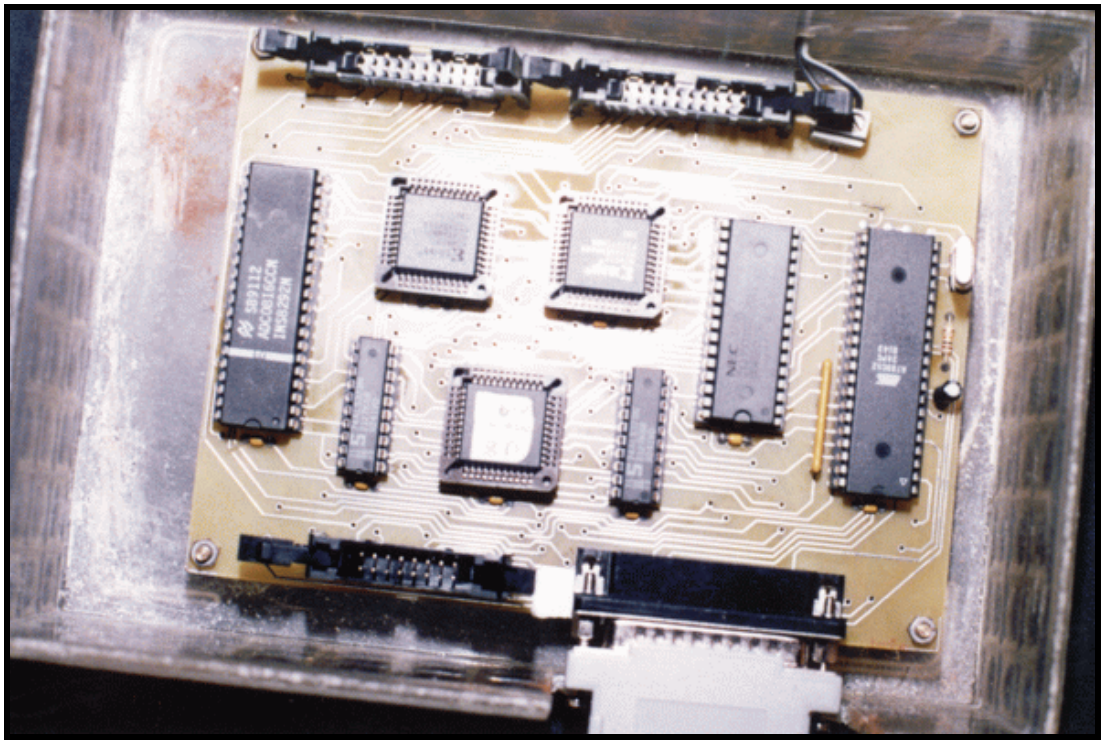


Figure 3.12 Our digital controller board

The order of turning on devices is important. For example if the amplifier is turned on when control loop is not started yet, undesired motion in motors might arise. In the right order, PC must come up first with brain program being executed. Brain clears parallel port and then asks user to turn on the controller.

Brain must be notified (say by pressing a key) when controller is switched on. Then, it uploads gain parameters to the controller. When upload is done, it informs you that you can turn on amplifier safely.

Since the logic circuit is very sensitive to source voltage, a high precision power-supply must be used. Employing a PC power-supply has a number of advantages. First of all it satisfies the mentioned requirement. In addition it is inexpensive and widely available.

We used an ATX power supply. It is capable to be switched on or off by main board of PC, so we had to manually resemble the turn on command of main board. Pin 14 of the 20-pin connector of ATX power-supply is an active low line termed *Power Supply On*. This line must be connected to a ground pin such as pin 13.

Printing controller circuit on the board cost 32,600 toomans or equivalently 40.75$. The final controller board is shown in Figure (3.12). The PCB of the controller is available in Appendix A. The pinouts of its connectors can also be found in Appendix B.

Chapter 4

Amplifier

Usually the current drawn by electrical motors is many times larger than what can pass through a microprocessor based circuit. So directly connecting PWM outputs generated by CPLD's to motors will damage the logic circuit. An amplifier must be placed between PWM generator and motors to provide necessary current for driving the motors.

A common amplifier used in motor driving applications is an H-Bridge. An H-Bridge is not only an amplifier, but also a polarity selector for directional change in motors. Indeed, if we wanted to use simple amplifiers such as a push-pull pair of transistors, we would require two power sources with a common ground to provide both positive and negative voltages.

There are various H-Bridge IC's available in industrial countries. But in Iran's market I could find just a few types of them. Unfortunately the IC's that I could find such as L293E were capable of delivering output currents to 1 Amp, which was too low for some motors such as those in the neck of the robot, especially at the start of rotation. Therefore I had to build my own H-Bridge circuit from simple components that I could find in our market such as transistors and resistors.

4.1 H-Bridge Constitutes

An H-Bridge can be built from switches (such as relays) for producing on/off outputs or continuous amplifiers such as transistors for producing both continuous and on/off outputs. Since commands sent to the amplifier are in form of PWM, that is either on or off, we consider the switching case of an H-Bridge only.

Depending on the state of a switch, it connects/disconnects a power source to/from the load, assuming both the source and the switch can deal with the high power consumed by the load. The state of a switch is determined by a command line that is supposed to dissipate a little power.

We used transistors for switching task in our H-Bridge. They have numerous advantages over electro-mechanical components such as relays. The principal advantages are:

- Low cost
- Small size
- High Speed
- High sensitivity

There are two basic types of transistors: bipolar junction transistors (BJT) and field effect transistors (FET). For switching purpose, FET acts better than BJT because it has a less saturation voltage* and consequently generates less heat. It also has a higher power rating. It's not strange to hear that FETs are more expensive than BJTs. Since each H-bridge requires at least four transistors, and there are a number of motors in robot's face, the overall cost is considerable. I decided to use power BJT's with heat-sinks mounted on them.

* Actual transistors used for switching are not identical to ideal switches. In an ideal switch, the electrical resistance of the switch is zero, so the whole voltage of the source will drop on the load. In transistors, there is always a small resistance when the switch is closed, which causes a drop of voltage across the transistor itself. This voltage is called saturation voltage, and it wastes a part of the source voltage.

4.2 Analysis of a simple H-Bridge

The simplest form of an H-Bridge is comprised of four switching/amplifying elements. In the previous section we described why we applied BJTs in our H-Bridge. There are two types of BJTs, PNP and NPN. Depending on what combination to use, there are two kinds of four-transistor H-Bridges. One uses four NPN transistors and the other one uses two NPN and two PNP transistors.

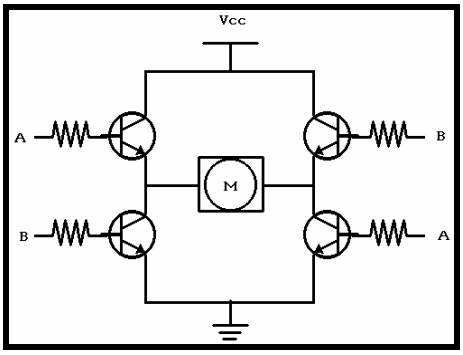


Figure 4.1 Four-NPN H-Bridge

Since understanding NPN transistors in this circuit may be easier than PNP transistors, we explain how an H-Bridge works based on a four-NPN H-Bridge (Figure (4.1)). In the PNP/NPN type, nothing changes, but two upper transistors are replaced by PNP having their emitters connected to Vcc and collectors to the motor.

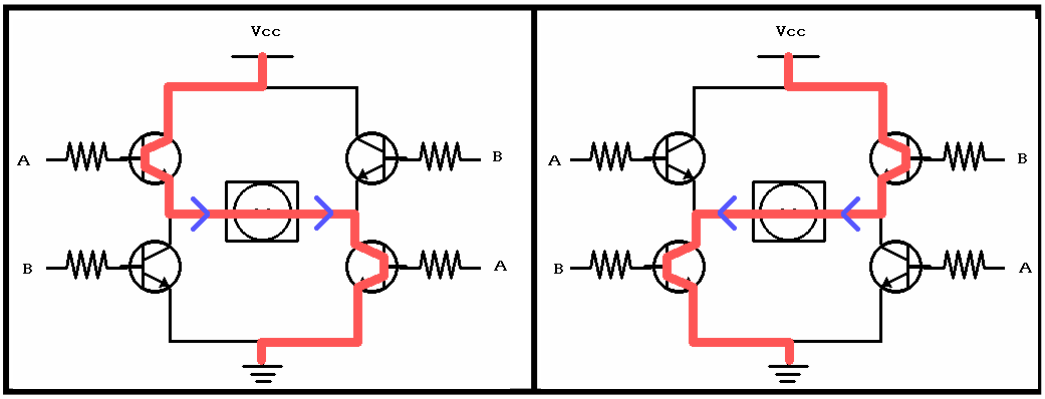


Figure 4.2 Switching Transistors and Current Flow

According to Figure (4.2), when *A* is high and *B* is low the upper left and lower right transistors are switched on while two other ones are off. In this case a current passes through the motor in the direction shown in the left half of Figure (4.2). Similarly, having *A* low and *B* high causes the current to pass through the motor in the opposite direction as shown in the right half of Figure (4.2). This feature allows direction control of motors.

When *A* and *B* are low, no transistor conducts and there will be no path from V_{cc} to ground. When *A* and *B* are both high, a short circuit occurs because it makes a direct path from V_{cc} to ground. The latter case may damage transistors and must be prevented. Possible states and their outcomes can be summarized as in table 4.1

Table 4-1 The effect of different switch states

A	B	Out
L	L	Off
H	L	CW Rotation
L	H	CCW Rotation
H	H	Short Circuit

Since transistors are supposed to be used as switches, they must be saturated when they are on. This can be achieved by appropriate choice of Base Resistors. One can find the suitable value of a resistor by starting from large values and measuring the voltage dropped across collector and emitter of the transistor. Gradual decrement of resistor's value decreases the voltage across collector and emitter too. This relation is true until reaching saturation point where decrement of resistor's value will no longer affect in collector-emitter voltage.

A disadvantage of a four-NPN H-Bridge is the equality between stimulus (control) voltage and output voltage. So control voltage bounds the voltage applied to the motors. This is troublesome when the control voltage is lower than nominal voltage of motors. In our project, we faced with this problem. TTL logic uses 5 volts and the output of CPLD's is about 3 volts while some of our motors needed 12 volts.

The NPN/PNP type eliminates this restriction. We used this type of H-Bridge in conjunction with an optical isolator. We will discuss about the complete design after a brief introduction to optical isolator.

4.3 Optical Isolator

Amplifier and motors deal with large voltages and currents whereas microprocessor and logic circuit are comprised of low power elements. A malfunction in motor or amplifier side may break down the whole logic circuit because of the electrical path existing between them. An easy and safe way to protect controller from high power side is to electrically isolate these parts.

A pair of optical devices, an emitter and a detector then connects these parts. High and low voltages turn the emitter on or off. Detector receives these optical signals and converts them back to electrical high or low signals.

An optical isolator usually adopts an infrared light-emitting diode as the light source and a photo transistor whose base is stimulated by light as the detector. This pair is protected in a package such that other lights in the environment do not interfere with emitter's light. This can be seen in Figure (4.3).

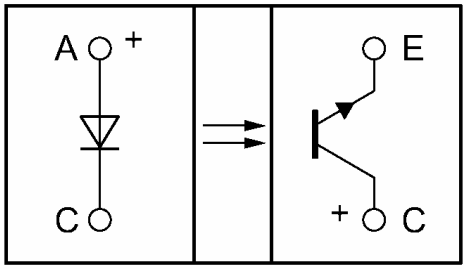


Figure 4.3 A typical optical isolator

Important parameters of an optical isolator for our application are:

- Forward voltage of the diode
- Maximum allowed current of the diode
- Collector-Emitter break down voltage (VCE max)
- Rise time
- Fall time

A resistor must be coupled in series with the diode to limit its current and keep it below the maximum allowed current. The value of this resistor can be computed as follows:

$$\begin{aligned} & I \le I_{\max} \Rightarrow V_R / R \le I_{\max} \\ & \Rightarrow (V_{control} - V_{forward}) / R \le I_{\max} \\ & \Rightarrow R \ge (V_{control} - V_{forward}) / I_{\max} \end{aligned} \tag{4.1}$$

Given forward voltage of the diode and its maximum allowed current from optical isolator's datasheet and maximum control voltage from the voltage of high state in the logic circuit, a lower bound for the limiting resistor can be obtained.

Since the detector is connected to the high power side of the circuit, care must be taken about the voltage that drops across collector-emitter of the detector; it should not exceed the collector-emitter break down voltage.

Although an ideal switching operation takes no time, in practice this transition occurs continuously and takes a short time. The time required for a transition from low to high is called rise time and the time required for transition from high to low is called fall time.

An optical isolator has its own rise and fall time independent of how fast the switching occurs in the input signal. Taking input pluses with a higher frequency than what the optical-isolator can cope with, results in a distortion in the output signal. These parameters can be roughly chosen according to the following relation:

$$\begin{cases} t_r \le 1/(4 f_{CLK}) \\ tf \le 1/(4 f_{CLK}) \end{cases} \tag{4.2}$$

Where f_{CLK} is the clock frequency for generating PWM pulses. Remember from chapter three that this frequency is not identical to the frequency of PWM pulses. In fact, the period of f_{CLK} determines the shortest possible width of a pulse in the PWM signal.

According to data sheet of our CPLD's, their high state voltage varies from 2.4 to 5 volts, which is a large range. So I had to find it empirically after completely building the controller. I measured PWM voltages coming out from CPLD's when pulses were full-scale. The measured value was about 3.7 volts, which is also the average of maximum and minimum voltages of high state in the CPLD's.

I used a set of TLP521-4 chips, each of which contained four optical isolators. Typical forward voltage for their input diodes was 1.15 volts with a maximum current of 50mA. Assuming maximum value of control voltages coming from CPLD's to be 3.75 volts as mentioned above, the adequate value for limiting current resistor was obtained to be 260Ω.

Collector-Emitter break down voltage of TLP521-4 is 55 volts, which is much less than the voltage of our power supply in the high-power side. The power supply was chosen so, because we did not have any motor with a nominal voltage above 12 volts.

The clock frequency used in the controller to generate PWM pulses was 500khz. From inequality 4.2, maximum values of *Rise Time* and *Fall Time* were obtained to be 0.5μs. *Rise Time* and *Fall Time* of TLP521-4 is about 3μs. So very narrow pulses may be distorted in our system.

But we do not expect an exact reconstruction of pulses. Further more, very narrow pulses are less likely to occur, considering possible all pulse widths. So distortion effect is negligible. In practice too, we observed a smooth and continuous rotation of motors.

4.4 The final design

The schematic of the final design is shown in Figure (4.4). *Q3* to *Q6* are the main transistors of the H-Bridge in a complementary fashion using NPN and PNP transistors. *Q4* and *Q6* were realized by TIP122. TIP127 transistors realized *Q3* and *Q5*. These are high power BJT Darlington transistors.

Since the base current required to saturate these transistors is large with respect to what can be tolerated by transistors of optical isolators *O1* and *O2*, supplementary transistors *Q1* and *Q2* were added. These transistors can be any general-purpose PNP amplifier such as 2N3906.

O1 and *O2* were realized by half of a TLP521-4, because each chip of this type has four optical isolators inside. *R7* and *R8* limit the current passing through diodes of *O1* and *O2*. Using relation (4.1) and given parameters of *O1* and *O2*, *R7* and *R8* were chosen to be 240Ω.

R2 and *R3* are to protect the system against short circuit. Short circuit may happen when any of transistors in *O1* or *O2* conducts. It is easier to obtain appropriate value for these resistors empirically instead of computational methods. A value of 100KΩ for both gave rise to acceptable results.

R1 and *R4* limit the current passing through the base of *Q1* and *Q2*. Since these resistors determine the base current together with *R2* and *R3*, and the latter pair was obtained empirically, *R1* and *R4* were determined by experiment too. A value of 270Ω for both was satisfactory.

R5 and *R6* limit the current passing through the base of *Q4* and *Q6*. A capacitor *C1* is optionally coupled in parallel with the motor to reduce its back-emf effect and impulsive behavior. Although this capacitor is brought in our design, we have not yet used it in our implementation.

D1 to *D4* are clamping diodes to prevent back-emf current generated by the motor to hurt transistors by a reverse bias. Since TIP122 and TIP127 have built-in clamping diodes, they were not actually used in our implementation.

Parts and their values can be summarized in table 4.2

Table 4-2 Parts and their values

R1	R2	R3	R4	R5	R6	R7	R8	O1	O2
270Ω	100KΩ	100KΩ	270Ω	180Ω	180Ω	240Ω	240Ω	TLP521	TLP521

Q1	Q2	Q3	Q4	Q5	Q6	D1	D2	D3	D4	C1
2N3906	2N3906	TIP127	TIP122	TIP127	TIP122	-	-	-	-	-

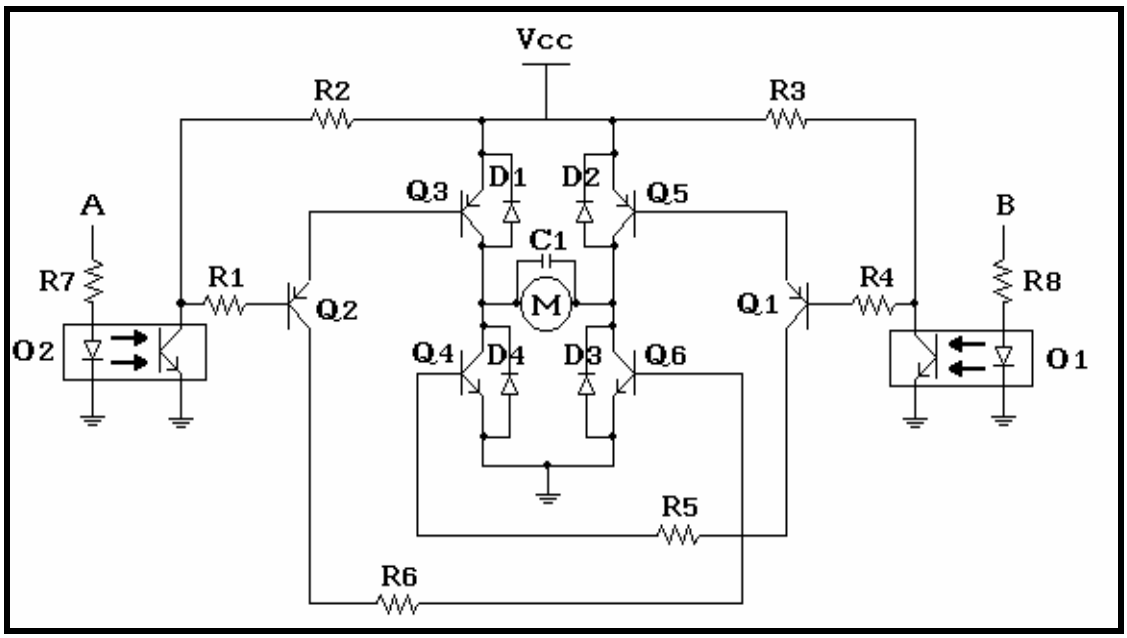


Figure 4.4 Schematic of the final H-Bridge

To evaluate our design we first simulated the circuit of Figure (4.4). In simulation, optical isolators were replaced by simple NPN transistors (eliminating diodes and optical channels) to investigate the behavior of this circuit in all possible voltages.

V_{cc} was a 10-volt power supply. Terminals *A* and *B* were connected to sinusoidal sources with frequency of 1Hz. Two sinusoids relatively had a 180-degree delay. The amplitude of the sinusoids was chosen to be half of the power supply's voltage. Simulation result is shown in Figure (4.5).

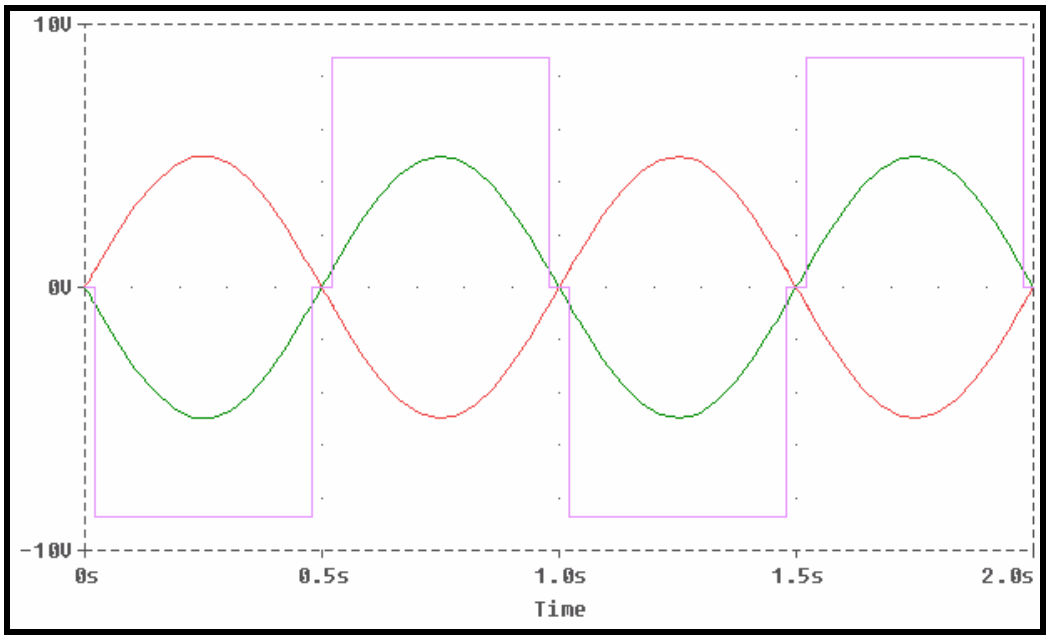


Figure 4.5 Simulation result of the final H-Bridge design

It can be seen that as soon as one of the inputs reaches a voltage above a small threshold, the output switches to V_{cc} until the voltage drops down below the threshold again. Negative voltages that cause a reverse bias of transistors have no effect on the output. Due to the saturation voltage in transistors, when output is high, it is a bit less than V_{cc}.

TIP122 and TIP127 have a large saturation voltage. For example, when collector current (which will eventually pass through the motor) is about 3A, saturation voltage will be about 2 volts per transistor. Since at any direction, two complementary TIP's surround the motor, totally 4 volts will drop on transistors when conducting.

This is a major drawback of applying BJT's as switches as mentioned earlier in section 4.1. To cope with this difficulty, one can choose a power supply that is about 4 or 5 volts above the maximum required voltage by motors.

4.5 Implementation

The physical circuit board is nothing but implementation of the schematic in Figure (4.4), and duplicating it as many as desired. We planned to build a 16-motor amplifier, but due to the large size of TIP transistors, to keep the size of the board reasonable, we eventually decided to build a 12-motor board.

The power supply used for high-power side of the circuit can be any general-purpose power supply able to provide the necessary voltage and current for motors. If a power supply with the required current is not available, one can separate (V_{cc}, *GND)* of each H-bridge when duplicating them, and use a low-current power supply for each H-Bridge.

The Printed Circuit Board of our amplifier is available in appendix A. The appearance of the final board is as shown in Figure (4.6)

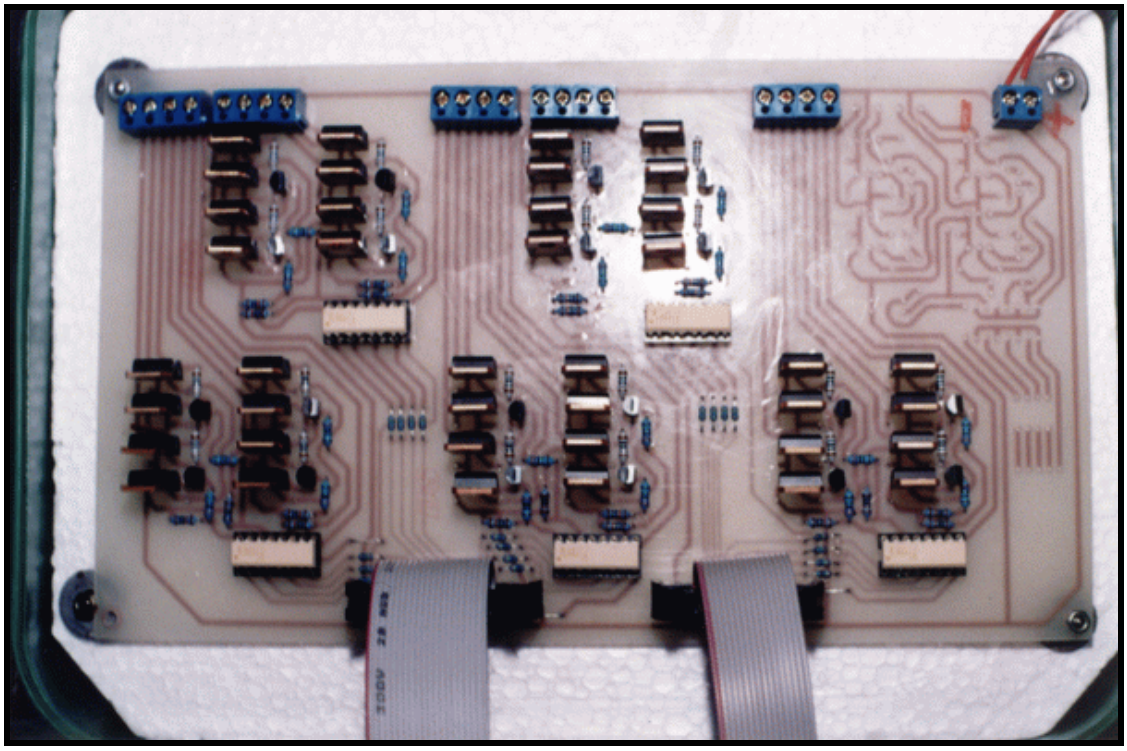


Figure 4.6 Implemented Amplifier Circuit

Chapter 5

The Brain

Previous chapters dealt with Aryan's physical parts. Here, we are going to show how we can achieve intelligence from many little modules, each non-intelligent itself. We will do this by developing a control system, comprised of various interacting modules coordinating all parts of Aryan and making it a whole. We will call this control system "brain". The brain helps Aryan to act autonomously. It is realized by a program running on a standard PC with a 200MHz Intel™ Pentium processor.

Microsoft Windows™ and Linux are two common Operating Systems (OS) these days for PCs. Linux was preferred for our work because it allows user to configure and optimize kernel according to his needs, in our case allocating resources for real-time image acquisition and processing. Eventually the modified kernel should be recompiled. In addition, Linux kernel provides an easy to use and powerful interface for transparently accessing capture cards (See 5.3 for details).

I have written brain's program in c language. Major reasons for choice of c as the preferred programming language were flexibility and near real-time execution. We also tried to take advantage of programming tips in c for speeding up execution time. The user interface of brain software is written using Mesa3D library for displaying video in "X Window" environment. A screen shot of the desktop is shown in Figure (5.1). The complete source code of brain program is available in Appendix C.

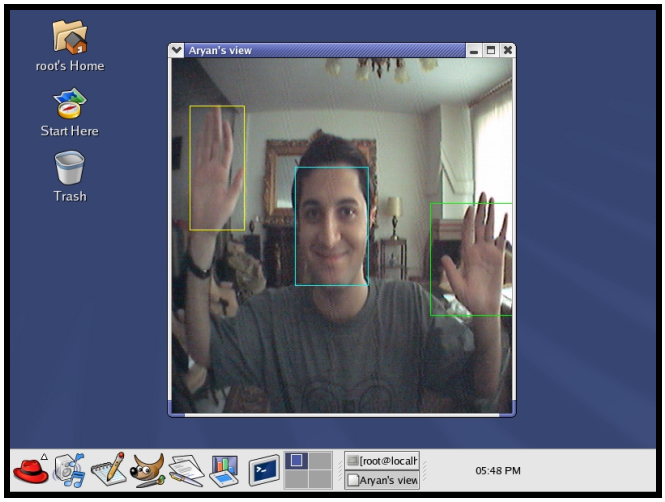


Figure 5.1 Brain Software running on Red Hat Desktop

5.1 Related Works

Almost any interactive robot requires software that models some aspect of brain to some extent. Here we shortly review some related works on interactive robots from their cerebral aspect. Recall that the mechanical and expressive abilities of these robots (with pictures showing their physical appearance) have already been discussed in chapter one.

Sparky, developed by Scheef *et al*. [Scheeff 00] cannot perceive people and is not autonomous, but teleoperated. It can express nine different emotional states: neutral, happy, sad, angry, surprised, fearful, inquisitive, nervous and sleepy. It's brain only tries to generate natural and life-like motions, given teleoperated commands. Motion generation in Sparky is based on Ken Perlin's work on "perlin noise".

eMuu is a robot developed by Bartneck to function as the interface between the user and intelligent home [Bartneck 02]. eMuu reasons emotionally about certain tasks of home and expresses its emotional state accordingly. It has no perception, so user has to instruct it through keyboard for example. or and receives its instructions from the user, e.g. using a keyboard, for performing certain tasks of home. eMuu is able to express three emotions happiness, sadness and anger.

The eMuu software consists of three components. The game engine, which implements the negotiation task, and the character engine, which controls the behavior of the character, are running on a PC using JAVA. The emotion engine, which controls the emotional state and the facial expression, is based on the OCC model [Ortony 88].

The only task of Feelix, a robot built by Canamero and Fredslund [Canamero 00], is emotional expression for the purpose of social interaction with humans. It uses a combination of two different emotion representations, discrete basic emotions and continuous dimensions. In Fellix, basic emotions are placed in an emotional space defined by arousal and valence dimensions.

Feelix only perceives tactile simulations. Such a cut down perceptual ability highly simplifies its brain design. For instance its perceived signals are immediately converted to emotions without need for decomposing perception to different levels of abstraction. On the contrary, in complex sensory mechanisms such as active vision, level decomposition is usually expected. In addition, since Feelix' perceptual data has less density than vision, the role of attention as an important part of brain for resource management is diminished.

Minerva [Schulte 99], which was developed by Thrun *et al.*, is an interactive tour-guide robot that displays four basic expressions. Minerva has for different moods: happy, neutral, sad and angry, and switches between these states in continuously by the mentioned order. A transition toward happy state is made when the robot can freely move and a transition toward sad state is made when people block its way. A state machine has modeled this dynamic. The emotional state of the robot is directly mapped to facial expressions.

In addition, Minerva learns how to attract people. It issues a series of actions (facial expression, gaze direction and sound type) each one taking 10 seconds. Then it evaluates them based on a reinforcement signal. A reward indicates the relative success of the behavior. This function was defined such that an increase in closeness and density of people around the robot was rewarded and a decrease was penalized.

Fumio Hara and his partners have been developing realistic animated talking heads. Recently, they have focused on teaching their robotic heads by natural instructions issued from humans. The robot learning is based on Q-Learning algorithm with delayed reward. Their goal is that the robot autonomously understands the values of human partner's states and then organizes sensory-motor coordination using such values. In their experiments [Iida 99], once the Face Robot recognizes that the human position has been changed, the Face Robot proceeds its learning step to the next one automatically.

Comparing with robots that were reviewed so far, Kismet has a very sophisticated brain consisting of various sub-systems: high-level and low-level vision, attention, motivation, behavior and motor. Subsystems interact to enable the robot to behave coherently and effectively. Although these subsystems differ in function, they all follow Synthetic Nervous System (SNS) mechanism [Breazeal 00].

Kismet specialized in para-linguistic communication, both in expressive and affective terms. It can sense human affect through vision and sound, and express itself with emotional posturing. Kismet uses a feedback loop of affective perception and affective behavior (in motion and speech) to explore social exchanges with humans [Breazeal 02*]. It achieves the fundamentals for face-to-face communication (proto-dialogs) that capture the dynamics of conversational turn taking, sharing attention and engagement [Breazeal 02#].

5.2 Brain Modules

We propose a simple modular brain architecture for our robot that is lightweight and easy to implement. It can cover essential abilities that our system needs for exhibiting simple emotional and interactive behaviors. A block-diagram representation of this architecture is shown in Figure (5.2).

Briefly, visual sensors capture visual information and represent them in an appropriate form for digital processing. Low level-vision tracks moving and skin-toned regions. Attention module directs limited and computationally costly resources toward the most motivating item among items that are being tracked. Motivation module determines which item should become the focus of attention. Attention system also controls activity of high-level vision.

High-level vision is comprised of neural networks trained to detect faces, hands and facial features. Depending on the type and situation of the motivating item, an emotion is triggered from a set of discrete emotions. Emotion module is modeled by a state machine and a counter for determining decay of emotion. These states are hardwired to robot's facial features for displaying appropriate facial expressions.

First phase of smooth gaze adjustment in Aryan is achieved by the observed scene through the middle camera. Given a rough distance of user from the robot, a bit of trigonometric manipulation provides an approximated place for the eyes. This situation is just a primary guess for the correct gaze direction. The accurate situation can be computed after that by exploiting depth information extracted from stereo pairs. Moreover, when Aryan sees no person around it, it becomes performs involuntary saccades (rapid eye movements), hoping to find a mate in its new sight.

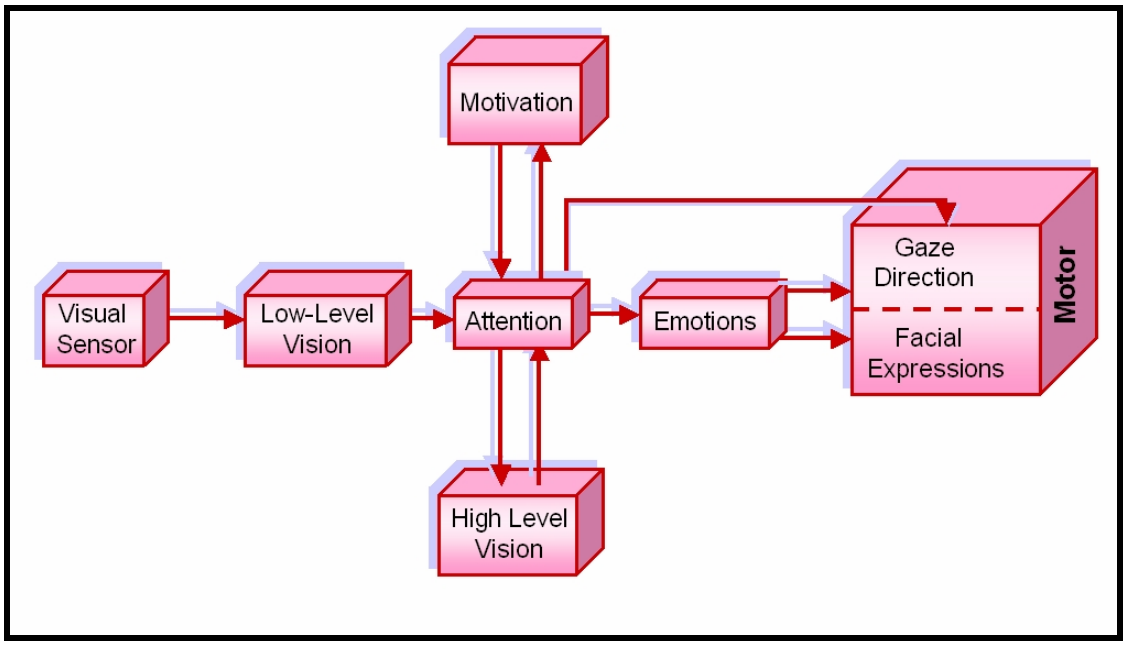


Figure 5.2 Brain Architecture

5.3 Visual Sensor

In chapter one, we saw how perceptual data is acquired from three cameras mounted on robot's head. Since the cameras produce analog outputs, a hardware device is required to convert them to digital form to be used by computer. This device is called "video capture card" or "frame grabber". A video capture card provides a sequence of digital images to be processed. We used a "FlyVideo EZ II" capture card for the middle* (wide field of view) camera. This card was manufactured by LifeView ™ and cost 55,000 Toomans, equivalently 67.85$.

In ideal conditions, a capture card is able to operate in real-time, e.g. 30 frames per second. However, this rate is reduced in practice due to the CPU and bus speed limitations, and also execution time of image processing algorithms. Using our old PC whose specification was described in the introductory part of this chapter, we obtained 5 frames per second. We believe that this rate can be improved at least twice faster, by using an up-to-date processor such as a Pentium IIIX.

Linux kernel provides user with a practical and powerful interface called "Video for Linux" or "V4L" for short, to transparently access a large group of capture cards, including FlyVideo. Plugging multiple cards and reading them simultaneously is also possible with this interface. This is an important factor for our later work on stereovision, where multiple views are required at the same time.

When working with a capture card, first it must be initialized with the right capture parameters. Using V4L functions, we easily set parameters such as resolution and color mode. Image resolution causes a trade-off between processing time and accuracy. We set image resolution to 256x256. This choice could keep a reasonable balance between mentioned factors.

For color mode, we chose 24-bit RGB image due to our previous experience with this color representation. We will see in section 5.4.2 that we require two successive images for motion detection. For motion computation, we subtracted two successive images. Therefore, a pair of buffers was allocated for capturing task, switching between buffers alternatively for storing the most recent captured image.

---

* Although the motion of cameras located in the eyes are controlled, but their acquired images is not used in this phase of project. They can be used later in stereovision tasks.

5.4 Low-Level Vision

Aryan is designed to interact with humans. Therefore, it must be able to detect human's limbs that are instructive for gesture and emotion expression, such as hand and face. It must also be able to track these limbs for action recognition and gaze adjustment. The acquired images from visual sensor contain a large amount of data. Searching the whole data space for complex patterns such as face and facial features is very time-consuming. Since our system is supposed to work in real-time, we have to find an efficient way for processing this data.

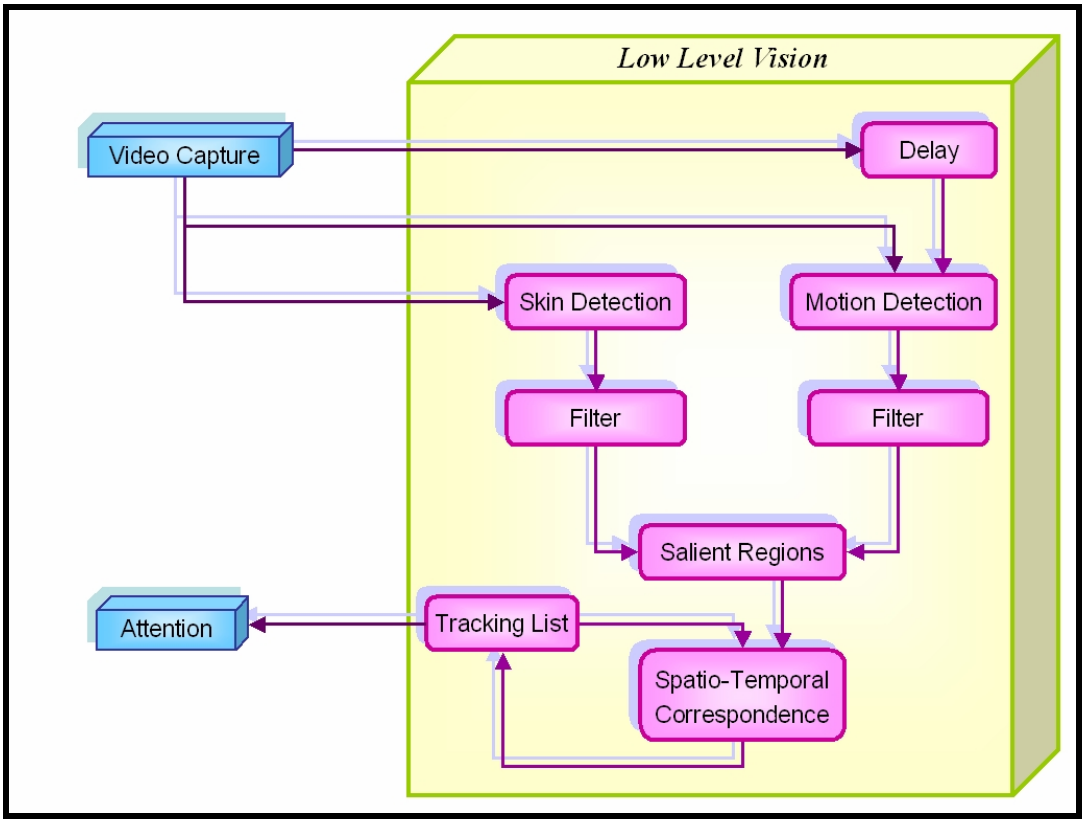


Figure 5.3 Low-Level Vision

This can be achieved by decomposing visual system into two levels of abstraction, a basic but fast module and a sophisticated but slow module. These two modules are called low-level and high-level vision respectively. The ultimate goal of low-level vision module is to detect and track regions that are likely (not certainly) to be a face or hand based on a rough inspection. Thus, the data space for high-level vision is highly reduced and it can perform more precise inspection on the remained data to achieve recognition.

The low-level vision is itself comprised of sub-modules as depicted in Figure (5.3). Briefly; the input of this module is a sequence of images acquired by the visual sensor. Current image is passed to skin detection sub-module to generate a binary image, showing skin or non-skin situation of each pixel. Since it is hard to extract motion from a still image, we pass two successive frames to motion detection module. It gives another binary image determining regions where a change has been sensed.

Skin and motion images are then filtered to reduce noise effects. Next, they are fused to guess possible face or hand regions, termed "salient patches". A box is superimposed around each salient patch to ease computations. These boxes are compared with previously being tracked boxes and a correspondence is established between them. Finally, the list of tracking patches is updated. So the output of this module is a list of being tracked boxes containing salient patches.

In the following sections we will describe each module in more details.

5.4.1 Skin Detection

Using skin color as a feature for detecting face or hand has advantages such as fast processing and scale or orientation invariance. However tracking human faces using color has several problems. Different cameras produce significantly different color values even for the same person under the same lightning conditions. Human skin colors also differ from person to person.

Good news is although skin colors of different people appear to vary over a wide range; they differ much less in color than in brightness. In other words, skin color of different people are very close, they differ mainly in intensities [Yang 96]. Therefore color spaces, which separate color from intensity, are suitable for detecting skin color. They can also reduce the effect of shadows and some changes in intensity of the light source.

YUV is one of such color spaces used by Pentland *et al*. [Wren 97] in their person finder system. Raja [Raja 98] used HSV, another color space with separated intensity, for face and object tracking. A different color space with similar property is chromatic space. It is shown that distributions of skin colors of different people from different races are clustered in chromatic color space. Figure (5.4-*a* and 5.4-*b*) are color distributions of skin under different lightning conditions and Figure (5.4-*c*) is the color distribution of two persons skins [Yang 96].

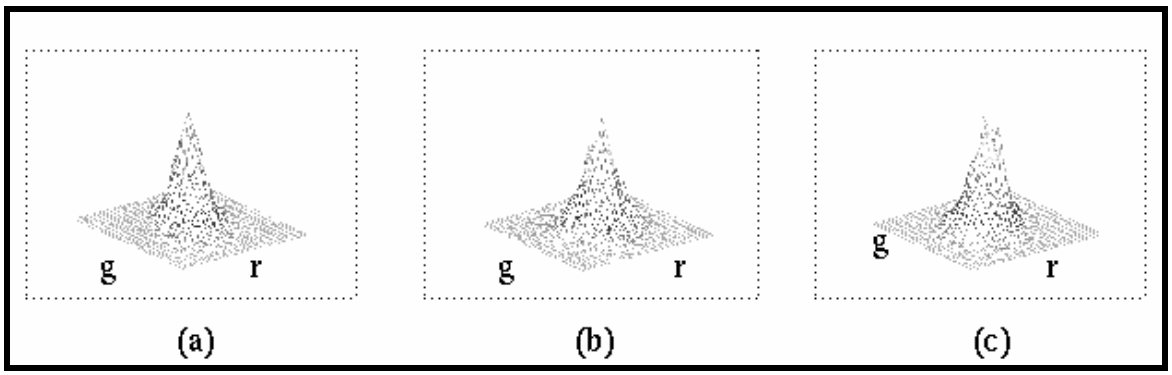


Figure 5.4 Skin-Color Distribution in Chromatic Space

We applied this color space because it can be computed very fast on the whole image. The components of this space are defined as below:

$$r = \frac{R}{R+G+B} \quad ; \quad g = \frac{G}{R+G+B} \tag{5.1}$$

Note that there are only two components in this space, because it intrinsically eliminates the independent component of intensity. In fact, r and g are red and green normalized by intensity. Therefore, intensity has no effect in this representation. This space has been successfully applied for face and lip tracking ([Yang 96], [Oliver 97]).

In Figure (5.4), it can be seen that human skin colors of different people under different lightning conditions follow a Gaussian-like distribution in the chromatic space. Therefore, we represented skin color distribution by a parametric Gaussian model $N(\boldsymbol{\mu},\boldsymbol{\Sigma})$:

$$f(x) = \frac{\exp(-(x-\mu)^T \Sigma^{-1}(x-\mu)/2)}{2\pi|\Sigma|} \tag{5.2}$$

Different cameras result in different parameters. Therefore, we computed the mean vector and covariance matrix of the model offline from several samples taken by our camera from different subjects. Since we eventually want to make a decision about skin and non-skin pixels by setting a threshold over skin probabilities, and also because logarithm is a monotonic function, we can take *Ln* from both sides. This helps to get rid of *Exp* function, which is computationally expensive. This has been in the following inequality (5.3):

$$\begin{aligned} &\ln(f(x)) > \ln(T) \\ &(x-\mu)^T \Sigma^{-1}(x-\mu) < -\frac{\ln(2\pi|\Sigma|) + \ln(T)}{2} \end{aligned} \tag{5.3}$$

The right hand side of (5.3) is another constant like T' and the left hand side is a parabolic function as follows:

$$\sigma_{g,g}(r-\mu_r)^2 + 2\sigma_{r,g}(rg - r\mu_g - g\mu_r + \mu_r\mu_g) + \sigma_{r,r}(g-\mu_g)^2 < T'' \tag{5.4}$$

Equation (5.4) is a bivariate parabolic decision surface. Skin pixels were considered as 2-tuplie random vectors and training pixels as different observations of these vectors. Parameters were computed using maximum likelihood estimation. The decision threshold was obtained empirically.

In experiments, we observed that a training set consisting of merely face samples couldn't model hand skin color properly. This means that face and hand are a bit different in color (hand looks more pink). Therefore we estimated parameters from a training set that contained both faces and hands. Chromatic components in low intensity (near singular) regions give poor results, so we did not involve them in skin/non-skin decision. Figure (5.5-a) and (5.5-b) show the original image and skin-toned patches respectively. For further improvement, we also passed the result through a median filter as shown in Figure (5.5-c).

(a) (b) (c)

Figure 5.5 Detected Skin-Toned Regions

It can be seen the detection result is imperfect, e.g. existence of undetected regions. This problem has two major sources. It may occur when background and skin colors are mixed due to the smallness rapid motion of a skin patch. This happened to the fingers in Figure (5.5). Another reason is the relative position of skin patch with respect to the light source. It influences the amount of reflected light from skin. Excessive light reflection diminishes skin color.

Unfortunately, we cannot conquer to these problems in this level, because pixel level operations are too local to take structure into the account. To overcome this problem, we will delete or merge some patches by simple heuristic rules. This is a matter of section 5.4.3.

5.4.2 Motion Detection

When a human interacts with another human or a robot, there is definitely motion. Several approaches have been proposed to detect motion. Cutler and Turk [Cutler 98] developed a motion tracking system based on optical flow. Ridder *et al*. [Ridder 95] used Kalman filter to classify foreground and background regions. Huwer and Niemann [Huwer 00] applied neural networks to this problem. Unfortunately the computational cost of these solutions is too high to be used in a complex real-time system. We must save CPU time for other modules as well as motion detection.

A naive but fast motion detection is to differentiate two successive images and threshold the result. Despite of its simplicity, it has been used as a significant part of complicated systems. McKenna *et al*. [McKenna 98] used it to track a group of people. Breig and Kohler [Breig 98] applied it as a part of their tracking module in ARGUS (a vision based system for human-computer interaction). Breazeal [Breazeal 99] used it in the vision system of her humanoid robotic face, Kismet.

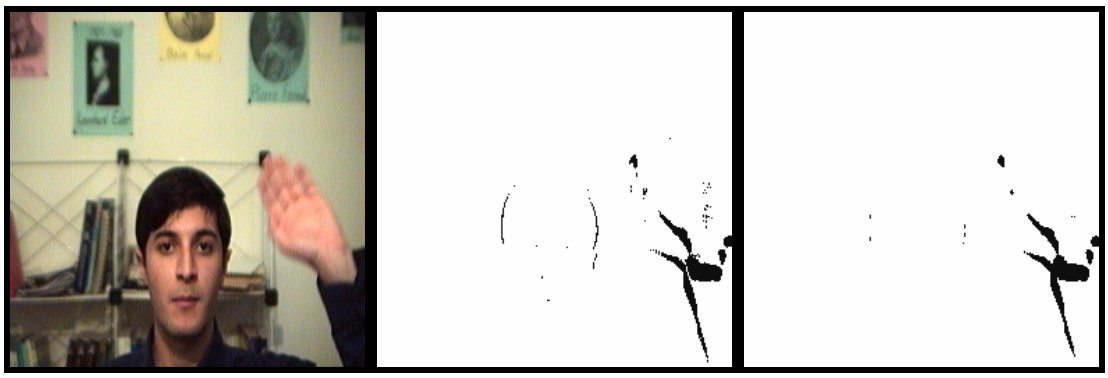
There are however two major problems with this method. The first one arises when the moving object is not textured enough. In this case, only its boundaries are detected not the whole moving patch. This is because differences are pixel-wise and too local to know what group of pixels belong to the same object. This is like the case when you look at a non-textured image through a very small hole. It is hard to say whether it is moving or not.

The second problem is wrongly detecting motion in parts of background. This can happen for the regions that were hidden in the previous frame but are visible in the current one. In fact, image differencing detects changes, not motion. Just like skin detector, we cannot and do not expect more accuracy from this local operation. In the next section we will show how to fuse skin and motion maps to obtain a reliable saliency map.

To measure the amount of change in a certain location we used maximum difference of color components as follows:

$$d = Max\left(|r_2 - r_1|, |g_2 - g_1|, |b_2 - b_1|\right) \tag{5.5}$$

This result along with a threshold value helps to know where motion has occurred. The threshold must be chosen carefully because high threshold values prevent contaminate detecting motion of non-textured objects. We obtained the best value empirically. Figure (5.6-b) shows the detected motion in Figure (5.6-a) and its previous frame. It can be seen that the result is noisy must be filtered. Since the result of detection is a binary map, median filter can be effectively applied to reduce noise effects. Figure (5.6-c) shows the filtered result.



(a) **(b)** **(c)**

Figure 5.6 Motion Detection by Image Differencing

Our vision system is active, i.e. its sensors can shift and adjust direction of gaze. This ego-motion of sensors causes a blur and even drastic changes in the whole image. In this case it becomes impractical to take apart absolute motion from differenced image. To prevent from confusion, we add up pixel changes in the entire image to compute overall change. If the global change is above a threshold, the computed motion is invalid and this module gives no output.

5.4.3 Salient Boxes

We can now fuse the resulted skin and motion maps to obtain salient regions for attracting attention. Bottom-up approaches mostly construct a saliency map from feature maps (such as color channels, edge, orientation, etc) using a many-to-one mathematical mapping. This mapping can be either linear, i.e. a weighted sum of features maps [Breazeal 99], or non-linear, e.g. using a neural network [Baluja 95, Lee 98].

In our case, there are only two feature maps, skin and motion. Fusing these features for obtaining a saliency map using mentioned methods is sensible in presence of simple backgrounds. However, it may cause unwanted attention jumps over salient regions in crowded environments. For instance, if a museum tour-guide robot is interacting with someone, the person can be almost motionless, but people behind him may be walking quickly. Irrelevant people cause high response in both feature maps. Nonetheless, this should not disturb the interaction process.

Recalling that the major application domains of interactive robots are in possibly crowded environments, a robust saliency representation must be adopted. To achieve this we will make use of motion only for detection purpose. Once a moving skin-toned region has been detected, its motion status is no longer checked (during tracking). This means that feature map(s) are explored selectively by tracker (Section 5.4.5)

What attention module needs for performing selection, is not the exact feature maps, but a rough shape of salient regions along with a motion flag. Since human limbs may be motionless for sometime, motion cue is not always present for shape computation. On contrary, human color remains almost constant in any situation. Therefore we use skin map for determining spatial characteristics of salient regions. Making use of a motion flag instead of combining motion and skin maps, prevents from saliency corruption by motionless regions. This idea was implemented as follows:

- Contiguous skin pixels are computed in a bottom-up manner using blob-coloring algorithm. While coloring a blob, the shadow of each pixel in the difference image is also examined.

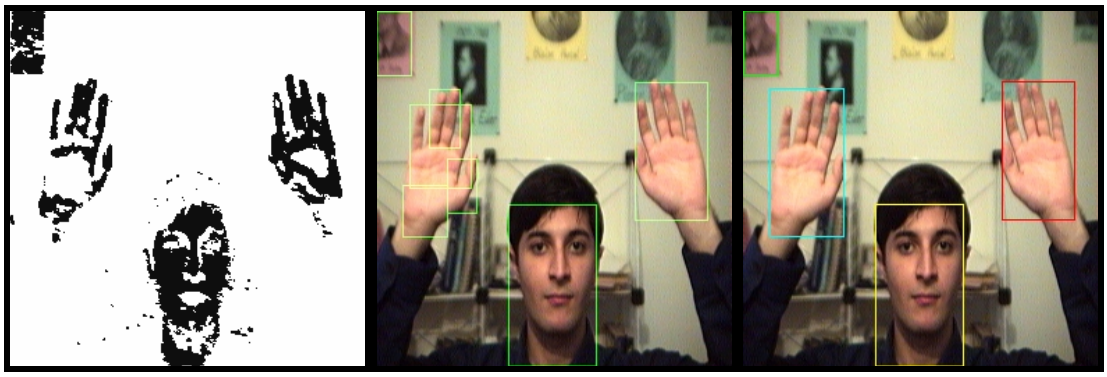
- If for a skin patch the number of its hits in the motion map exceeds a threshold we will set its motion flag to *true*. This threshold must be chosen such that it can deal with less textured skins.

- When all pixels of a blob are colored, a bounding box is superimposed about that blob. The parameters of this box are stored as a vector.

Actually, instead of generating a saliency map, only these parameters are used by attention module. The advantage of approximating each patch by a box is reducing computational complexity and improving real-time performance of system. Moving from pixel level to region level also provides a means for refining salient boxes according to high-level rules.

Applying simple heuristic rules can exclude wrongly detected boxes. For instance, an intuitive rule is to delete very small boxes. This can be achieved by computing the surface of a box and comparing it with a threshold. We found that a ratio of 1/400 between the surface of a box and screen's surface is an appropriate choice.

Another rule is needed to cope with missing pixels as shown in Figures (5.7-a) and (5.7-b). In the worst case, this problem occurs in a critical region that splits an actually single patch into smaller ones. In this case, none of the patches covers the whole face or hand as shown in Figure (5.7-c). Dilation could be a solution, but it would need to be applied iteratively to the whole image until achieving a satisfactory result. This would require a considerable computation time.



(a) (b) (c)

Figure 5.7 Box-Level Filtering

A less accurate but much faster approach is a box-level merge. We merge each two boxes that satisfy proximity or overlap conditions. Again, the combined region is represented by a box. This operation is repeated until no more boxes can be merged. Although checking overlap condition is trivial, proximity is a subjective issue. We quantified proximity by measuring the Euclidian distance between two boxes. We defined two boxes be close if the distance between them is below a threshold. Figure (5.7-a) shows the original image and Figures (5.7-b) and (5.7-c) show the results obtained with and without these rules.

5.4.4 Correspondence

So far we have detected salient boxes in the current frame. For tracking purpose, we must associate these items to previously tracked ones. Let's give a simple example; suppose that we have been tracking two boxes, namely *A* and *B*. Now we take a new frame and extract two salient boxes. How do we know which one is *A* and which one is *B*? The problem gets harder when *A* and *B* change over time (in terms of shape, lightning conditions, etc.).

The correspondence analysis is often supported by prediction. Based on previously detected items and possibly high level knowledge, the state of the items (appearance, position, etc.) in the next frame is predicted and compared (using some metric) with the states of items found in the actual image. More on the prediction will be discussed in section 5.4.5. Due to the real-time constraint of our robot, we used the center of each box as its state variable[*].

After establishing correspondence, parameters of tracked boxes are updated according to the latest observation. Therefore, tracking parameters influence the result of correspondence and correspondence influences tracking parameters. This tight loop can be seen in Figure (5.3).

---

[*] We will also keep track the age of each box to analyze its persistency situation. This increases robustness of the tracking system. Persistency will be discussed in this section.

We will talk about prediction in the next section. Here we assume that the next state of each box is already predicted and we are left to implement the correspondence process. To achieve this, first we place a search window around each predicted box. The size of this window is proportional to the predicted size (with a ratio larger than unity) of that box plus a scaled version of prediction error in previous frame. This means search window is enlarged when prediction error has been high in the previous frame. Eventually, tracked boxes are examined to check which one(s) fall inside or overlap with the window.

One may ask about the advantage of using such a search window about each predicted location. First it limits search scope to prevent from exploring less promising regions. If this constraint was not imposed, a tracked item whose actual box was missed (e.g. due to occlusion, bad lightning conditions, etc.), could takeover a salient box that belongs to another tracked item. Yet this scope reduction is not so small as the predicted size at predicted location of a tracked item. So small prediction errors do not cause missing a tracked item.

Occasionally it may happen that one salient box fall inside two or more search windows of tracked items. A common case occurs when items occlude each other in space. In this situation, the 2D projection of these items may overlap on the image plane while there were two tracked items for them in past frames. Another cause is a noise box that is wrongly detected due to the imperfectness of saliency detector.

In such cases, tracked boxes compete to win the common salient box. We must design this competition such that the most promising tracked item wins the competition. Fortunately it is not hard to recognize tracked noise items and decreasing their chance to win. Since noise usually varies over time, tracked items resulted by noise appear and disappear all the time at different locations.

According to the fact that tracked noise has a short life, keep tracking the age of boxes can be useful for recognizing noise. This can be achieved by allocating a counter to each tracked item and incrementing its value in each time-step. We call a tracked item to be persistent if its age exceeds a threshold, otherwise it is called feeble. Although nothing can be said about feeble items, persistent items are less likely to be noise.

The competition is implemented by incorporating a state machine in each salient box. Once a new frame is captured, and salient boxes are detected, their state machines start from an initial state. Each tracked item finds the nearest salient box that falls inside of its search window. Then the state of this box is updated according to persistence situation of that tracked item.

An action may take place between state transitions. This action deletes from the list of tracked boxes the previous tracked box assigned to the salient box and establishes a new correspondence between the salient box and the last explored tracked item. The state machine of a salient box is shown in Figure (5.8). When all tracked items are investigated, the final state of each salient box is read and interpreted as follows:

- **Start State**: No tracked item has been found for this salient box. So this is possibly a new item to be tracked. An entry in the tracking list is created for this item in Update Tracking List module (see next section).

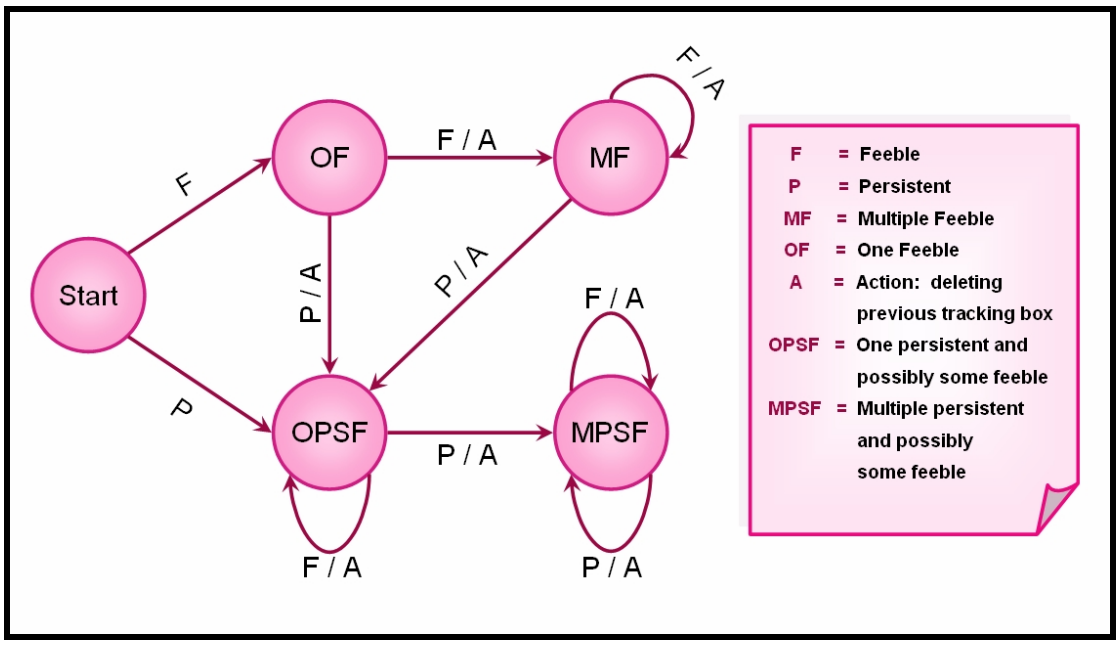


Figure 5.8 Persistency Detection

- **One Feeble**: The feeble tracked item has been the only option for this salient box. We optimistically assume this is not a noise patch, but a young accurate patch and establish correspondence.

- **One Persistent and possibly some Feeble**: The persistent box has higher priority over feeble item. Therefore, if there had been any feeble boxes, they would have been deleted during updates of the state machine and only this persistent box is left. Its correspondence with this salient box is established.

- **Multiple Feeble**: There has been a group of feeble items (with the same priority). So we can not exactly decide to which tracked item this salient box should be assigned. Therefore, all of these items are deleted to prevent from a wrong decision and tracking. In fact, all of these items have been already deleted during state transactions, except the last item that is deleted in this stage.

- **Multiple Persistent**: Similar to the previous case, all items are treated equally and no one is different. So all of them are deleted.

For instance, consider a salient box in a new frame. Assume that it is first assigned to a feeble tracked box *A*. As the loop of checking tracked boxes proceeds, another feeble tracked box *B* is also assigned to this salient box. At this step, when the state of this salient box is updated, the tracked box *A* is deleted from the list of tracked boxes. The loop continues until all tracked boxes are examined.

Note that deleting tracked items in confusing situations such as occlusion doesn't make a serious problem, because these boxes can be detected again when the occlusion is gone. Even if the situation is sustained for sometime, system detects the overlapped regions as a whole new item and tracks it. Since high-level vision module cannot detect this mixture neither as a face nor as a hand, system does not make any

mistake until when items pass over each other and occlusion is gone. In this case, one of the boxes is assigned to the old tracked item (depending on the order of examining salient boxes) and a new tracked item is created for the other box. Now in the next identification attempt items can be recognized (as face or hand) and tracking is continued as before.

5.4.5 Updating Tracking Parameters

In the previous section we got familiar with the role of predictor in a tracking system. Typically a process model of item's behavior is assumed for prediction purpose. The predictor loops anticipating the next state of a tracked item by observing its past behavior and updating its internal model parameters accordingly. Therefore, when the correspondence is established, we should feed the result to the predictor to update its internal parameters.

A popular predictor for tracking purpose is Kalman filter, which has been applied in tracking module of large projects such as LAFTER [Oliver 97] and ARGUS [Breig 98]. Kalman filter recursively updates its noisy state equation based on the current noisy measurement. This filter provides the optimal estimation in terms of mean-squared error. Kalman filter assumes noises are Gaussian white.

Nonetheless, there were difficulties for applying Kalman filter in our system. Its general problem is requiring an initial estimation for covariance matrices of process and measurement noises. This is not a trivial task, particularly for image-based tracking systems. Another problem was the real-time constraint for an interactive robot. Kalman filter requires several matrix operations, most notably multiplication and inverse, in each process cycle. This was too high for a complex system like ours that is implemented on a standard PC. Therefore, we had to employ a simpler approach to achieve real-time performance.

We used the nearest neighbor technique, i.e. assuming object of interest will appear in the same location as it is in the current frame. Although this postulation fails with rapid movements, it is acceptable for very smooth motions. To improve the result, we also incorporated prediction error for adjusting the size of search window as discussed before. Totally, we could obtain satisfactory results in the context of our application.

Besides predictor parameters, there are other tracking variables that should be updated here. One of these parameters is the age of each tracked item, which is incremented at this point. In the previous section we talked about the role of age factor in determining persistence or feebleness of items. We will also see how the age factor is used by attention module for achieving an effective use of high-level module.

Creating a new entry in tracking list for a new detected limb is also a task of this submodule. If the state machine of a salient box ends in its start state, it means the box could not be matched with any of the tracked boxes and a new entry must be created for it. Moreover, the motion flag of the salient box is checked here to be *true*. When both of these conditions are satisfied, the entry is created. Obviously, its age and type are initialized to zero and unknown respectively.

Since a limb may occasionally stay stationary during tracking, motion flag is only examined at the beginning for ensuring that a salient box really moves. Once it shows some signs of motion continuously for sometime, we give up its motion investigation. Therefore, the motion flag is checked when creating a new and as long as it is feeble. If a feeble box stops moving, it is considered as a wrong detection and its entry is deleted. However, when it becomes persistence, its motion status is no longer checked.

Another modification required in the tracking list status is when a salient box has ended in multiple-feeble or multiple-persistent states (and sometimes feeble-state). In these states, no box can be preferred against others. Therefore, we delete all of them to prevent from confusion. Note that the state machine has already deleted all of them except the last tracked box. So this last item is deleted here to complete the deleting process.

5.5 High-level Vision

Face and hands are two key parts of human body used by people for natural communication and interaction. Moreover, facial features carry significant information about one's emotional state that can influence the interaction. Therefore, our robot must be able to detect and track these parts. High-level vision is the module that is responsible for this task. In recent years many methods have been proposed for hand, face and facial features detection or tracking [Gavrila99, Moeslund01,Yang02].

Two popular approaches used in this context are knowledge-based and appearance based methods. The problem with knowledge-based approach is that is it difficult to translate our knowledge about faces into rules effectively. So we preferred to use an appearance-based method. A number of techniques can be used for appearance-based recognition such as template matching, statistical modeling, eigen space and neural networks. We found neural network the most promising one for our purpose. Here is a brief description of our reasons for choosing neural network:

- **Template Matching:** Typically all pixels of the template are equally likely in sense of importance [Sakai 69, Yuille 92]. This is a not true most of time and it is a considerable constraint.

- **Statistical Modeling:** Generally requires a pre-defined model such as a mixture of Gaussians [Sung 95]. The right choice of the models is not a trivial task and bad choice can give very poor results.

- **Eigen Space:** It relies on computation of eigen values and eigen vectors of the observations covariance matrix [Moghaddam 97]. This computation is time-consuming and not suitable for real-time tasks.

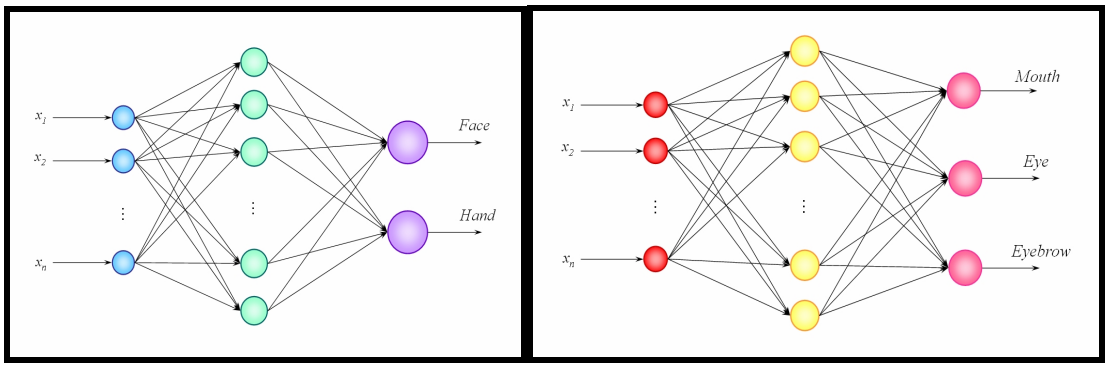
- **Neural Network:** It utilizes data in a weighted manner. Weights are adjusted automatically to achieve a (sub)optimal result. It doesn't need *a priori* model. It has a good generalization capability and works fast, specially in recall mode. Most of Neural Networks are potentially able to approximate any arbitrary mapping with the desired accuracy if enough neurons are allocated.

Perhaps the most notable work on neural network based face detection is the one carried out by Rowley *et al.* [Rowley 98]. They use a multilayer neural network to learn the face and non-face from intensities and spatial relationships of pixels. It takes a few minutes to for their network to respond and this is clearly not suitable for real-time tasks. This delay is however due to the size of their search space, not neural network itself. Instead of applying their brute-force search scheme, we first reduced search space by a fast low-level processing. In this level, we are left with a few boxes with known locations and sizes. Nevertheless, pose and rotation are still a problem. To keep up the response time of our system low, we made a simplifying assumption. Since a face-to face situation is likely to arise during interaction, we only train our neural network with frontal view of faces with no rotation[*].

[*] In on hand, since the type of a box does not change, once it has been recognized successfully, recognition is no longer applied to that box during tracking. On the other hand, it is not odd to expect

The same scenario holds for hand detection. Waving hands is a common way for attracting one's attention. When waving, observer sees a frontal view of hand with open fingers rotating with a limited range. Therefore, it is highly expected to have in observer's sight a view where hand and fingers are seen almost vertical. We trained this situation to our network.

Face/Hand detector was implemented using a multilayer perceptron network with one hidden layer. We used Hyperbolic Tangent as the activation function of neurons. The network has 12 inputs and 2 outputs. Each output determines the type of input (face or hand) as shown in Figure (5.9-a). When none of outputs fires, the box is considered to be unknown.



(a) **(b)**

Figure 5.9 Multilayer Perceptrons

To ignore background pixels in input data, some researchers have proposed an oval-shaped binary mask to be multiplied by the input window [Sung 95, Rowley 96]. This idea is supported by the fact that a fontal face looks almost oval-shaped. This shape is however an approximation, and it is possible that some face pixels get ignored or even worse, some background pixels enter inside the mask.

We adopted a different approach for eliminating background pixels that not only solves the mentioned problem, but also works for hand detection where a simple mask can't represent its shape. We exploit skin-color map of the box of interest so that non-skin pixels vanish automatically. Since in both of these limbs skin color is the dominant color, this approach yields a good representation of the their actual shape.

The box is down-sampled by replacing each pixel block with a single pixel whose value is equal to the average color of the block. Block sizes are chosen such that the down-sampled image is of size 6x6. For further reduction of dimensionality, we projected this box to horizontal and vertical axes and computed their histograms. These 6+6 features were normalized and fed to the mentioned network with four hidden neurons. The network was trained by back propagation learning algorithm with momentum.

---

capturing a frontal view of face during interaction. So relying on frontal view doesn't make a serious problem in our application.

Training samples were obtained from different people in canonical situation (waving for hand and frontal for face), possibly with small deviations. Negative samples were obtained by recording wrong detections during tracking and retraining network by them. Satisfactory result could be obtained as shown in Figure (5.10).

For facial feature detection, another network was employed Figure (5.9-b). Three types of facial features were considered: mouth, eye and eyebrow Our experiments showed that in contrast to face detection, color has less important role in facial feature detection and the major contribution comes from intensity channel. This is good news because we can work with a smaller dimension without performance degradation.

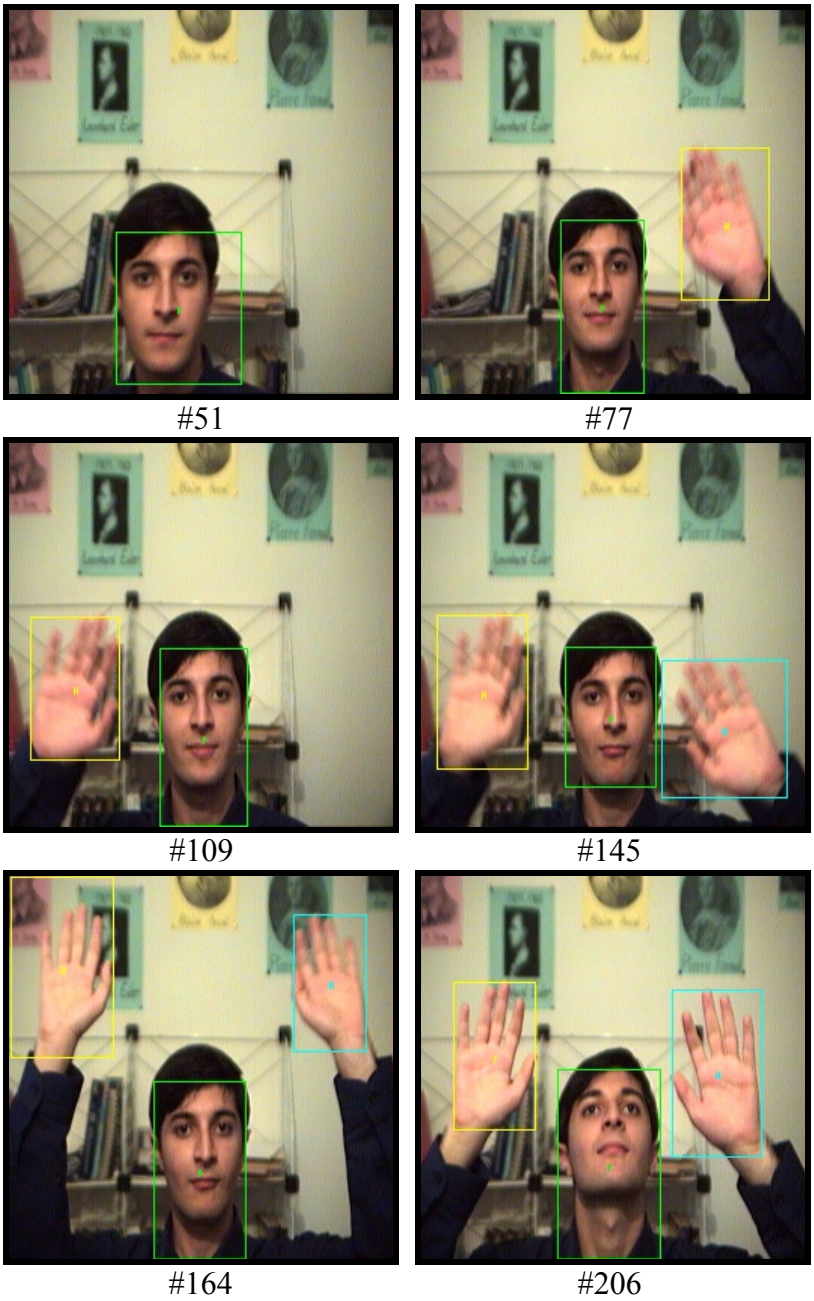


Figure 5.10 Recognized Body Parts

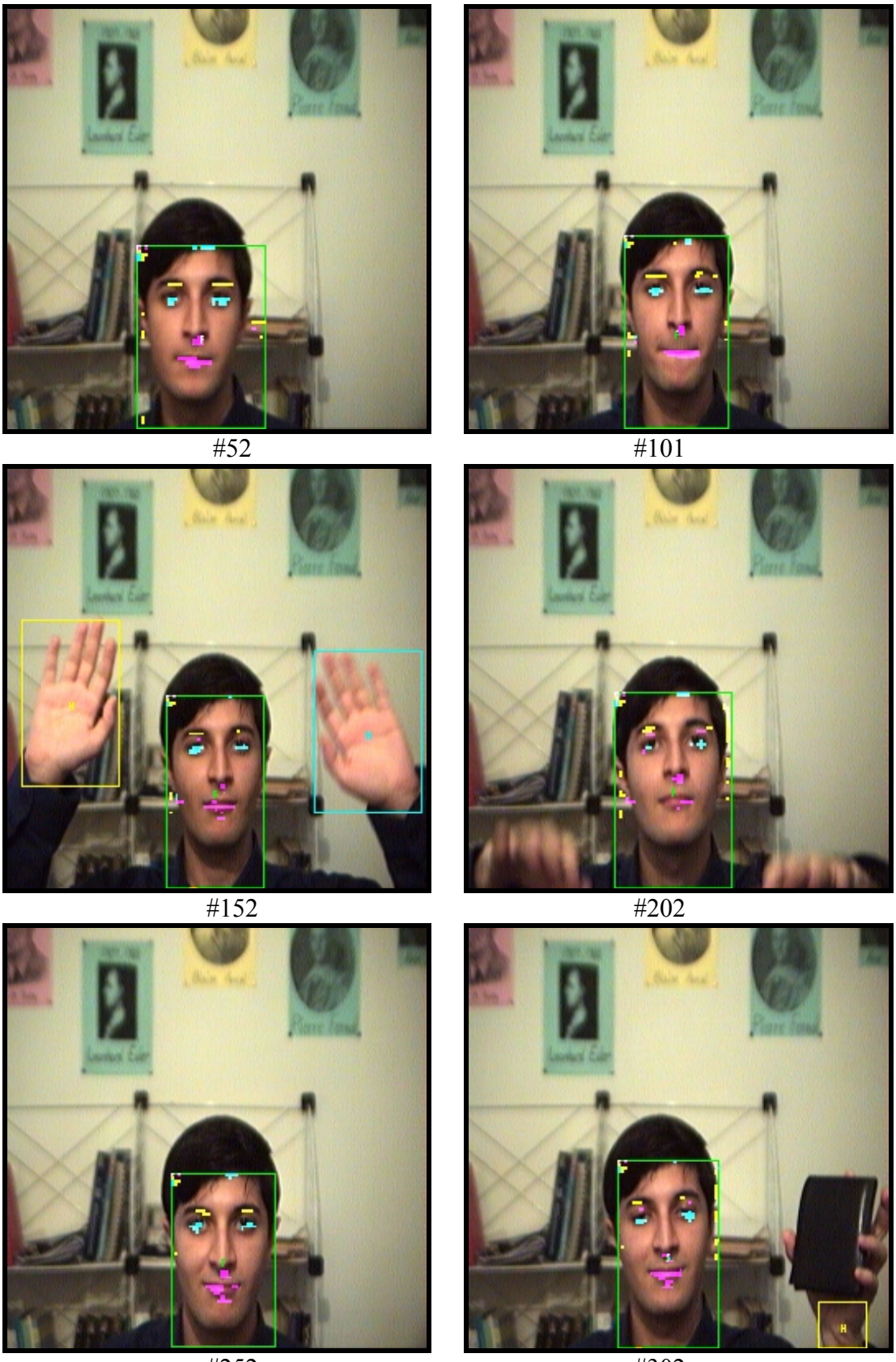


Figure 5.11 Facial Feature Tracking

For preparing inputs of the network, we first rescale the face box to a 35x35 image. This was the smallest size that could preserve the visibility of facial features. A 7x7-sliding window is moved over this image. In each cycle, the contents inside this window are read and normalized to be fed to network. The output of the network determines the class to which the centroid pixel of the window belongs. 8 neurons were allocated for hidden layer and the network was trained by error back propagation algorithm. Results of facial feature detection are shown in Figure (5.11).

Unfortunately due to the time constraint for accomplishing this project, I could not work more on this level. To make the results of facial features detector usable for an accurate tracking and an precise parameterized representation, some works must be done in the future. Here I propose some suggestions for the future work of this project.

First of all, the output of the network, which can be assumed as an image itself, must be filtered to remove isolated pixels. Then expert knowledge (e.g. if-then rules) should be applied to eliminate wrong clusters based on geometrical relationships between features. Finally this result should be used as the initial location for deformable models or active contours to capture the shape of facial features. Such a parameterized representation can be helpful for further analysis e.g. action or emotion recognition.

5.6 Attention

Attention refers to selection a subset of the information that is arriving at the brain and is potentially available. It implies allocating resources to some thing at the expense of not allocating them to something else. Thus it helps brain to respond promptly by concentrating computational resources on the information of interest, in particular, about the vast information falling on the retina. In engineering language, the effect of simulating attention helps an agent to demonstrate a satisfactory behavior in real-time.

However, psychophysical studies as well as introspection indicate that we are not blind outside the focus of attention, and that we can perform simple judgments [Braun 90]. But those judgments are less accurate than in the presence of attention on objects not being attended to [Lee 99, Yeshurun 98]. Based on this observation, we provided the ability of tracking multiple items, with most motivating item chosen as the focus of attention for directing expensive or limited resources toward it. Here we will describe, in separate sections, how we achieved these two goals simultaneously. Having a look at Figure (5.12) can be helpful for our further discussion.

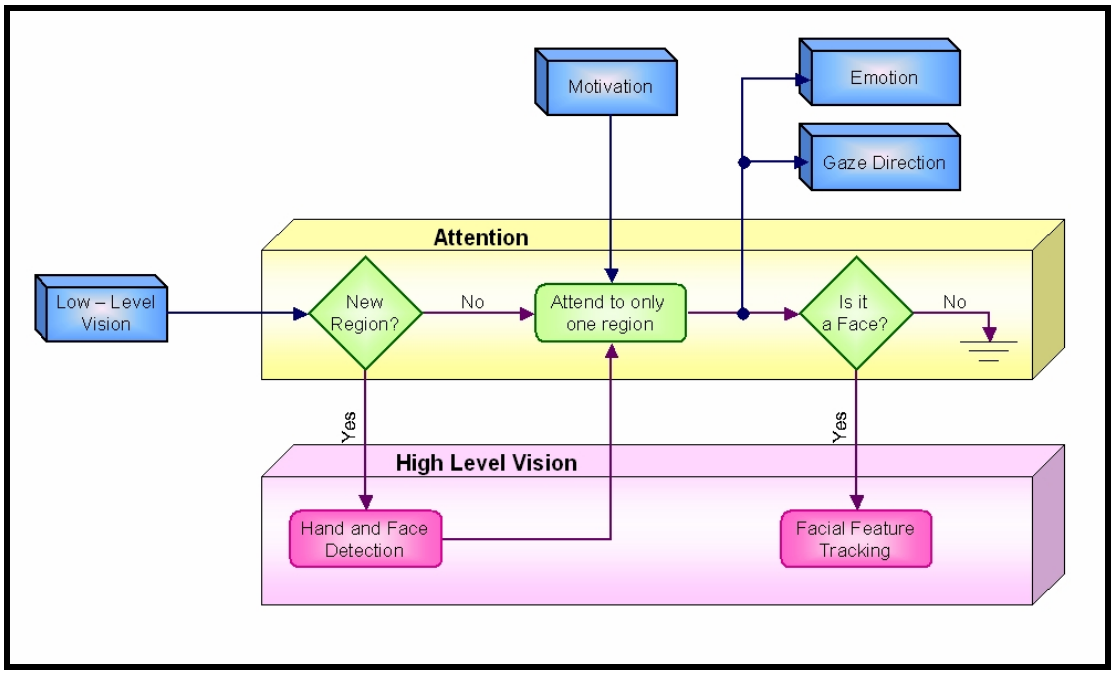


Figure 5.12 Attention Module

5.6.1 Attending to multiple items

Primary perceptual selection in attention module occurs at the entry where low-level vision information is arriving. At this place, feeble boxes are ignored because it is not known in advance whether they are noise or real items. Since this will be clear in a short time (using state machine proposed in 5.4.4 for persistency analysis), it's better to ignore feeble items (until becoming persistent) rather than the contaminating resources by noise.

Since face/hand recognition is a high-level task, it consumes an extensive amount of computational resources. On the other hand it is not always possible to recognize an item as soon as it appears. For instance, when a face is first detected, it could appear in any pose causing not necessarily a frontal view*. Therefore, the attention mechanism should adopt a mechanism for managing high-level vision in an efficient way. Otherwise high-level vision will attempt to recognize unknown items all the time even if it fails repeatedly.

To cope with the mentioned problem, we have incorporated a curiosity factor in our attention model to create more attract for recognizing unknown boxes. However the curiosity decays exponentially over time. Accordingly, The less successful recognition of an item is, the less attractive the item becomes.

The degree of curiosity at any moment is inversely proportional to the number of attempts since when it was first detected till that time. So as soon as a new item is detected, high attention is paid to it, i.e. more quickly high-level vision tries to recognize it. If recognition fails frequently, longer delays take place between successive recognition attempts. Here is its formulation:

$$C = \exp(-k \times n) \qquad ; \qquad D = \frac{1}{C} \tag{5.6}$$

Where C denotes curiosity, D is the delay between successive recognition attempts, N is the number of attempts so far and k is a positive constant that is inversely proportional to the program execution speed on a machine. Therefore, it may differ from one computer to another. In our system, we chose k to be one.

5.6.2 Focus of attention

Although it is possible to attend to multiple items for a simple exploration with aid of inexpensive resources, there are resources that are either computationally expensive such as tracking facial features, or they are physically limited, such as the ability of fixation on only one item. Therefore, the most motivating item must be selected and such resources must be directed toward it. The motivated item is determined by motivation module, which will be discussed in section 5.6.3.

Once this item is determined, attention module reports its type and location to other modules such as emotion system and gaze control. Moreover, if the type of the item, determined by high-level vision to be a face, high-level vision is invoked for tracking facial features within that face. As mentioned earlier, due to the time constraint if this project, the result of facial features tracking is not used at the moment. However, they are to be used in the future to influence other modules, particularly emotion module of Aryan.

---

* Recall that we trained Aryan's high-level vision with frontal face views. Therefore, it will not be able to recognize a limb until it appears in the canonical pose.

5.7 Motivation

In human terms a “motive” may be described as a need or a desire that causes a person to act. In information-processing terms, it is a variable that has influence on the decision-making process and may cause action. “Motivation” is a driving force (dependent both on the internal state of the agent and the state of the external world) that arouses and directs action towards the achievement of goals [Norman 95].

Accordingly, in Aryan motivation deals with high-level concepts perceived by its high-level vision, to achieve high-level goals. Since Aryan can only exhibit reactive behaviors, its goals are not in form of plans, but high-level instantaneous goals. Currently the robot’s only motivation is to behave socially. That means to be in the presence of people and to be stimulated by people [Breazeal 98]. To achieve this goal, Aryan's attention must be biased toward the item that best satisfies this goal.

Since Aryan's behavior is reactive, there is no internal state in motivation module. We implemented motivation to be fully hardwired, static (no change over time) and always active (no intensity level). Although in our architecture, motivation interacts with attention module only (due to the central role of attention in our architecture for forming behaviors), it can indirectly influence other modules and totally, Aryan's behavior. For example, the motivated item determines direction of gaze. This will eventually affect other modules such as low-level and high-level vision and emotional system.

We implemented this scheme by ordering tracked boxes with respect to their potential contribution to social interaction. A face has the highest motivational degree. Hand and an unknown region (yet moving and skin-toned) are in the second and third place respectively. When there are multiple items of the same type, the largest one is defined to be more motivating. The most motivating item is selected to attract attention.

5.8 Emotion and Facial Expression

Emotions play a key role for improving believability in life-like agents [Koda 96, Reilly 96]. This is typically based on the claim that such agents can interact better, in a more natural way with humans and look more realistic and believable [Hayes 95]. The expression of emotion in the face is an important biological aspect of emotion that has significant implications for how emotion is communicated in social situations [Darwin 1872].

Generally, there are two different emotion representation methods, discrete basic emotions and continuous dimensions. Basic emotions are those that can be taken as building blocks out of which other, more complex emotions form. A particular case of basic emotions is the one proposed by Ekman [Ekman 92]: anger, disgust, fear, happiness, sadness and surprise.

In continuous space representation, the two most commonly used dimensions are valence (positive/negative) and activation or arousal (calm/excited) [Whissel 89]. It has been shown that these two dimensions are not enough to distinguish among all the basic emotions. For example fear and anger are both characterized by negative valence and high arousal. Therefore sometimes another dimension, such as potency (powerfulness/powerlessness) [Canamero 97] or stance [Breazeal 00#] is added.

We adopted basic emotion representation, because it is easier for human interpretation and for implementation. In addition, these emotions correspond to distinct facial expressions, which are supposed to be universally recognizable across cultures [Ekman 93, Ekman 78, Izard 83]. We discussed in Chapter 1 that emotions like happiness, sadness and disgust are highly relied on lips posture. Since Aryan's lips are rigid, it cannot express emotions them, but the rest of basic emotions: anger, fear and sadness plus a neutral face. These emotions mainly involve eyebrows and mouth postures and are less dependent on lip movements.

Different emotional models have been proposed in the literature. For instance, Damasio suggests somatic-marker mechanism for modeling emotions [Damasio 94]. TABASCO architecture [Staller 98] is a model based on the emotion appraisal theory. OCC theory of emotions is another popular model [Ortony 88]. Most of these models focus on modeling emotional system as it seems to be in humans and animals. Therefore, using them is helpful when improving our knowledge about the nature of emotion and is considered.

However, the main application of interactive robots is in domains such as toys, educational tools, entertainment tools, and therapeutic aids [Doyle 99, Koda 96, Rizzo 99]. In these applications the major concern is not to simulate naturally occurring processes using hypothesized underlying mechanisms, but obtaining high degree of believability and a natural interface for human-robot interaction [Hayes 95]. The emotional model that we look for should enable Aryan to argue about emotions in the same way as humans do. An event that scares humans, for example a sudden approach toward him, should also scare the robot. The emotion model must be able to evaluate all situations that the robot might encounter. Such an emotion model enables Aryan to show right emotion at the right time. This model can be simply implemented using a state machine as shown in Figure (5.13).

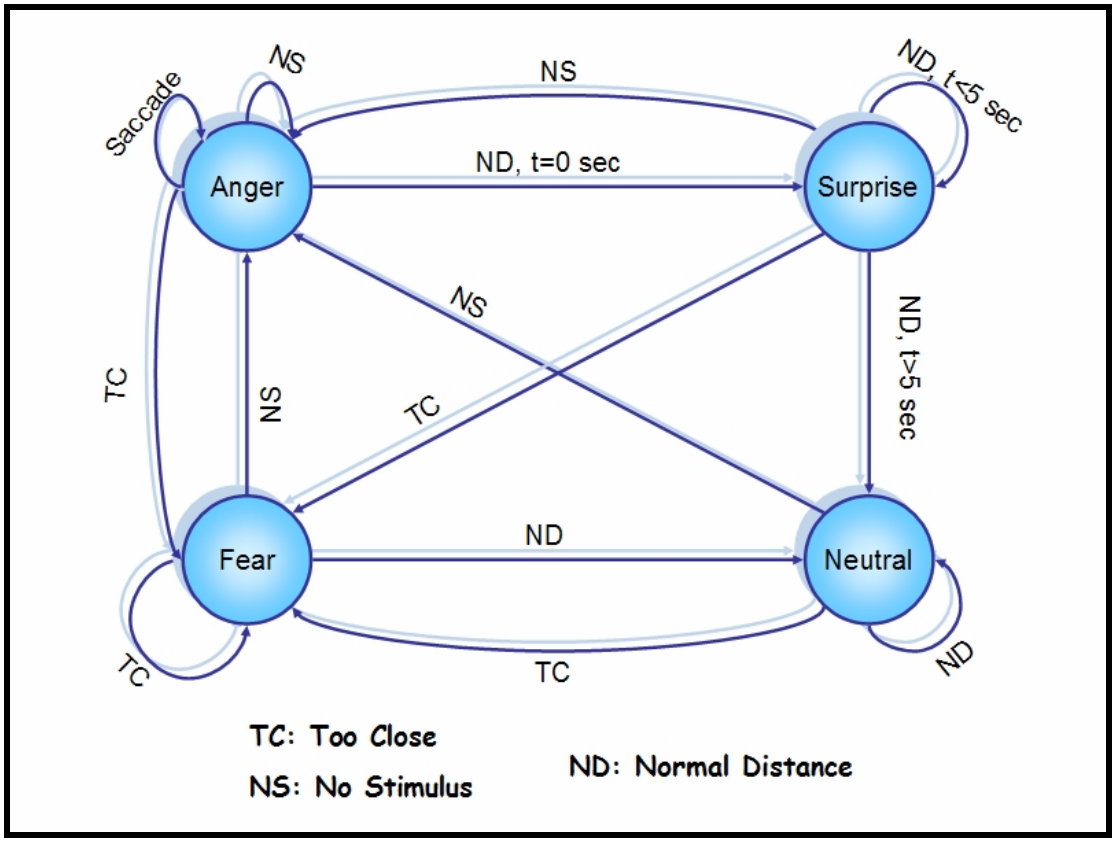


Figure 5.13 Emotion State Machine

The goal of this state machine is taking Aryan to the appropriate emotional state according to the previous state, an internal counter value and its visual stimulation. In this model states correspond to emotional states and visual features direct transitions over states. There is also a counter denoted by *t*, which is increased over time. Here visual stimulation means only that part of visual information that Aryan has attended to. Thus, there is a path from attention to emotion. The emotional state will influence facial expression module.

Facial expression module does nothing except mapping emotional states to facial expressions by a set of rules. Expressive rules are already discussed in section 1.4 with details. We will devote a separate section to discuss about facial expression module. We mentioned there that due to the physical constraints of Aryan's face, it can only express those emotions that are less dependent on lip postures. These expressions (anger, fear and surprise) plus a neutral expression correspond to the states of emotion machine.

One could ask why considering separate modules for emotion and facial expression, when the only task of emotion module is facial expression. In fact, their separation was not necessary in this stage and these modules could be combined. However, for the sake of extendibility, e.g. incorporating emotional decision-making in emotion module, we preferred to consider separate modules. So we will not need to change the structure of the current architecture for later extensions.

Some emotions are comprised of a large impulsive response followed by a gradual decay back down to a base state. Among emotions that Aryan can express, surprise is

the best candidate for incorporation of decay factor. The rest of emotions were considered to exist as long as their corresponding stimulation survives. In our model, we control decay term by increasing a counter value denoted by *t* in Figure (5.13). This value is increased over time and when a transition occurs between states, it is reset to zero. When it becomes larger than a threshold, 5 second for example, Aryan returns to neutral state, regardless of its current emotional state.

Another interesting feature of the proposed model is incorporating saccades (rapid eye movements). When Aryan is left alone, it becomes angry. In this state it occasionally issues an involuntary saccade that to make large changes in its sight, hoping to find a mate in the new view. That's why there is also a path from emotion module to gaze adjustment. Performing saccade is determined by dicing, with a predefined probability distribution.

Figure (5.14) shows Aryan in action. At the beginning Aryan is in neutral state, but as soon as it detects human's hand, it gets surprised and displays its emotion by facial expression.

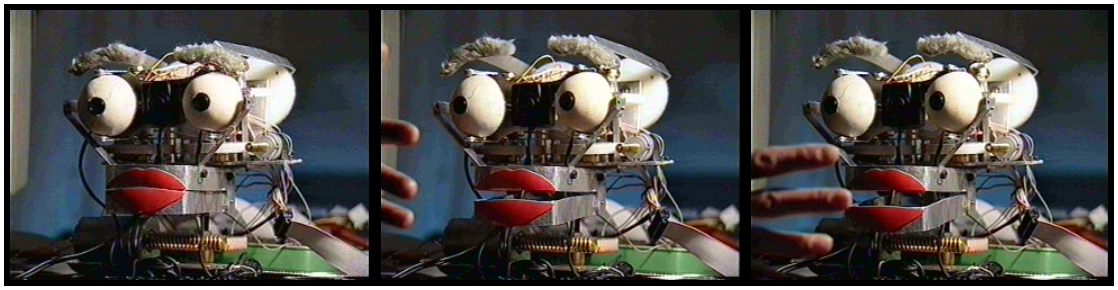


Figure 5.14 Aryan getting surprised

5.9 Gaze Adjustment

Gaze direction is a powerful social cue that people use to determine what interests others. By directing the robot's gaze to the visual target, the person interacting with the robot can accurately use the robot's gaze as an indicator of what the robot is attending to. This greatly facilitates the interpretation and readability of the robot's behavior, since the robot reacts specifically to the thing that is it looking at [Breazeal 02].

5.9.1 Eye Movements

Two anthropomorphic narrow field of view eyes not only supply Aryan with this social ability, but can also provide potentially a high-resolution depth map of what the robot sees. Although at this phase of project, we do not use disparity of binocular views to compute depth map, we control direction of gaze in both eyes. This will make disparity computation more effective for later use because the region of interest must be almost visible from both views. In addition, this will realize the social aspect of gaze control.

It is obvious that when looking at an object near to eyes, the eyes converge. On the contrary, when looking at distant objects, the eyes diverge such that ultimately gaze direction of the eyes become parallel. Vergence angle of eyes can be computed analytically from 3D coordinates of the center of interesting region in space, using simple trigonometric manipulation.

This is illustrated in Figure (5.15). *P* is the projection of the point of interest in space onto *x-z* plane. *L, R* and *M* are locations for left, right and middle camera respectively. *B* stands for half of base-line distance, i.e. *LR/2*. α and β denote vergence angles of left and right eyes respectively.

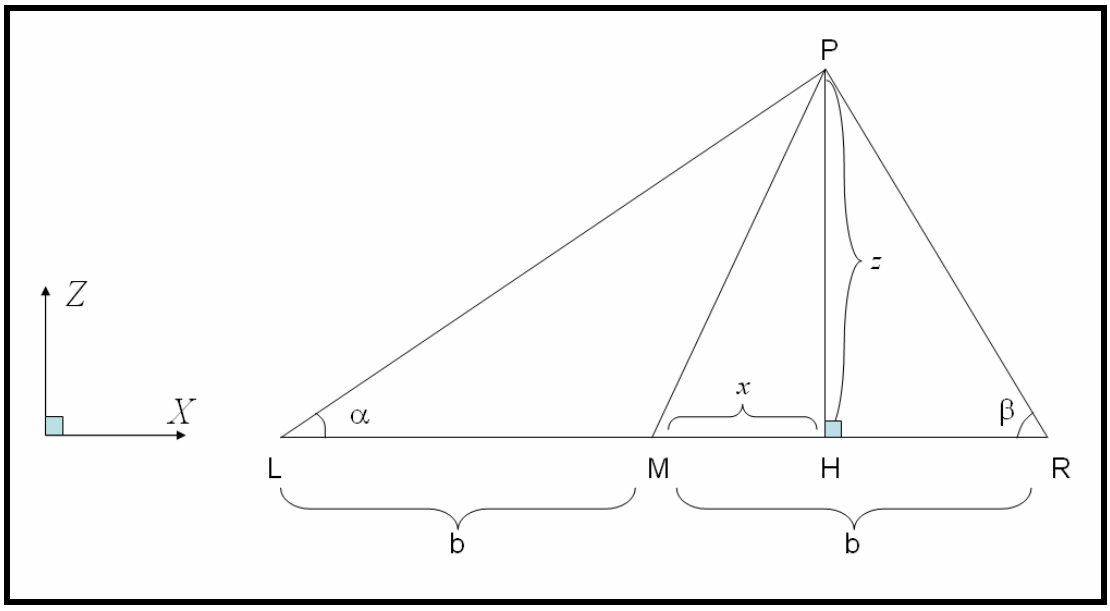


Figure 5.15 Trigonometric relations for vergence approximation

Now vergence angles are computed as follows:

$$\begin{aligned} \Delta LHP : \tan\alpha &= \frac{z}{b+x} \\ \Delta RHP : \tan\beta &= \frac{z}{b-x} \end{aligned} \tag{5.7}$$

We used the middle camera, which is wide field of view, to coarsely inspect environment and determine region of interest. Once this region in the image is determined, attention module shifts gaze toward it. The middle camera's frame is not affected by pan motion of the eyes. Therefore, the desired vergence angle for each eye can be computed directly, regardless of their current situation.

Similar manipulation yields the (joint) tilt angle for the eyes. However, since all three cameras share a common tilt, computed angle is always relative to the current tilt angle. This is shown by a differential compensator in equation (5.8). Note that y is instantaneously measured value relative to the current situation of camera.

$$\dot{\theta} = K \tan^{-1}\frac{y}{z} \tag{5.8}$$

So far we assumed that the 3D coordinates of P in space is available. However, we only have its projection on the middle camera lens, i.e. a plane. We need the equation, so called Camera Calibration, which relates points in the scene to points in the image and vice versa. There are well-known methods such as Tsai's algorithm for camera calibration [Horn 00]. Generally, in a pinhole camera model, the projection of a point in space onto image plane is computed by equation (5.7) where s is a scale factor converting the underlying nonlinear mapping to a linear one:

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} f_x & \tan(\alpha) & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \Re^T & -\Re^T t \\ 0_3^T & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{5.9}$$

Since we want measurements to be relative to robot's head, we can consider the projection center is at the origin of the world frame. This relaxes extrinsic parameters matrix to identity matrix. We also assumed zero skew, i.e. image axes are completely orthogonal. So the position of a point on each axis is computed by a linear equation with only one independent variable. Additionally, we considered the center of image as the origin of image frame. This will eliminate bias terms in the linear equation, i.e. u_0 and v_0, and yields a proportional relation as shown below:

$$\begin{aligned} u &= \frac{f_x}{z} x \\ v &= \frac{f_y}{z} y \end{aligned} \tag{5.10}$$

An iconic demonstration of this configuration is shown in Figure (5.16). Equation 5.10 yields a unique solution for u and v. However, we are interested in the inverse mapping, which is underdetermined (three unknowns with two equations). To solve the inverse mapping, we considered *z* being known. A reasonable approximation for *z* is the distance between Aryan and people interacting with the robot. We chose it being 125 centimeters.

Although deviations from this value results in non-centered view of the interesting region in the eyes, it suffices as a primary shift toward that region. In fact, it just tries, with incomplete data, to make the region visible in both eyes, not necessarily in the center. Later, stereovision module can be added to utilize depth and high-resolution images of each eye to finely move eyes from rough situation to the center of image in each eye.

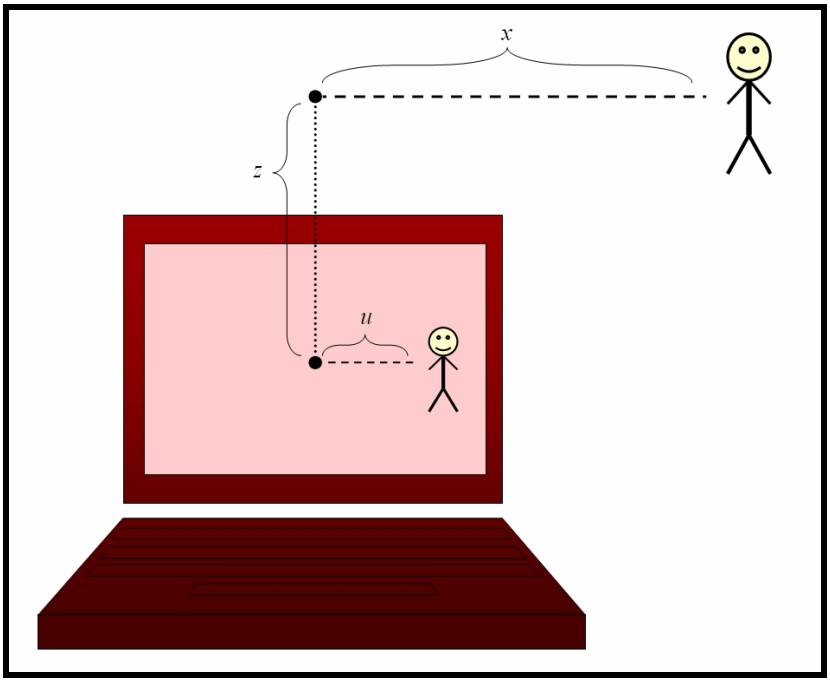


Figure 5.16 Simplified extrinsic configuration

Returning to calibration problem, there are two parameters, f_x and f_y, to be estimated. We selected some training points that their 3D coordinates (with respect to camera) were known. Obviously, the depth of these points was about the assumed value of *z*. We then measured the corresponding coordinates on image plane. Using these two sets, the proportional coefficient was computed by least square method as follows:

$$u = K_x x \rightarrow K_x = \frac{\sum_{i=1}^{n} x_i u_i}{\sum_{i=1}^{n} u_i^2} \qquad v = K_y y \rightarrow K_y = \frac{\sum_{i=1}^{n} y_i v_i}{\sum_{i=1}^{n} v_i^2} \tag{5.11}$$

The baseline of Aryan's eyes is 5cm (so b is 10cm in equation 5.7). Assuming *z* to be about 125cm, the value for *K*'s in 5.11 is obtained being about 0.75cm/pixel.

5.9.2 Neck movements

In nature, depending on the complexity of species, they can combine a group of body movements to affect their viewpoint and achieve fixation. For instance most of insects have fixed eyes attached to their bodies. Simple animals such as fish or frogs can move their eyes and body independently, but motion of their body results in very drastic changes in their view.

Therefore, they can only pursuit a moving object as long as they do not have to move their body. Evolution has solved this problem by equipping advanced animals and human beings with another limb, the neck, to control their viewpoint more effectively. By moving the neck and eyes, one can easily pursuit a moving object and it is rarely required to move his body.

When the head is free to move, people frequently engage in coordinated head and eye movements to bring a target object to the fovea. The question here is how to decompose the total gaze shift into head and eye movements. Freedman and Sparks [Freedman 97] found that the relative contributions of head and eye movements to the total gaze shift are non-linear functions of the initial eye position and the total gaze displacement.

Although the redundancy in neck and eyes allows for a wide range of movements to achieve a desired gaze shift, the decomposition of the task shows little variance in highly trained monkeys [Freedman 97]. Generally, eye movements are solely responsible for small shifts, while the head becomes increasingly involved in larger movements. This is due to the following facts:

- The anatomy and relative position of human's eye with respect to his head is such that the range of motion of the eye (+/- 45 degree pan/tilt) is limited to almost half of the range of motion of the neck (+/-80 pan, +90 tilt up, -60 tilt down) [Batista 95]. Therefore neck motion can sweep a wider view than the eye.
- The mass of the head and consequently its inertia is so much larger than the mass and inertia of the eye. Therefore moving the eye, especially for fast saccadic movements, is more efficient than moving the neck.

Based on these basic observations, we implemented a very simple mechanism for neck control. Normally, Aryan tracks people by its eyes. However, when any of its eyes reaches its extreme limit, a compensatory motion is performed by neck in the right direction. When moving the neck, eyes stay stationary to keep system stable.

Currently neck motions are of fixed amount, about 30 degrees. This value was obtained empirically. Indeed, the right amount of neck movement highly depends on person's distance from Aryan, However, assuming people interacting with Aryan are in a roughly known distance from it (similar to 5.9.1), fixed neck motions did not make much problem for gaze adjustment. Reaching eye limits was realized by considering a threshold value for nearness.

Figure (5.17) shows how neck motions compensate the fovea error in the middle camera when I am moving my head from left to right. Looking at the figures, one can observe that the compensation is achieved with a delay. We could overcome to this difficulty by increasing control gains of neck joints (see Chapter 3). However, it could bring instability problems, especially for people interacting in a farther distance than the assumed one. So we preferred this delay over the risk of instability. Incorporation of prediction for tracking can also reduce tracking latency.

Figure 5.17 Neck Compensatory Motion

Chapter 6

Conclusion

6.1 Summary

In this project, I developed a robot face named Aryan as a platform for future research and study on natural human-robot interaction. The robot is able to perceive people around it by active vision and express its emotion accordingly. I constructed Aryan from elementary, widely available and low-cost components with my limited personal budget. Although I wouldn't have learned as much should these parts were readily available, the disadvantage was the longer time required for implementation. The price of components paid for the whole robot is about 375$.

I constructed the face from metal pieces to reflect its robotic appearance and avoid from people's false expectations of the robot's capabilities. Aryan's looking was enhanced by augmenting its facial features with colorful and suitable materials. Since humans attribute a communicative value to eye movements, we constructed an anthropomorphic visual mechanism, a binocular active vision system. We evaluated the recognizability of Aryan's facial expressions and obtained satisfactory results.

The face is actuated through eight degrees of freedom namely: neck pan, neck tilt, jaw, left eye pan, right eye pan, eyes joint tilt, eyebrows elevation and eyebrows arcing. Actuators are home-built servomotors with gears or drum and rope mechanism for force transmission. We compared DC motor vs. Stepper motor and potentiometer vs. shaft encoder for constructing our servos.

We modeled a DC motor and reviewed related mathematics from control theory and concluded that a PI controller could be a good choice for our task. We also presented our reasons for preference of a digital controller instead of an analog one. We designed a suitable controller and discussed each of its modules separately in details. The main chips used for controller development were AT89C51 microcontroller, XC9572 Complex Programmable Logic Device and ADC8016 Analog to Digital Converter. We explained software drivers for both microcontroller and PC side using flow charts and pseudo-codes.

To provide the required power for motors, we also designed an amplifier board capable of controlling speed and direction of 12 motors simultaneously. The design was based on H-Bridge circuit and implemented by BJT transistors. We explained our reasons for preference of BJT against FET transistors. We analyzed the circuit and

also equipped it with optical isolators to secure controller against high-power malfunctions in amplifier side. We included simulation result of the circuit.

I proposed modular brain architecture for Aryan. Briefly, visual sensors capture visual information and represent them in an appropriate form for further processing. Low level-vision tracks moving skin-toned regions. Attention module directs expensive and limited resources toward the most motivating region in terms of high-level goals. Emotion module controls emotional state of the robot according to perceptual data. Finally appropriate motor commands for facial expressions and gaze direction are generated.

Low-level vision detects skin-colored regions based on a statistical model of skin color in chromatic color space. Motion is also detected using image differencing. These results are combined in a meaningful way to find possibly interesting regions. Finally, a lightweight and efficient tracking algorithm has been proposed

High-level vision utilizes neural networks to recognize hands and faces and also detect facial features. We mentioned advantages of employing neural networks versus other pattern recognition methods. We also explained what pre-processing was required for obtaining practical features to feed them to neural network.

We defined attention and mentioned our inspiration for attending to multiple objects from a biological viewpoint. We implemented multiple attention mechanism and incorporated curiosity by a simple model to reduce its computational cost. We also explained how attention bridges between high-level and low-level vision modules

We defined motivation and discussed about its role. It was comprised of one goal only, being social, which biases Aryan's attention such that it tends to interact with people. This is achieved by ordering tracked boxes according to their type and size. We reviewed two general representations of emotions, discrete and continuous, and reasoned about why we adopted discrete basic emotions. Since Aryan has rigid lips, we selected emotions that are less relied on lips posture, i.e. neutral, anger, fear and surprise.

Gaze direction is a powerful social cue that people use to determine what interests others. We implemented a basic gaze adjustment using the middle camera. Complete adjustment can be achieved later when stereovision software is added. This basic adjustment is required because before stereo matching, the region of interest must be almost visible in both views. We described how we roughly took eyes to their right positions by trigonometric manipulations and parameter estimation.

6.2 Conclusion and Future Works

My most major finding of this experience was to learn how a real complex robot works. Though I had learned the theoretical part of the matter during the studies or class courses but it was a new experience for me to work on real problems and see the difficulties that you face when you are working on them. I had a feeling of being part of the problem interacting the other parts of it. I believe that behind every theorem, parameter or formula there is a real concept that you can't get a deep feeling of it if you don't live with them. I think it was valuable.

Aryan, comparing with some emotionally expressive robot faces like Feelix [Canamero 00], eMuu [Bartneck 02] and Sparky [Scheeff 00], has richer sensory capabilities. Its active vision system significantly involves brain functions such as attention, low-level and high-level processing, feature extraction and pattern recognition. Right now, its facial feature tracking system is not complete yet; it roughly tracks facial features as you could see in chapter 5. In the future we are going improve its accuracy using active contour models and also get help of depth cue obtained from binocular views. We hope to improve Aryan's visual system in the future, such that it can compete with sophisticated robot faces like Kismet [Breazeal 00]

We also concluded that currently constructed facial features couldn't represent some important emotions such as happiness and sadness. This is due to the rigidity of Aryan's lips. For resembling lip movements with satisfactory accuracy, we noticed that at least four motors are required. Since we had to develop the whole face in a limited time, we ignored lips at this stage such that we could work on the parts of the robot. In the future, we are going to add flexible lips.

Another important but missing factor in Aryan is learning ability. We did not incorporate learning due to deadline issues for this project. We however emphasize the need for this feature in a sociable robot. Recent research on learning mechanisms for individualized (in opposite of collective) social agents suggests learning by imitation as a natural and promising learning method for such agents [Dautenhahn 02].

Right now, Aryan's brain acts reactively. Integrating it with deliberative capabilities for high-level tasks such as planning and also incorporating more a larger set of motivations can result in more intelligent behaviors such as non-verbal dialogues, turn taking, decision-making and reasoning. Combining it with a superior emotion model can also realize emotional decision making for achieving acceptable sub-optimal solutions in real-time.

Currently neck movements are not coordinated with eyes and the head just performs fixed rotations. An efficient decomposition of the total gaze shift into head and eye movements is required for achieving a life-like behavior. Ultimately, we would like to add other sensory motor capabilities such as vocalization, utterance and simple natural language processing. We would also like to develop a body for Aryan, particularly hands for manipulating objects.

References

[Baluja 95] S. Baluja and D. Pomerleau, "Using the Representation in a Neural Network's Hidden Layer for Task-Specific Focus of Attention", IJCAI 1995, pp. 133-141

[Bartneck 01] C. Bartneck and M. Okada, "Robotic user interfaces", in: Proc. Hum. And Comp. Conf., 2001.

[Bartneck 02] C. Bartneck, "eMuu an embodied emotional character for the ambient intelligent home", Unpublished Ph.D. thesis, Eindhoven University of Technology, Eindhoven. 2002.

[Batista 95] J. Batista, J. Dias, H. Arajo, A. Traça de Almeida, "The ISR Multi-Degrees-of-Freedom Active Vision Robot Head", M2Vip95-Second International Conference on Mechatronics and Machine Vision in Practice, Hong Kong, September 1995.

[Braun 90] J. Braun, D. Sagi, "Percept Psychphys", 1990, 48(1): 45-58

[Breazeal 00#] C. Breazeal(Ferrell) and B. Scassellati, (2000), "Infant-like Social Interactions Between a Robot and a Human Caretaker". To appear in Special issue of Adaptive Behavior on Simulation Models of Social Agents, guest editor Kerstin Dautenhahn.

[Breazeal 00*] C. Breazeal, A. Edsinger, P. Fitzpatrick and B. Scassellati "Social Constraints on Animate Vision", IEEE-RAS International Conference on Humanoid Robots 2000.

[Breazeal 00] C. Breazeal, "Sociable machines: Expressive social exchange between humans and robots," Ph.D. dissertation, Dept. Elect. Eng. Comput. Sci. Mass. Inst. Technol., Cambridge, 2000.

[Breazeal 02#] C. Breazeal, "Regulation and Entrainment for Human-Robot Interaction", D. Rus and S. Singh (Eds.), in: Int. Journal of Experimental Robotics, 21(10-11), pp. 883-902, 2002

[Breazeal 02*] C. Breazeal and L. Ariyananda, "Recognizing Affective Intent in Robot Directed Speech", in: Autonomous Robots, 12:1, 2002

[Breazeal 02] C. Breazeal, "Designing Sociable Robots", Copyright 2002, MIT Press.

[Breazeal 98] C. Breazeal, "A motivation system for regulating human-robot interaction", in: Proc. Nat. Conf. Art. Intel., 1998.

[Breazeal 99*] C. Breazeal, "Robot in Society: Friend or Appliance?". In Agents99 workshop on emotion-based agent architectures, Seattle, WA.

[Breazeal 99] C. Breazeal, B. Scassellati, "A Context-Dependent Attention System for a Social Robot", Thomas Dean (Ed.), Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99, Stockholm, Sweden, July 31 - August 6, 1999. 2 Volumes, pp. 1146-1153

[Breig 98] M. Breig, M. Kohler, "Motion Detection and Tracking under Constraint of Pan-Tilt Cameras for Vision-Based Human Computer Interaction", Research Report No. 689/1998, Fachbereich Informatik, Universitet Dortmund, 44221 Dortmund, Germany

[Buchanan 99] W. Buchanan, "PC Interfacing, Communications and Windows Programming", Copyright 1999, Addison-Wesley.

[Buchanan 99] W. Buchanan, "PC Interfacing, Communications and Windows Programming", Copyright 1999, Addison-Wesley.

[Bugard 98] W. Burgard, A.B. Cremers, D. Fox, D. Hahnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, "The Interactive Museum Tour-Guide Robot", Proc. of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)

[Canamero 00] L. D. Canamero, J. Fredslund, "How Does It Feel? Emotional Interaction with a Humanoid LEGO Robot", In K. Dautenhahn (Ed.), Socially Intelligent Agents: The Human in the Loop. Papers from the AAAI 2000 Fall Symposium (Menlo Park: AAAI Press.

[Canamero 01] L. D. Canamero, J. Fredslund, 2001. "I Show You How I Like You-Can You Read it in My Face?", IEEE Transactions on Systems, Man and Cybernetics, Part A, 31(5): 454-459.

[Canamero 97] L. D. Canamero, "Modeling motivations and emotions as a basis for intelligent behavior", in: W. Johnson, ed., Proc. Intl. Conf. Auton. Agents, 1997.

[Chen 86] W. Chen, "Passive and Active Filters : Theory and Implementations", Copyright 1986, John Wiley & Sons.

[Cole 98] J. Cole, (1998), "About Face", MIT Press, Cambridge, MA.

[Cutler 98] R. Cutler, M. Turk, "View-based Interpretation of Real-time Optical Flow for Gesture Recognition,", IEEE International Conference on Automatic Face and Gesture Recognition, April 1998, Nara, Japan

[Damasio 94] A. R. Damasio, "Descartes' error - Emotion, Reason and Human Brain", Picador, London, 1994

[Darwin 1872] C. Darwin, "The expression of emotions in man and animals", New York: D. Appleton and Company, 1872.

[Dautenhahn 02] K. Dautenhahn, C. Nehaniv (eds.), "Imitation in Animals and Artifacts", MIT Press, to appear Spring 2002.

[DiSalvo 02] C. DiSalvo, F. Gemperle, J. Forlizzi and S. Kiesler, "All Robots Are Not Created Equal: Design and the Perception of Humanness in Robot Heads", DIS2002 Conference Proceedings, pp. 321-326.

[Doyle 99] P. Doyle, "When is a Communicative Agent a Good Idea?", In Workshop on Communicative Agents of the 3rd International Conference on Autonomous Agents, 1999, Seatlle.

[Ekman 78] P. Ekman and W.V. Friesen, "Pictures of Facial Affect", Palo Alto, CA: Consulting Psychologists Press, 1978.

[Ekman 82] P. Ekman, W. Friesen, "Measuring facial movements with the Facial Action Coding System", in Emotion in the Human Face, Cambridge University Press, 1982, UK, pp. 178-211.

[Ekman 92] P. Ekman, "An argument for Basic Emotions", Cognition and Emotion, 6(3/4), pp 169-200, 1992.

[Ekman 93] P. Ekman, "Facial expression and Emotion", Am. Psychologist, vol. 48 pp.384 392, 1993

[Feedman 97] E.G. Freedman and D.L. Sparks, "Eye-head coordination during head-unrestrained gaze shifts in rhesus monekys", J. Neurophysiol., 77:2328–2348, 1997.

[Gavrila 99] D. M. Gavrila, "The visual analysis of human movement: A survey", Computer Vision and Image Understanding, 73(1): 82-98, January 1999

[Hara 98] F. Hara, "Personality Characterization of Animate Face Robot through Interactive Communication with Human", in "Proceedings of the 1998 International Advanced Robotics Program (IARP98)", Tsukuba, Japan, pp. IV-I.

[Hayes 95] B. Hayes-Roth, "Agents on Stage: Advancing the state of the art of AI", Proceedings of 14th International Joint Conference on AI, 1995, pp. 967-971.

[Horn 00] B. P. Horn, "Tsai's camera calibration method revisited", unpublished, 2000.

[Huwer 00] S. Huwer, H. Niemann, "Adaptive Change Detection for Real-Time Surveillance Applications", Third IEEE International Workshop on Visual Surveillance - VS-2000, Dublin, Ireland, July 2000, p. 37-45

[Iida 99] F. Iida, H. Ayai, F. Hara: "Behavior learning of Face Robot using human natural instruction", Proc. of 8th IEEE International Workshop on Robot and Human Communication, pp171-176 (1999).

[Izard 83] C. Izard, L. Dougherty, and E.A. Hembree, "A System for Identifying Affect Expressions by Holistic Judgements", Technical Report, Univ. of Delaware, 1983.

[Karki 00] J. Karki, "Active low-pass filter design", Application Report, SLOA049A, Copyright 2000, Texas Instruments.

[Keisler 97] S. Keisler and L. Sproull, "Social Human Computer Interaction, Human Values and the Design of Computer Technology", B. Friedman, ed. CSLI Publications: Stanford, CA.:1997, 191-199.

[King 96] W. King and J. Ohya, "The representation of agents: Anthropomorphism, agency, and intelligence", In Proceedings of CHI-96, 1996

[Kirsch 99] D. Kirsch, "The Affective Tigger: Astudy on the construction of an emotionally reactive toy," M.S. dissertation, Prog. Media Arts Sci., Mass. Inst. Technol., Cambridge, 1999.

[Koda 96] T. Koda and P. Maes, "Agents with Faces: A Study on the Effect of Personification of Software Agents", Proceedings of the 5th IEEE International Workshop on Robot and Human Communication (RO-MAN 96), 189-194.

[Kozima 01] H. Kozima and H. Yano, "A robot that learns to communicate with human caregivers", in: Proc. Intl. Wksp. Epigenetic Rob., 2001.

[Kuo 82] B. C. Kuo, "Automatic Control Systems", 4th Ed., Copyright 1982, Prentice-Hall Inc.

[Lee 98] D. Lee, H. S. Seung, "A neural network based head tracking system", Advances in Neural Information Processing Systems 10, 908-14 (1998)

[Lee 99] D. Lee, L. Itti, C. Koch et al., Nat Neurosci, 1999; 2(4): 375-81.

[McKenna 99] S. J. McKenna, Y. Raja and S. Gong, "Tracking Colour Objects using Adaptive Mixture Models", Image and Vision Computing, Vol 17, No 3-4, 1999, 225-231

[Michaud 00] F. Michaud et al., "Artificial emotion and social robotics", in: Proc. Intl. Symp. Dist. Auton. Rob. Sys., 2000.

[Moeslund 01] T. B. Moeslund, E. Granum, "A survey of computer vision-based human motion capture", Computer Vision and Image Understanding, 18 (2001), 231-268.

[Moghaddam97] B. Moghaddam and A. Pentland, "Probabilistic Visual Learning for Object Representation", Pattern Analysis and Machine Intelligence, PAMI-19 (7), pp. 696-710, July 1997

[Niku 01] S. B. Niku, "Introduction to Robotics", Copright 2001, Prentice-Hall Inc.

[Norman 95] T. J. Norman, D. P. Long, "Goal Creation in Motivated Agents", In M. J. Wooldridge & N. R. Jennings (eds) Intelligent Agents: Proceedings of the First International Workshop on Agent Theories, Architectures and Languages, volume 890 of Lecture Notes in Artificial Intelligence, Springer-Verlag, pages 277-290, 1995.

[Ogata 00] T. Ogata and S. Sugano, "Emotional communication robot: WAMOEBA-2R emotion model and evaluation experiments", in: Proc. Intl. Conf. Humanoid Rob., 2000.

[Oliver 97] N. Oliver, A. Pentland, F. Berard, "LAFTER: Lips and Face Real Time Tracker With Facial Expression Recognition", Proceedings of Computer Vision and Pattern Recognition Conference, CVPR'97 , pp. 123-129

[Ortony 88] A. Ortony, G. L. Clore, A. Collins, "The Cognitive Structure of Emotions", Cambridge University Press, Cambridge, UK, 1988

[Raja 98] Y. Raja , S.J. McKenna, S. Gong, "Segmentation and Tracking using Colour Mixture Models" , Third Asian Conf. on Computer Vision, Hong Kong, China, 1998, Springer Verlag, pp. 606-614

[Reichard 78] J. Reichard, "Robots: Fact, Fiction, and Prediction", Penguin Books,1978.

[Reilly 96] W. Reilly, "Believable social and emotional agents", Ph.D. Thesis, Computer Science, Carnegie Mellon University, 1996

[Ridder 95] C. Ridder, O. Mukelt, H. Krichner, "Adaptive Background Estimation and Foreground Detection using Kalma-Filtering", Proceedings of International Conference on recent Advances in Mechatronics (ICRAM), pp. 193-199, 1995.

[Rizzo 99] P. Rizzo, "Emotional Agents for User Entertainment: Discussing the Underlying Assumptions", In Proceedings of the International Workshop on Affect in Interactions held in conjunction with the AC99, Annual Conference of the EC 13 Programme, 1999, Siena.

[Rowley 96] H. Rowley, S. Baluja, and T. Kanade, "Human Face Detection in Visual Scenes", Advances in Neural Information Processing Systems, 1996, pp. 875 - 881.

[Rowley 98] H. Rowley, Sh. Baluja, T. Kanade, "Neural Network based Face Detection", IEEE Transactions of Pattern Analysis and Machine Intelligence (TPAMI), vol.20, no.1, pp 23-38, 1998.

[Sakai 69] T. Sakai, M. Nagao and S. Fujibayashi, (1969). Line extraction and pattern detection in a photograph, Pattern Recognition, 1, 233-248.

[Scassellati 98] B. Scassellati, "Eye Finding via Face Detection for a Foveated, Active Vision System", AAAI-98, 1998.

[Scheeff 00] M. Scheeff et al., "Experiences with Sparky, a social robot," in Proc. Workshop Interactive Robot. Entertainment (WIRE), Pittsburgh, PA, Apr. 1-May 30 2000.

[Schulte 99] J. Schulte, C. Rosenberg, and S. Thrun, "Spontaneous short-term interaction with mobile robots in public places", In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 1999

[Sen 96] P. C. Sen, "Principles of Electric Machines and Power Electronics", 2nd Ed., Copyright 1996, John Wiley & Sons.

[Shigley 01] J. E. Shigley, Ch. R. Mischke, "Mechanical Design Engineering", 6th Ed., Copyright 2001, McGraw Hill

[Smith 97] C. Smith, H. Scott, "A computational approach to the meaning of facial expressions", in J. Russel and J. Fernandez-Dols, eg.s, "The phsychology of Facial Expression", Cambridge University Press, 1997 Cambridge, UK, pp. 229-254.

[Sung 95] K. Sung and T. Poggio, "Finding Human Faces with a Gaussian Mixture Distribution-based Face Model", In proceedings of the Second Asian Conference on Computer Vision (ACCV'95), 1995.

[Takeuchi 95] A. Takeuchi, and T. Naito, "Situated Facial Displays: Towards Social Interaction", Human Factors in Computing Systems: CHI'95 Conference Proceedings, ACM Press: NewYork, 1995, 450-455.

[Toschi 99] T. Doi citation, Excerpt from Computists' Weekly, AI Brief section in Intelligent Systems, December 1999

[Velasquez 98] J. Velasquez, "A computational framework for emotion-based control", in: Proc. Wksp. Grounding Emotions in Adap. Sys., Intl. Conf. SAB, 1998.

[Whissel 89] C. M. Whissel, The dictionary of affect in language, R. Plutchnik and H. Kellerman (Eds) "Emotion: Theory, research and experience: vol 4, The measurement of emotions", Academic Press, New York, 1989

[Wren 97] C. Wren, A. Azarbayejani, T. Darrell, A. Pentland, "Pfinder : Real-Time Tracking of the Human Body", IEEE Transactions on Pattern Analysis and Machine Intelligence, Jul 1997, Vol. 19, No.7, pp. 780-785

[Yang 02] M.H. Yang, N. Ahuja and D. Kriegman, "A Survey on Face Detection Methods", IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), vol. 24, no. 1, pp. 34-58, 2002

[Yang 96] J. Yang, L. Wu, and A. Waibel, "Focus of attention in video conferencing", CMU CS technical report, CMU-CS-96-150, 1996

[Yeshurun 98] Y. Yeshurun, M. Carrasco, Nature, 1998; 396(6706): 72-75

[Yuille 92] A. Yuille, P. Hallinan, and D. Cohen. Feature Extraction from Faces using Deformable Templates. International Journal of Computer Vision, 8(2):99--111, 1992.