

M. S. Ramaiah Institute of Technology
(Autonomous Institute Affiliated to VTU)
MSR Nagar, M.S.R.I.T Post, Bangalore - 560054

Project Entitled
**Cognitive Learning Assisted Robotic Arm
(CLARA)**

submitted in partial fulfillment for the award of degree of
**Bachelor of Engineering in
Electronics and Communication Engineering**

By

| | |
|------------------|------------|
| Naveen N. Murthy | 1MS09EC062 |
| Nitin J. Sanket | 1MS09EC069 |
| Raghunandana R. | 1MS09EC074 |
| Vyshak A. V. | 1MS09EC134 |

Under the guidance of
Prof. M. S. Srinivas



Department of Electronics and Communication
M. S. Ramaiah Institute of Technology

May 2013

M. S. Ramaiah Institute of Technology
(Autonomous Institute Affiliated to VTU)
MSR Nagar, M.S.R.I.T Post, Bangalore - 560054

Project Entitled
**Cognitive Learning Assisted Robotic Arm
(CLARA)**

submitted in partial fulfillment for the award of degree of
**Bachelor of Engineering in
Electronics and Communication Engineering**

By

| | |
|------------------|------------|
| Naveen N. Murthy | 1MS09EC062 |
| Nitin J. Sanket | 1MS09EC069 |
| Raghunandana R. | 1MS09EC074 |
| Vyshak A. V. | 1MS09EC134 |

Under the guidance of
Prof. M. S. Srinivas



Department of Electronics and Communication
M. S. Ramaiah Institute of Technology

May 2013

Department of Electronics and Communication
M. S. Ramaiah Institute of Technology
Bangalore



CERTIFICATE

This is to certify that the following students, who were working under our guidance, have completed their work as per our satisfaction with the topic **Cognitive Learning Assisted Robotic Arm (CLARA)**. To the best of our understanding the work to be submitted in this report does not contain any work, which has been previously carried out by others and submitted by the candidates for themselves for the award of any degree anywhere.

| | |
|------------------|------------|
| Naveen N. Murthy | 1MS09EC062 |
| Nitin J. Sanket | 1MS09EC069 |
| Raghunandana R. | 1MS09EC074 |
| Vyshak A. V. | 1MS09EC134 |

M. S. Srinivas
Professor

Dr. S. Sethu Selvi
Head Of Department (E & C)

Name of the examiners

- 1.
- 2.

Signature with Date

Department of Electronics and Communication
M. S. Ramaiah Institute of Technology
Bangalore



DECLARATION

We hereby declare that the entire work embodied in this project has been carried out by us at M. S. Ramaiah Institute of Technology under the guidance of Prof. M. S. Srinivas. This project has not been submitted in part or full for the award of any diploma or degree of this or any other University.

| | |
|------------------|------------|
| Naveen N. Murthy | 1MS09EC062 |
| Nitin J. Sanket | 1MS09EC069 |
| Raghunandana R. | 1MS09EC074 |
| Vyshak A. V. | 1MS09EC134 |

Acknowledgements

We are highly indebted to our project guide **Prof. M. S. Srinivas** for his guidance and constant support as well as for providing necessary information regarding the project.

We are grateful to our Head of Department **Dr. S. Sethu Selvi** for her encouragement and helpful insight which was extremely important to us. We are also thankful to **Asst. Prof. Sadashiva V. Chakrasali**, Department of Electronics & Communication, who helped us out with clarifications in the auditory part of our project. We would like to take this opportunity to thank the principal of MSRIT, **Dr. S. Y. Kulkarni**, for giving us this opportunity to showcase our ideas.

We remain deeply indebted to **David Lowe** for making a demo version of his patented SIFT code easily accessible, which helped us implement it in our application.

We would like to thank our colleagues in helping us develop the project and people who have willingly helped us out with their abilities. We would also take this opportunity to thank the VTU e-learning center for having given us an opportunity to present the template used to prepare the final year project using L^AT_EX.

Abstract

A longstanding problem in the field of robotic research has been to replicate the human ability to understand and learn from the environment around us. We have certainly come a long way in creating intelligent machines, but are yet to come up with a system which is intelligent enough to observe its surroundings and discover new things. We propose a Cognitive Learning Assisted Robotic Arm, which has a clean slate for a memory and mimics a baby's ability to constantly learn from its surroundings. This would go a long way in creating a generic platform which could train itself to perform various tasks in alien regions.

This work is dedicated to the world of RESEARCHERS,
TEACHERS, OPEN SOURCE COMMUNITY and OUR PARENTS.

“Science is a wonderful thing if one does not have to earn one’s living
at it.” - Albert Einstein

Contents

| | | |
|----------|---|-----------|
| 1 | Introduction | 1 |
| 2 | Problem Definition and Contributions | 3 |
| 3 | Literature Survey | 4 |
| 3.1 | Cognitive Systems | 4 |
| 3.1.1 | iCub | 4 |
| 3.1.2 | Nao | 5 |
| 3.2 | Audio Processing | 5 |
| 3.2.1 | Various algorithms used in speech processing . . | 5 |
| 3.3 | Object Detection and Feature Extractors | 7 |
| 3.3.1 | Scale Invariant Feature Transform (SIFT) . . . | 7 |
| 3.3.2 | Hue Histogram Descriptor | 8 |
| 4 | Proposed System | 10 |
| 4.1 | Overview | 10 |
| 5 | Ocular System | 12 |
| 5.1 | Choice of HSV Colorspace | 12 |
| 5.2 | Hand Detection | 14 |
| 5.3 | Saturation Threshold | 15 |
| 5.4 | Morphological Operations | 18 |
| 5.5 | Shadow Removal | 19 |

CONTENTS

| | | |
|-----------|--|-----------|
| 6 | Auditory System | 22 |
| 6.1 | Pause Detection | 23 |
| 6.2 | Word Segmentation | 25 |
| 6.3 | Feature Extraction | 26 |
| 6.4 | Speech Recognition | 28 |
| 7 | Cognitive Engine | 29 |
| 8 | Robotic Arm | 37 |
| 8.1 | Insight | 37 |
| 8.2 | Need for CLARA | 37 |
| 8.3 | Specifications | 38 |
| 8.4 | Mechanical Design | 39 |
| 8.5 | Electrical Design | 41 |
| 8.6 | Integration with ocular system | 44 |
| 9 | Alternate Designs | 46 |
| 9.1 | Stereoscopic Vision | 46 |
| 9.2 | Arm Designs | 47 |
| 9.2.1 | Conventional Arm | 47 |
| 9.2.2 | Universal Robotic Gripper | 48 |
| 10 | Components | 50 |
| 10.1 | Arduino UNO | 50 |
| 10.2 | List of Components and Tools | 50 |
| 11 | Potential Applications | 55 |
| 11.1 | Industrial Applications | 55 |
| 11.2 | Cognitive Platform | 56 |
| 11.3 | Hazardous Environments | 56 |
| 11.4 | Robotic Surgery | 56 |
| 11.5 | Space Exploration | 57 |
| 11.6 | Arm for an amputee | 58 |

CONTENTS

| | |
|---|-----------|
| 12 Future Work | 59 |
| 12.1 Wider range of objects | 59 |
| 12.2 Cluttered environment | 59 |
| 12.3 Increase in resolution of FOV | 60 |
| 12.4 Auto-calculation of servo values | 60 |
| A MATLAB to Arduino Serial Communication | 61 |
| B Servo Torque Ratings | 63 |
| C A Beginner's Guide to Arduino | 64 |
| D Servo Control using Arduino | 66 |
| E HD44780 LCD | 67 |
| F LUT of robotic arm | 69 |
| G Configuring an external webcam with MATLAB | 71 |
| H Addresses | 74 |
| Bibliography | 80 |

List of Figures

| | | |
|-----|--|----|
| 3.1 | Cognitive humanoid robotic platforms (a) iCub (b) Nao. | 4 |
| 3.2 | (a) Speech recognition system (b) Audio signal and its processed output. | 6 |
| 3.3 | Comparison between SIFT descriptor and Hue Histogram descriptor. | 8 |
| 4.1 | Proposed block diagram implemented in CLARA. . . . | 10 |
| 5.1 | Visual system used in CLARA. | 12 |
| 5.2 | Colorspaces (a) Sample image of a ball and its (b) Red (c) Green (d) Blue (<i>RGB</i>) (e) Hue (f) Saturation (g) Value (<i>HSV</i>) components. | 13 |
| 5.3 | Hand Detection (a) Hand portion seen in the frame (b) Output of YCgCr thresholding. | 15 |
| 5.4 | Scenarios involving (a) a single object (b) multiple objects. | 16 |
| 5.5 | Choosing a threshold (a) Saturation component (b) Histogram of (a) (c) Histogram after clustering (d) Output of thresholding. | 17 |
| 5.6 | Bounding box obtained using suitable morphological operations. | 18 |
| 5.7 | Shadow removal using $L^*a^*b^*$ colorspace ('a' component). | 20 |
| 5.8 | Some objects learned by CLARA and the words associated with them. | 21 |
| 6.1 | Auditory system implemented in CLARA. | 22 |

LIST OF FIGURES

| | | |
|------|--|----|
| 6.2 | Pause Detection (a) Original signal (b) Signal with pauses suppressed. | 23 |
| 6.3 | Word segmentation of the word ‘ <i>nothing</i> ’. | 25 |
| 6.4 | Original signal and Power Spectral Density of the words (a) ‘a’ (b) ‘this’. | 27 |
| 7.1 | Database built-up in CLARA over time. | 29 |
| 7.2 | Rotational invariance provided by SIFT. | 30 |
| 7.3 | Scenario involving an object already present and an unknown object. | 31 |
| 7.4 | Incorrect detection of objects. | 33 |
| 7.5 | Successful recognition of the word ‘ <i>yes</i> ’. | 36 |
| 8.1 | Human Arm. | 37 |
| 8.2 | A 4 DOF Robotic Arm. | 38 |
| 8.3 | Implemented 4 DOF Robotic Arm showing each DOF in a different color; Cyan - Base, Yellow - Shoulder, Red - Elbow, Green - Wrist. | 39 |
| 8.4 | Mechanical designs of the arm. | 40 |
| 8.5 | Snapshot of LM7805’s datasheet. | 41 |
| 8.6 | Electrical connections of CLARA. | 42 |
| 8.7 | Power Supply Board used to power CLARA. | 43 |
| 8.8 | Front Panel of CLARA. | 43 |
| 8.9 | Camera field-of-view (FOV) mapped onto a trapezoid. | 44 |
| 8.10 | Interface of robotic arm with the ocular system. | 45 |
| 9.1 | Stereoscopic camera design using two webcams. | 46 |
| 9.2 | Conventional design for fingers of a robotic arm. | 47 |
| 9.3 | Design of a conventional robotic arm. | 48 |
| 9.4 | (a) Universal Robotic Gripper (b) Rolling pump used to create vacuum. | 49 |
| 11.1 | Industrial robotic arms <i>Romeo</i> and <i>Juliet</i> | 55 |
| 11.2 | Miners working in hazardous environments. | 56 |
| 11.3 | Robotic surgery. | 57 |

LIST OF FIGURES

| | | |
|------|--|----|
| 11.4 | Robotic arm of Mars rover <i>Curiosity</i> | 57 |
| 11.5 | Luke Arm - An arm for amputees. | 58 |
| 12.1 | A few objects which CLARA cannot detect. | 59 |
| A.1 | Serial COM port used by Arduino. | 62 |
| B.1 | Determination of servo torque ratings. | 63 |
| C.1 | Pinout of Arduino UNO. | 65 |
| E.1 | Command set of HD44780 (See Ref. [62]). | 67 |
| E.2 | HD44780 LCD. | 68 |
| E.3 | Character set of HD44780. | 68 |
| F.1 | Trapezoidal FOV as seen by CLARA. | 69 |

List of Tables

| | | |
|------|---|----|
| 8.1 | Robotic Arm Specifications | 38 |
| 10.1 | List of Components used with their price and place of availability. Refer Appendix H for details. | 51 |
| 10.2 | List of Tools used with their place of availability. Refer Appendix H for details. | 54 |
| F.1 | LUT used by CLARA to navigate the trapezoidal FOV. | 70 |

Chapter 1

Introduction

Project CLARA takes inspiration from the human brain. A baby during the initial days of its life knows nothing. The baby sees things around it and asks its parents for information about the object it has seen. Later, the baby learns the things gathered from its parents and grows up. CLARA's working coincides with the working of the human brain.

Cognitive learning is defined as the acquisition of knowledge and skill by mental or cognitive processes, i.e. the procedures we have for manipulating information 'in our heads'. In cognitive learning, the system learns by listening, watching, touching, reading, or experiencing, and then processing and remembering the information. Cognitive learning differs a lot from conventional learning. Extensive research done in the past has proved that cognitive learning is faster and is a better form of learning compared to conventional learning. In cognitive learning, the learning takes place in the mind and not according to the conventional behavior exhibited. Cognition involves the formation of mental representations of the elements of a task and the discovery of how these elements are related. Cognitive learning stresses on the idea that learning comes about as a result of processes related to experience, perception, memory, as well as overtly verbal thinking and interaction with the user. It is a powerful mechanism that provides the means of knowledge, and goes well beyond simple imitation of others. Information processing is based on a learning mechanism that describes the processing of storage and retrieval of knowledge from the system's memory.

Cognitive learning might seem to be passive learning, because there is no motor movement. However, the learner is quite active, in a cognitive way, in processing and remembering newly incoming information. Since cognitive activity is involved in many aspects of human behavior, it might seem that cognitive learning only

takes place in human beings. However, we have developed a system to show that cognitive learning is not only limited to human behavior but it can be extended to machines and other mechanical devices, leading to reduced human intervention in tough environments.

Chapter 2

Problem Definition and Contributions

Cognitive Robotics is a field of extensive research and is the future towards which all fields are going to merge. Though extensive research is on-going in this field, a relatively low number of products employs such algorithms due to the high cost involved. Also, individuals and start-ups cannot work on cognitive robotics due to the high funding required which can be afforded by few entities such as the government and defence. We have tried to bridge this gap by building a low cost cognitive platform deployed on a robotic arm called CLARA. The key contributions of CLARA to the research community are:

- i. A low cost Cognitive platform to aid low budget research and development in developing countries like India.
- ii. Completely open source hardware design based on Arduino platform.
- iii. Completely open source based low cost arm prototype.
- iv. An efficiently working platform deploying object recognition based on SIFT and voice recognition based on power spectral density using Yule-Walker, both weaved together by a cognitive engine using a Situational reduction in dictionary, which substantially improves recognition rates.

Chapter 3

Literature Survey

3.1 Cognitive Systems

The present day world is moving towards cognitive learning systems. iCub and Nao are the leading ones in this regard. Project CLARA derives motivation from these two cognitive learning based robots.

3.1.1 iCub

iCub is a 1m high humanoid robot testbed for research into human cognition and artificial intelligence [1, 2, 3]. The iCub is a humanoid robotic platform, which is



(a)



(b)

Figure 3.1: Cognitive humanoid robotic platforms (a) iCub (b) Nao.

shown in Fig. 3.1a, developed at IIT (Istituto Italiano di Tecnologia) as part of the EU project RobotCub and subsequently adopted by more than 20 laboratories worldwide [4, 5]. It has 53 motors that move the head, arms and hands, waist, and legs. It can see and hear, it has the sense of proprioception (body configuration) and movement (using accelerometers and gyroscopes). Work to give the iCub the sense of touch and to grade how much force it exerts on the environment is still in process.

3.1.2 Nao

One other such advanced robot is Nao, shown in Fig. 3.1b. Nao is an autonomous, programmable humanoid robot developed by Aldebaran Robotics, a French startup company headquartered in Paris [6]. The robot's development began with the launch of Project Nao in 2004. On 15 August 2007, Nao replaced Sony's robot dog Aibo as the robot used in the Robot Soccer World Cup (Robocup) Standard Platform League (SPL), an international robotics competition. The Nao was used in RoboCup 2008 and 2009, and the NaoV3R was chosen as the platform for the SPL at RoboCup 2010.

The cost involved in the R&D of these systems is very huge. Each robot costs about 250,000 Euros, which is very expensive. So, the need of the hour is to produce similar systems at a lower price so that it can be used by the society. One such attempt made by us is CLARA.

3.2 Audio Processing

3.2.1 Various algorithms used in speech processing

1. A computer system is described in which isolated words, spoken by a designated talker, are recognized through calculation of a minimum prediction residual [7]. A reference pattern for each word to be recognized is stored as a time pattern of linear prediction coefficients (LPC). The total log prediction residual of an input signal is minimized by optimally registering the reference LPC onto the input autocorrelation coefficients using the dynamic programming algorithm (DP). The input signal is recognized as the reference word which produces the minimum prediction residual. A sequential decision procedure is used to reduce the amount of computation in DP. Frequency normalization with respect to the long-time spectral distribution is used to reduce effects of variations in the frequency response of telephone connections. The system has been implemented on a DDP-516 computer for the 200-word recognition experiment. The recognition rate for a designated male talker is 97.3 percent for telephone input, and the recognition time is about 22 times real time.

2. The recognition of speech has been thought to require frequency-specific (spectral) cues. Spectral energy peaks in speech reflect the resonant properties of the vocal tract and thus provide acoustic information on the production of speech sound. Spectral information was removed from speech by replacement of the frequency specific information in a broad frequency region with a band limited noise. The acoustic signal was divided into several frequency bands and the amplitude envelope was extracted from each band by half wave rectification and low pass filtering [8].
3. Spoken utterances are represented by time sequences of cepstrum coefficients and energy. Regression coefficients for these time functions are extracted for every frame over an approximately 50 ms period. Time functions of regression coefficients extracted for cepstrum and energy are combined with time functions of the original cepstrum coefficients, and used with a staggered array DP matching algorithm to compare multiple templates and input speech [9].
4. For speech processing both deterministic and stochastic models have shown good results. One type of stochastic model is hidden Markov model. The HMM model is characterized by:
 - N, the number of states in the model
 - M, the number of distinct observation symbols per state
 - The state transition probability $A = \{a_{ij}\}$
 - The observation symbol probability distribution in state j, $B = \{b_j(k)\}$
 - The initial state distribution $\pi = \{\pi_j\}$

Given appropriate values of N, M, A, B and π , the HMM can be used to generate an observation sequence. The parameters in this sequence are observed to create the best models for real world applications [10].

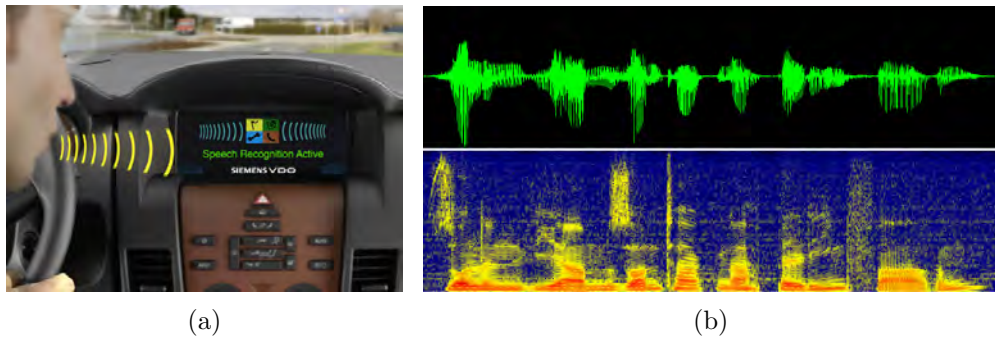


Figure 3.2: (a) Speech recognition system (b) Audio signal and its processed output.

A sample speech recognition system is shown in Fig. 3.2a, while Fig. 3.2b shows a speech sample and the output of the processing system.

3.3 Object Detection and Feature Extractors

Object detection and segmentation are the most important and challenging fundamental tasks of computer vision [11]. It is a critical part of many applications such as image search, image auto-annotation and scene understanding. However, it is still an open problem due to the complexity of object classes and images.

The easiest way to detect and segment an object from its background is using color based methods. The colors in the object and the background should have a significant difference in order to segment objects successfully using color based methods.

3.3.1 Scale Invariant Feature Transform (SIFT)

SIFT keypoints of objects are first extracted from a set of reference images and stored in a database. An object is recognized in a new image by individually comparing each feature from the new image to this database and finding candidate matching features based on Euclidean distance of their feature vectors. From the full set of matches, subsets of keypoints that agree on the object and its location, scale, and orientation in the new image are identified to filter out good matches. The determination of consistent clusters is performed rapidly by using an efficient hash table implementation of the generalized Hough transform. Each cluster of 3 or more features that agree on an object and its pose is then subject to further detailed model verification and subsequently outliers are discarded. Object matches that pass all these tests can be identified as correct with high confidence.

Lowe's method for image feature generation transforms an image into a large collection of feature vectors, each of which is invariant to image translation, scaling, and rotation, partially invariant to illumination changes and robust to local geometric distortion [12, 13]. These features share similar properties with neurons in inferior temporal cortex that are used for object recognition in primate vision [14]. Indexing consists of storing SIFT keys and identifying matching keys from the new image. Lowe used a modification of the k-d tree algorithm called the Best-bin-first search method that can identify the nearest neighbors with high probability using only a limited amount of computation [15].

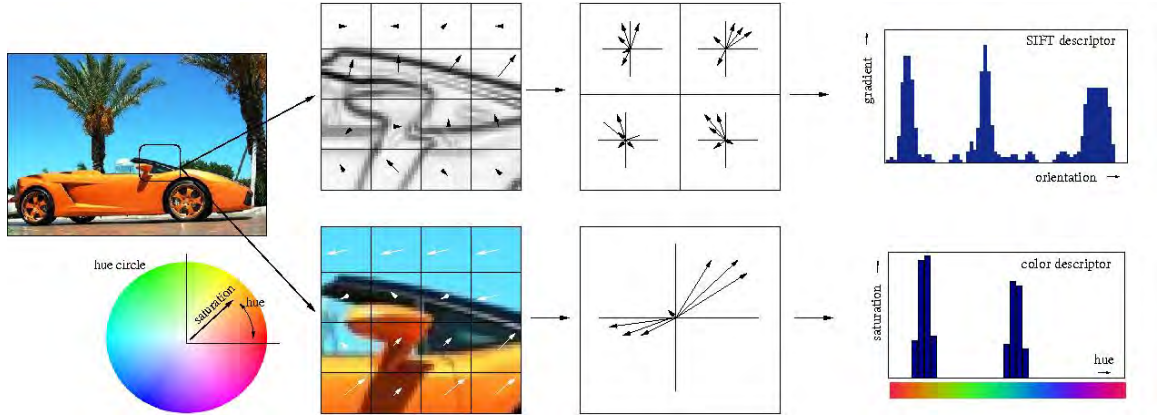


Figure 3.3: Comparison between SIFT descriptor and Hue Histogram descriptor.

Hough Transform is used to cluster reliable model hypotheses to search for keys that agree upon a particular model pose. Hough transform identifies clusters of features with a consistent interpretation by using each feature to vote for all object poses that are consistent with the feature.

SIFT-based descriptors which are region-based, are the most robust and distinctive, and are therefore best suited for feature matching. A recent feature descriptor named as SURF (Speeded Up Robust Features), which is faster than SIFT, can also be used [16]. However, SIFT is generally found to be more robust than SURF when speed is not critical [17].

SIFT features can essentially be applied to any task that requires identification of matching locations between images. Work has been done on applications such as recognition of particular object categories in 2D images, 3D reconstruction, motion tracking and robot localization. In robot localization a trinocular stereo system is used to determine 3D estimates for keypoint locations. Keypoints are used only when they appear in all 3 images with consistent disparities, resulting in very few outliers. As the robot moves, it localizes itself using feature matches to the existing 3D map, and then incrementally adds features to the map while updating their 3D positions using a Kalman filter. This provides a robust and accurate solution to the problem of robot localization in unknown environments [18].

3.3.2 Hue Histogram Descriptor

The SIFT extractor described previously does not take the color information of the object into account. Color of an object can also be a distinctive feature [19]. However, it is not extensively used since it is susceptible to changes in lighting

3.3 Object Detection and Feature Extractors

conditions and varying image quality. Instead of using color as a stand-alone distinguishing feature, it can be used to supplement shape and texture information obtained using other descriptors such as SIFT.

In Ref. [20] Van de Weijer et al. describe a feature extractor utilizing the local color content of the image. RGB colorspace is not very robust to variations in illumination and hence, HSV colorspace is considered. The color descriptor of the object is concatenated with its shape descriptor. The hue values occurring in the image are separated into evenly spaced bins and the resulting histogram is stored in the database. A Hue Histogram descriptor of an object is shown in Fig. 3.3 along with the SIFT descriptor of the same.

Chapter 4

Proposed System

CLARA (Cognitive Learning Assisted Robotic Arm) is an automated system which has the ability to interact with its environment and learn accordingly. Possessing a clean slate for a memory, it *remembers* objects that it encounters, and can then be trained to perform specific actions when it comes across those objects again.

4.1 Overview

The various processes involved in the training of the system and the selection of a particular action are illustrated in Fig. 4.1. A human stores information about an object in two ways: the visual content of the object and the word associated with

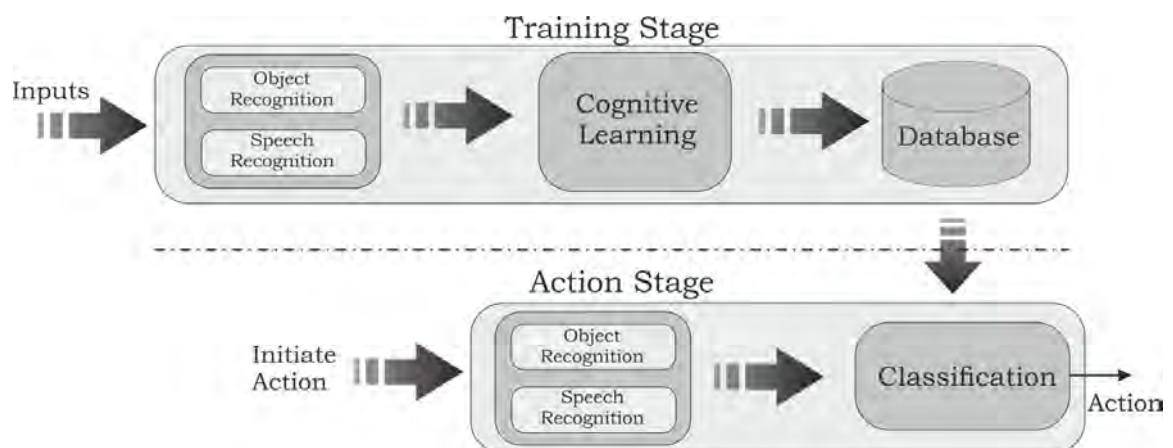


Figure 4.1: Proposed block diagram implemented in CLARA.

it. In a similar manner, CLARA observes its environment for a hitherto unknown object. Once a new object is detected, the system records information with respect to the visual representation of the object. CLARA is then trained with voice samples of the word associated with the object. A database is created in which the voice samples are tagged to the particular object. CLARA is also trained with samples of a few possible actions such as **pick**, **show**, **nothing** etc. and two words which form the basis of its cognitive learning: **yes** and **no**.

The system continuously monitors its surroundings and when it detects a group of objects, it is ready to perform the action desired by the user. A command spoken by the user is deciphered to determine the action to be performed and the object on which it is to be done. This is done by comparing the user's command with the voice samples already stored in the database, and CLARA executes the desired function.

Chapter 5

Ocular System

The Video Processing Unit of CLARA or in other words, its *eye*, has to continually watch the scene in front of it and, if an object is placed in front of it, has to separate it from the background. Another important aspect is that the system has to neglect those frames which are hindered by factors such as the user's hand or the improper placement of the object, and must only process those frames in which the objects are completely in its field of view without any obstructions. An overview of the various techniques used for this purpose is shown in Fig. 5.1.

5.1 Choice of HSV Colourspace

Clutter is one of the major issues impeding proper object detection. It is possible to detect a known object which is lying in a cluttered environment but is extremely difficult to learn about a new object in such circumstances. The reason for this

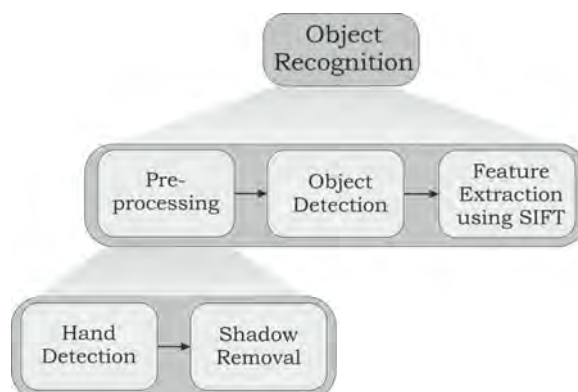


Figure 5.1: Visual system used in CLARA.

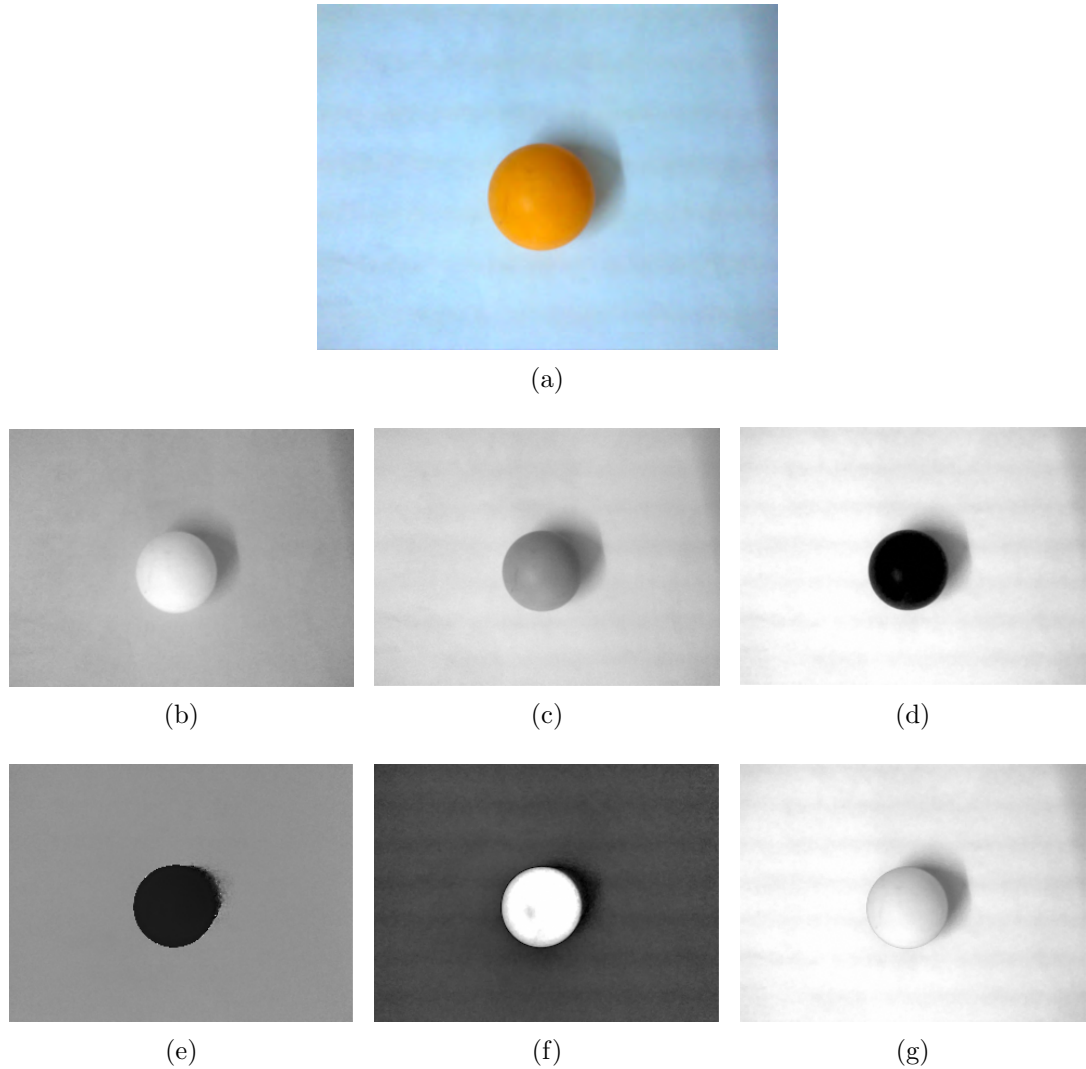


Figure 5.2: Colorspaces (a) Sample image of a ball and its (b) Red (c) Green (d) Blue (*RGB*) (e) Hue (f) Saturation (g) Value (*HSV*) components.

is that, since we do not know what the object looks like, separating it from the background becomes a herculean task. Thus, we have chosen to have a white background since it reduces the complexity involved in object detection.

After zeroing in on a white background, the choice of the colorspace used is extremely critical since we need a characteristic property to distinguish between the object and the scene around it. Fig. 5.2 shows a sample frame containing a yellow ball, and the various components of its RGB (Red, Green, Blue) and HSV

(Hue, Saturation, Value) colorspace. The primary feature of a good visual system is color-independence, i.e. the component chosen should behave in almost the same way irrespective of the color of the object. Thus, the **red**, **green** and **blue** components shown in Figs. 5.2b, 5.2c and 5.2d are not useful owing to their rich color content. The **hue** component shown in Fig. 5.2e is also ruled out since it represents color content in HSV colorspace.

The **saturation** component shown in Fig. 5.2f is promising since the white background is unsaturated (very low color content) and hence, a lot of distinction can be seen between the highly saturated object and its background. We are making use of the property that **saturation** is a measure of how much color is contained in the image and hence, perfect grayscale images (no color content) are unsaturated [21].

Another reason to choose the **saturation** component is its inherent property of illumination-invariance. Under slight illumination variations, the white background appears to be a little gray; however, it remains unsaturated and thus, the **saturation** component is not affected much. This provides the system with robustness to illumination, to a great extent.

5.2 Hand Detection

The training of the system is done dynamically, i.e. an object is brought into its field of view, trained and then replaced dynamically with another object. In the process of picking up the object, the hand of the user appears in the field of view and hinders detection or sometimes causes detection of false objects. Thus, we apply a **skin detection** algorithm to ascertain if the current frame contains a portion of a hand, and if so, we move on to the next frame until the actual object is obtained without obstructions. A few popular skin detection algorithms are explained in detail in Refs. [22, 23]. Fig. 5.3 shows a frame containing a portion of a hand and the output of skin thresholding.

The hand consists of skin areas and YCgCr thresholding is applied to detect the skin areas in the image [24]. The YCgCr color space is a modification of YCbCr color space in which the blue-difference chroma component ($B-Y$) is replaced by the green difference chroma component ($G-Y$). In the YCbCr color space Y is the luminance component, Cb and Cr are the blue and red difference chroma components respectively. YCgCr components can be procured from RGB color model using the matrix equation given in Eq. 5.1.

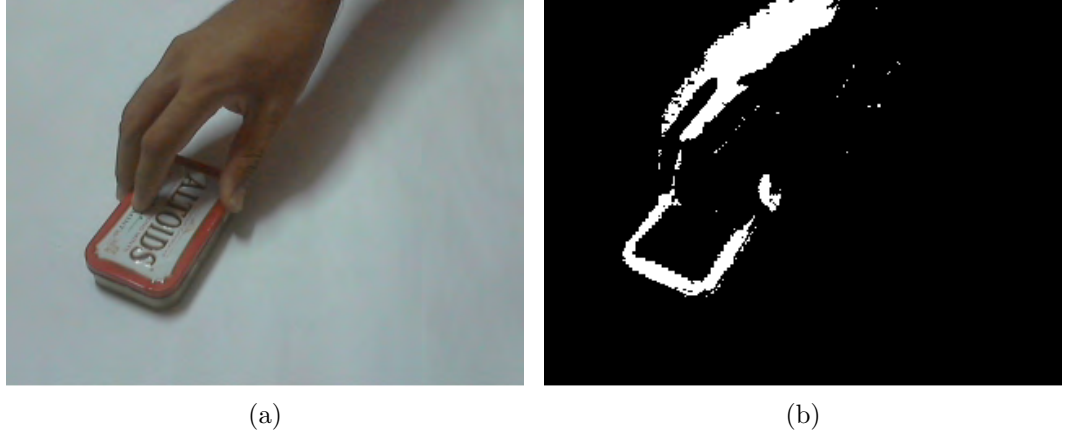


Figure 5.3: Hand Detection (a) Hand portion seen in the frame (b) Output of YCgCr thresholding.

$$\begin{bmatrix} Y \\ Cg \\ Cr \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 65.481 & 128.553 & 24.966 \\ -85.085 & 112 & -30.915 \\ 112 & -93.786 & -18.214 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (5.1)$$

A message called “Hand Seen” pops up on the screen if at all the system detects any portion of skin. This operation is performed monotonously and the message is popped up until a clear image of the object is obtained.

An important aspect to be considered here is that the object itself might have characteristics similar to the human skin. In such a case, the object itself appears in the YCgCr thresholded image, as seen in Fig. 5.3b. To avoid false detection of a hand, we check only the outer boundary of the YCgCr thresholded image and not the entire image. If the image contains only the object and not the portion of a hand, it lies entirely in the field of view and hence will not be detected wrongly as a hand.

5.3 Saturation Threshold

After choosing the saturation component to work with, we have to choose a particular threshold value, called the **saturation threshold**, which separates the objects from the background. Ideally, the background is plain white and it has a very low saturation value. However, due to various factors such as shadows, extreme

5.3 Saturation Threshold

illumination variations etc., the saturation content of the background is not constant and hence, has to be deduced differently as per the situation. Consider two different scenarios : one involving a single object and the other consisting of multiple objects, as illustrated in Fig. 5.4. The various steps involved in determining the **saturation threshold** are shown in Fig. 5.5.

Consider the saturation components of both images, shown in Fig. 5.5a. Though both backgrounds are the same, i.e. a white sheet, the saturation content in the backgrounds of both images varies owing to different ambient conditions. Thus, the algorithm must be robust enough to account for these differences in saturation. Fig. 5.5b shows the histogram of the saturation component. As expected, a peak is obtained corresponding to the saturation value of the background, since it occupies a large area. It is easy for us to identify this peak visually but, due to the presence of local maxima, it is difficult for the system to detect this peak. The histogram needs to be flattened, i.e. the local variations around the peak need to be suppressed in order to arrive at a clustered histogram, shown in Fig. 5.5c. This is done by moving an overlapping window over the histogram, and assigning the maximum value of the window to the central index. The size of the window is arbitrary and should be determined depending on the ambient conditions.

It can be seen in Fig. 5.5c that, the uneven peaks in the original histogram have been leveled and this helps in identifying the background clearly. The flat peak with the maximum value, in both cases, corresponds to the background while the other cluster corresponds to the objects. *It has to be noted that the number of clusters **may not** correspond to the number of objects, since the saturation content of different objects might be the same.* The average of the locations of the two



Figure 5.4: Scenarios involving (a) a single object (b) multiple objects.

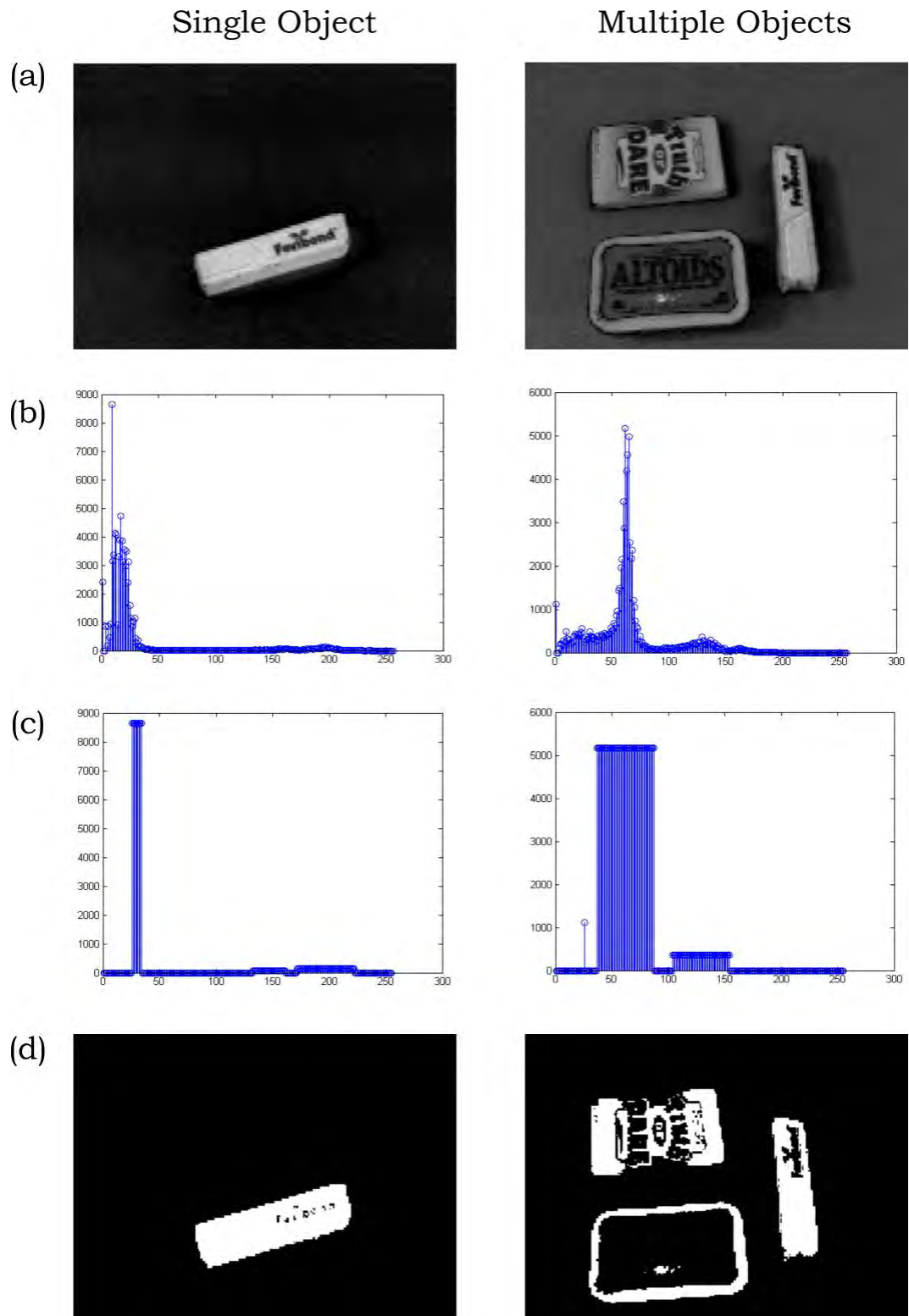


Figure 5.5: Choosing a threshold (a) Saturation component (b) Histogram of (a) (c) Histogram after clustering (d) Output of thresholding.

clusters gives the desired threshold, as detailed in Eq. 5.2.

$$\textit{saturation threshold} = \textit{mean}(S_{bg}, S_{obj}) \quad (5.2)$$

where *saturation threshold* is the desired threshold, while S_{bg} and S_{obj} are the saturation values of the clusters corresponding to the background and object in the histogram shown in Fig. 5.5c respectively.

Fig. 5.5d shows the result of thresholding using the obtained **saturation threshold** for both scenarios. It can be seen that the calculated threshold has separated the objects from the background, but this is not the final resultant image. It lays ground for further morphological operations to obtain each object present in the scene.

5.4 Morphological Operations

The output of thresholding shown in Fig. 5.5d is not sufficiently clear to obtain a bounding box around the detected objects. Appropriate morphological operations need to be performed in order to distinguish the objects from their background [25]. Fig. 5.6 shows the thresholded output of a frame and the morphological

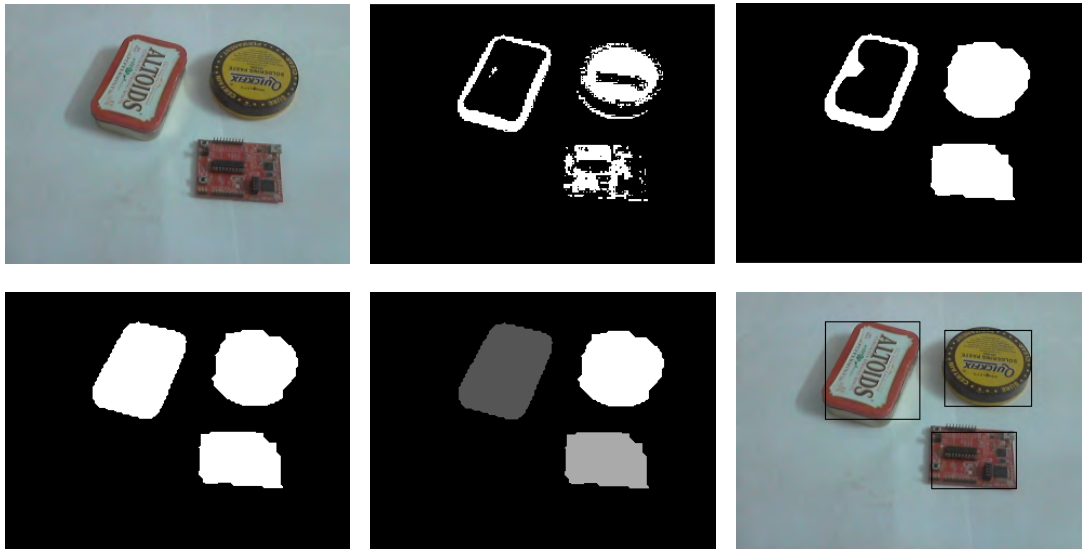


Figure 5.6: Bounding box obtained using suitable morphological operations.

operations performed to get an almost perfect bounding box around each of the objects. MATLAB provides a variety of morphological techniques to obtain the desired result. Some operations used by are given below:

i. *imclose*

It performs morphological closing on the binary image. The morphological close operation is a dilation followed by an erosion, using the same structuring element for both operations.

ii. *imfill*

This fills holes in the binary image. A hole is a set of background pixels that cannot be reached by filling in the background from the edge of the image.

iii. *bwareaopen*

This morphologically opens the binary image, i.e. removes small objects.

iv. *bwlabel*

This labels connected components in a binary image. The output contains the detected objects having different tags or labels, as illustrated in Fig. 5.6 with different colors.

After performing the above morphological operations, we obtain the objects in a contiguous manner, not hindered by factors such as disconnected components or holes. Then, a MATLAB function called *regionprops* is used to obtain the bounding box around each object.

5.5 Shadow Removal

The choice of a proper threshold and the application of appropriate morphological operations do not necessarily lead to successful object detection. The shadow of the object plays a critical role in detection. Under certain conditions, the shadow has saturation characteristics similar to the object, and hence, the threshold groups the shadow with the object.

Why is the shadow detrimental to the performance of the system? The answer lies in the fact that we want the object to be represented by the features of itself and not of its shadow. Prior to feature extraction and matching, if the shadow is not removed, it provides its own features which are included in the descriptor of the object and this has to be avoided. A robust algorithm for real-time shadow removal

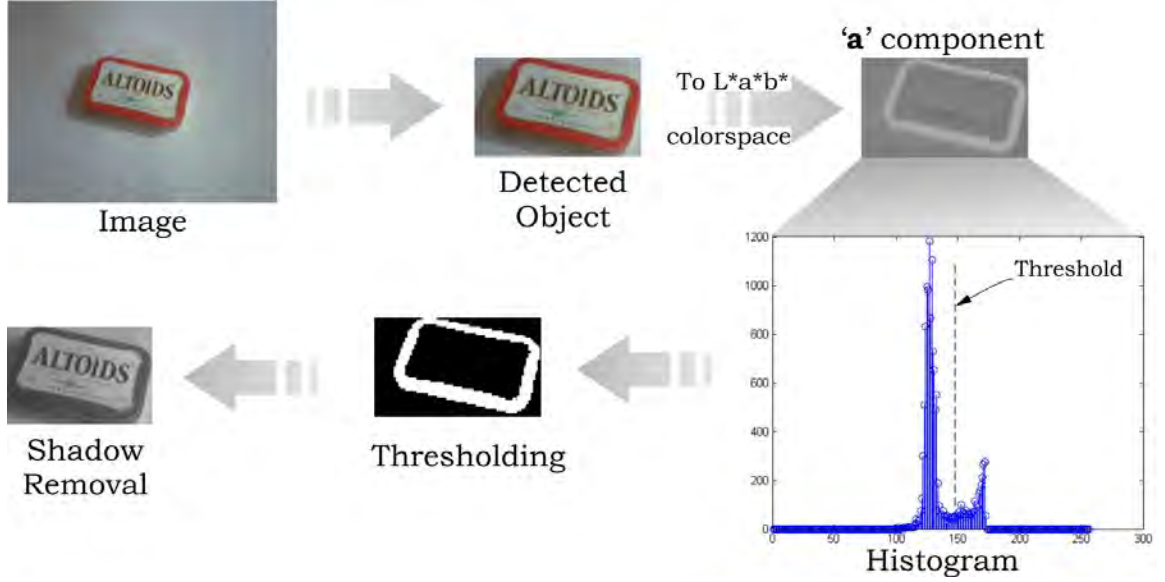


Figure 5.7: Shadow removal using $L^*a^*b^*$ colorspace ('a' component).

is given in Ref. [26].

The $L^*a^*b^*$ colorspace was proposed in 1976 by the Commission Internationale d'Eclairage (CIE) [27]. Here, L defines the lightness of the color, and a and b define the color along a red/green and blue/yellow axis, respectively. We have used the a component for shadow removal, the steps of which are illustrated in Fig. 5.7.

The detected object initially has its shadow associated with it, due to the prevailing ambient conditions. In the $L^*a^*b^*$ colorspace, and specifically the a component, it can be seen that the shadow merges with the background and can be removed. The tricky part is to obtain the threshold dynamically from its histogram, instead of using a pre-defined value. This is calculated as the mean of the two peaks in the histogram, and is shown as a dotted line in Fig. 5.7. By using this threshold, the shadow can be eliminated and a suitable bounding box is applied to obtain the object finally. Fig. 5.8 shows a few objects used by us to train CLARA and the words associated with each of them.



Figure 5.8: Some objects learned by CLARA and the words associated with them.

Chapter 6

Auditory System

Now that we have understood how CLARA sees its environment, in the previous chapter, let us see the way in which it undertakes the task of hearing. The various steps involved in its auditory system are detailed in Fig. 6.1. *Speech recognition* refers to the process of understanding someone's speech by recognizing the words, while *speaker recognition* is the process of identifying the speaker. We are concerned with achieving robust *speech recognition* rather than *speaker recognition*, since the person who trains the system is more likely to use it as well. Having said that, CLARA did demonstrate a reasonably high rate of *speaker recognition* as well, by understanding the words of multiple users. An existing speech recognition system is explained in Ref. [28].

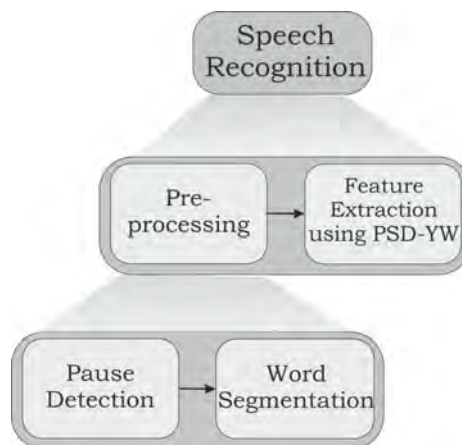


Figure 6.1: Auditory system implemented in CLARA.

The auditory system of CLARA is implemented in four stages.

- i. Pause Detection
- ii. Word Segmentation
- iii. Feature Extraction
- iv. Speech Recognition

6.1 Pause Detection

This stage is concerned with removing the pauses between the words spoken by the user. All the information content of the user's speech is contained in the words spoken by him and if the pauses between them are not removed, they introduce erroneous information. Fig. 6.2 illustrates a signal representing a sentence of four words, and the signal resulting from pause detection and suppression.

Pause detection implemented in CLARA has two inherent advantages :

- By suppressing the pauses, we are reducing the effect of noise in our signal processing algorithm.

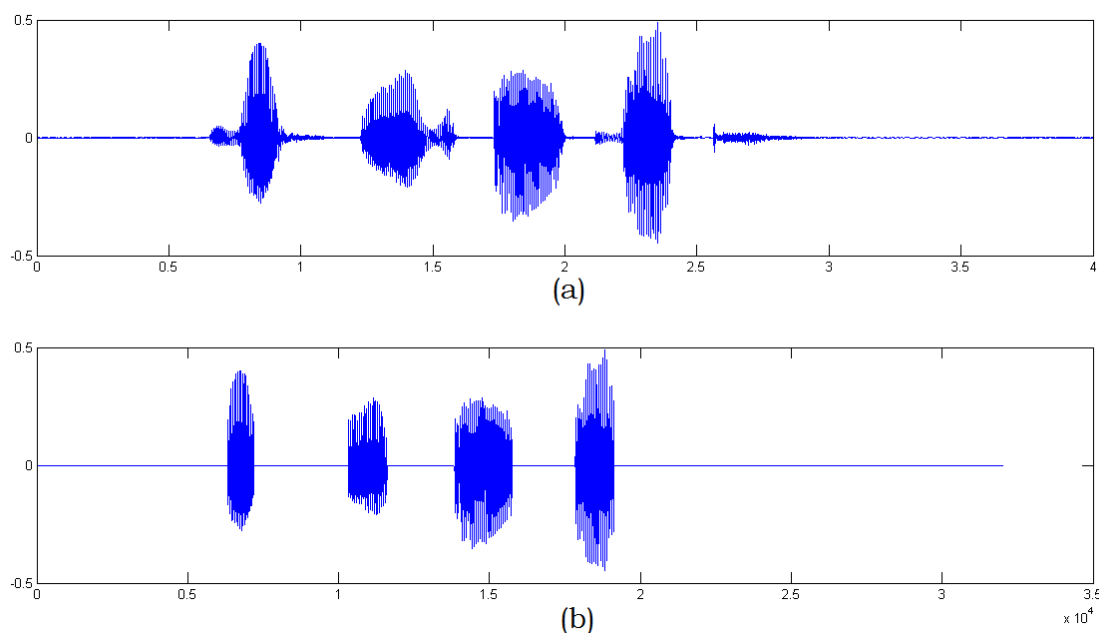


Figure 6.2: Pause Detection (a) Original signal (b) Signal with pauses suppressed.

- Also, pauses are generally long in duration and by removing them, we reduce the processing time of the auditory system.

An important motivation to perform pause detection is that, ironically, *silence* is a big part of a person's speech. This can be seen first-hand in Fig. 6.2, where the four words occupy a lesser duration in comparison with the silence before, after and between the words. If this signal is taken as the raw input to the auditory system, unnecessary processing is performed on the non-speech parts of the signal. Thus, a robust pause detection stage is a part of any good auditory system.

The underlying principle of pause detection is to threshold the given signal at a level which separates the required signal, i.e. spoken words, from the undesired silence or pauses [29]. This threshold cannot be a pre-defined constant since it depends on the ambient noise to a large extent. The relation used to determine the threshold dynamically from the given signal is given in Eq. 6.1.

$$T = \bar{N} + \left[1 - \frac{N_{min}}{N_{max}} \right] \bar{S} \quad (6.1)$$

where T is the chosen threshold, \bar{N} is the mean energy of noise or in other words, the mean energy of a pause, \bar{S} is the mean energy of the input signal, N_{min} and N_{max} are the minimum and maximum noise energy values of the detected pause [30].

Algorithm

```

for i = 1 to NumSamples:
    if S[i] < T
        S[i] = 0;
    end
end

```

Here, *NumSamples* represents the total samples present in the input signal S and T denotes the threshold determined using Eq. 6.1.

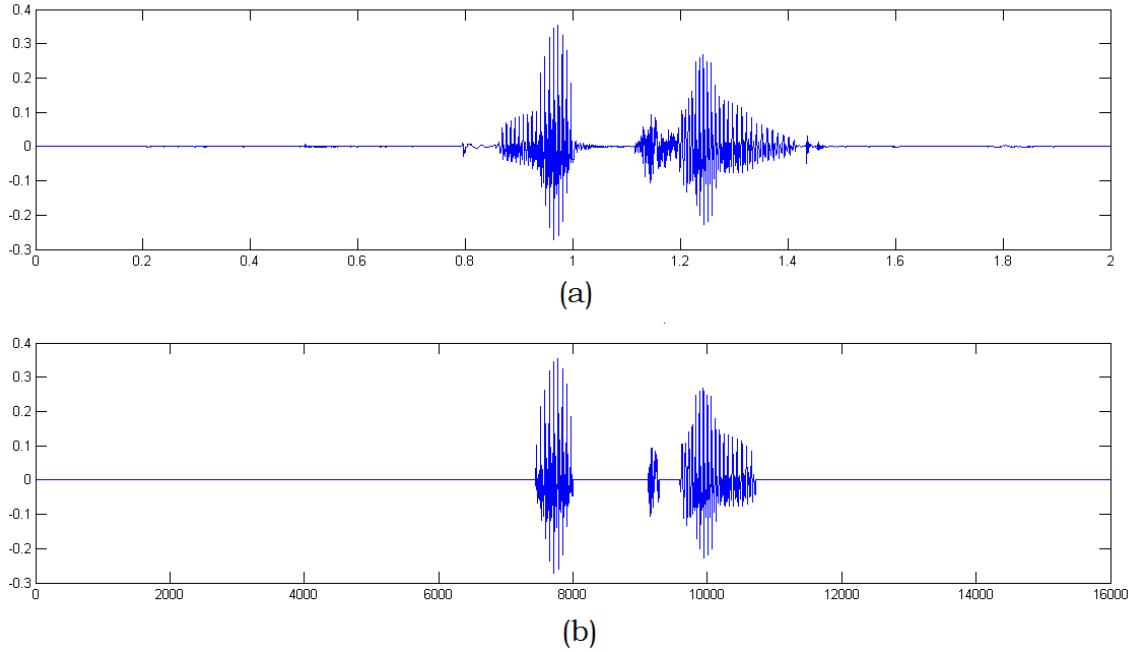


Figure 6.3: Word segmentation of the word ‘*nothing*’.

6.2 Word Segmentation

After detecting the pauses in the input, the next thing to be done is to segment the signal into its constituent words, i.e. we need to obtain the first and last indices of each word in the signal. While this appears to be a simple task of finding non-zero samples in the signal, it is further complicated by the fact that *there are samples of zero value not just between words but also within each word*.

Initially, we remove the zero value samples at the beginning of the sentence and at the end of it. Then, a *non-overlapping* window, with a window size that depends on the average duration of pauses between words (chosen by us as 200), is passed over the signal. This window is moved over the signal until there exists at least one non-zero sample in the window. When the window contains all samples of zero value, it implies the end of the word, and the ending index of the word is obtained from the previous position of the window.

The technique explained above segments the given input into words by removing the pauses entirely. However, in a few exceptions, the algorithm does not work in the intended way. One such example is shown in Fig. 6.3. The original signal contains two parts though it represents a single word *nothing*. This is due to the way in

which we say *nothing*: there is a pause-like quality in the middle of the word. After applying the word segmentation algorithm explained above, the signal is thought to be containing multiple words instead of just *nothing*.

To counter this problem, we have considered the beginning and ending indices of each segmented word. We have chosen a threshold such that, if the duration between two segmented words is less than a pre-defined value, then the two words are combined into one.

Algorithm

```

for i = 1 to (NumWords - 1) :
    if (Words[i+1].begin - Words[i].end) < Threshold
        combine(Words[i], Words[i+1]);
        i = i + 1;
    end
end

```

where,

$Words[i]$ indicates the i^{th} word of the segmented word list $Words$,
 $NumWords$ denotes the number of segmented words,
 $Threshold$ is the pre-defined pause duration (measured in number of samples) below which it is deemed to be a single word,
 $Words[i].begin$ and $Words[i].end$ denote the beginning and ending indices of the i^{th} word,
 $combine(Words[i], Words[i+1])$ is a function which combines the i^{th} and $(i + 1)^{th}$ words, forming a single word.

6.3 Feature Extraction

The segmented words contain just samples and cannot be directly used for speech recognition for two reasons :

- It is highly susceptible to noise.
- It contains a lot of local variations which hinder effective classification.

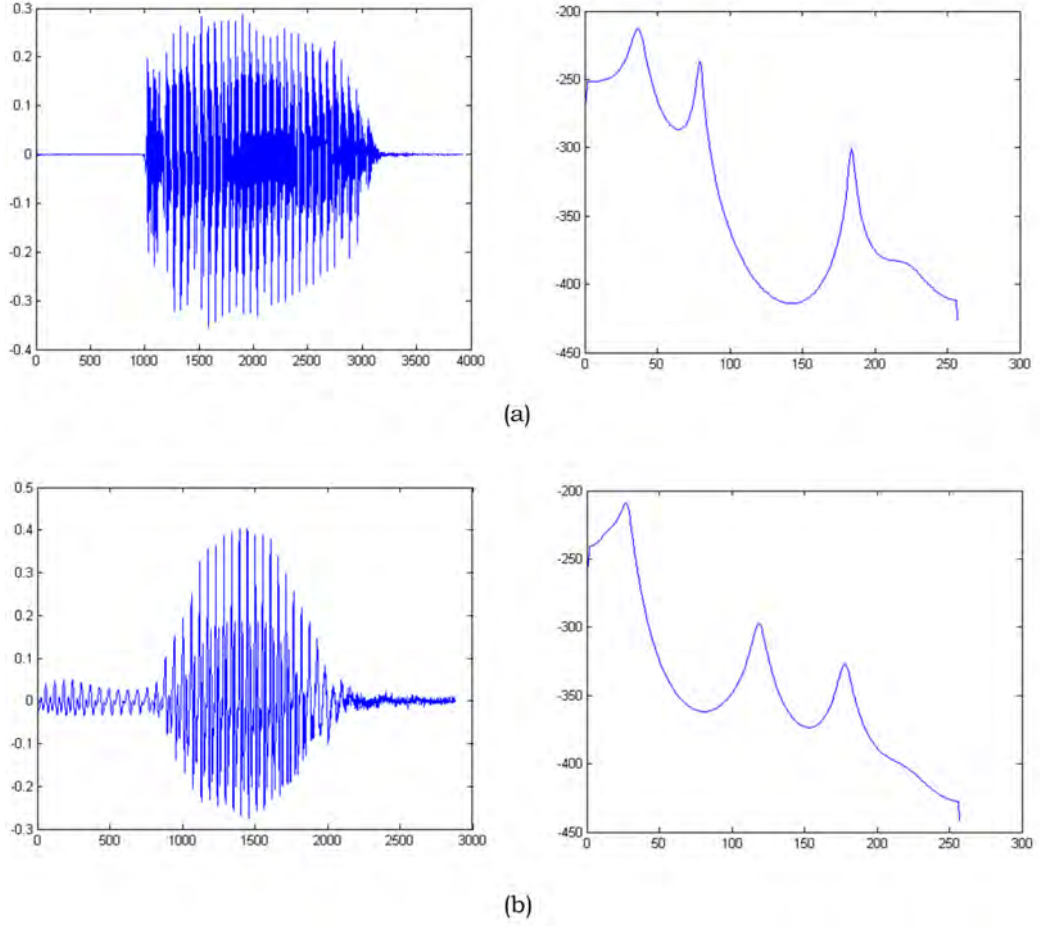


Figure 6.4: Original signal and Power Spectral Density of the words (a) ‘a’ (b) ‘this’.

For the above reasons, we have to find a suitable representation for the words which would be robust to factors such as noise and different ambient conditions. One such representation is **Power Spectral Density (PSD)** of the signal, which is calculated in units of power per radians per sample and could be determined in many possible ways. We have chosen to compute PSD of a signal using the **Yule-Walker algorithm**, for which we have used the function *pyulear()*.

The function *pyulear()* implements the Yule-Walker algorithm, a parametric spectral estimation method, and returns an estimate of the PSD of the input signal [31]. The input signal is nothing but a vector containing samples of a discrete-time signal, which in our case is an audio signal. The PSD obtained using the Yule-Walker algorithm is also an estimate of the maximum entropy contained in the signal. In general, the length of the FFT and the values of the input signal determine

the length of PSD of the signal and the range of the corresponding normalized frequencies. The default length of PSD using Yule-Walker is 256.

The PSD representations of two words ‘a’ and ‘this’ are shown in Fig. 6.4, along with their original signals. It can be seen that the wide variations of the input signal are mapped onto a steady PSD curve. Each word has a characteristic quality inherent to it, and this is seen in the positions of the *peaks* and *troughs* in the PSD curves. During training, CLARA stores the 256 valued descriptors of samples of each word in its auditory database.

6.4 Speech Recognition

The raw input signal, i.e. the sentence spoken by the user, is first segmented into words and represented using the PSD with Yule-Walker algorithm. This representation is suitable for comparison since it contains characteristic peaks and troughs for each word, *which occupy the same position for different instances of the word*. Also, the variations in the 256 valued descriptor are steady and not sudden, thereby aiding the classifier used.

Whenever the user issues a command, the sentence is segmented into words and its feature descriptor is compared with the descriptors stored in the auditory database. For efficient comparison, the choice of a good classifier is a must. We have used the Euclidean classifier in CLARA because of its simplicity and low cost of processing.

The measure of distance used here, to match the test sample to a word in the database is the Euclidean distance (straight-line distance) and hence, it is called the Euclidean classifier [32]. Let V_S and V_T be the feature vectors of the database sample and the test sample of a word respectively. Then, the Euclidean distance D is calculated using the relation given in Eq. 6.2.

$$D = \sqrt{\sum_{d=1}^M (V_S - V_T)^2} \quad (6.2)$$

where M is the number of features (256 in our case). The sample in the database corresponding to the least Euclidean distance gives the matched word in the database, and in this way the sentence is deciphered word-by-word.

Chapter 7

Cognitive Engine

The previous chapters have explained the visual and auditory systems present in CLARA. However, the database that is maintained by the system has not been touched upon. This database, shown in Fig. 7.1, is initially empty since the system knows nothing. Over time, as it begins to explore its surroundings and encounter new objects, it stores information about them in its database.

An important aspect to be kept in mind is that CLARA is trained dynamically, i.e. it is **not** one of those systems which are trained with objects for a pre-defined period initially, and only then are they tested with scenarios mimicking the real world. CLARA performs its training on the go, and hence, there is no necessity of a separate period involving offline training.

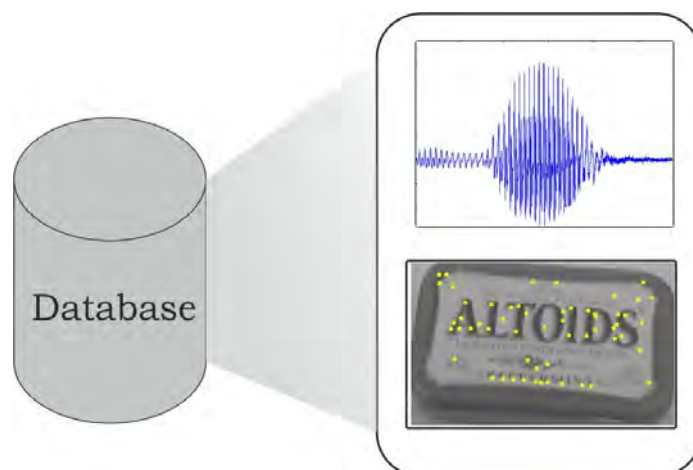


Figure 7.1: Database built-up in CLARA over time.

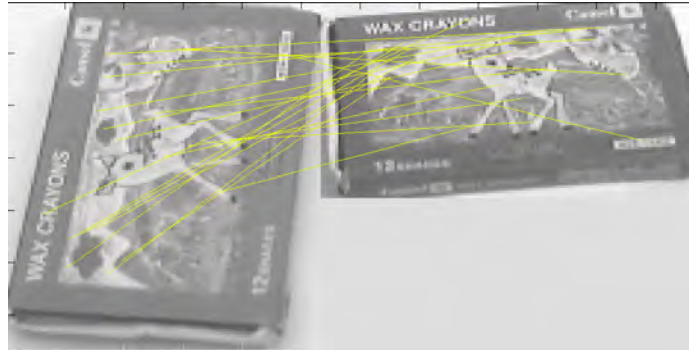


Figure 7.2: Rotational invariance provided by SIFT.

The *cognitive engine* of the system is responsible for determining several key issues such as deciding if a new object is correctly detected and added to the database, alongside other important functions. This chapter details many important functions carried out by the *cognitive engine* in making CLARA an interactive system in the true sense of the word.

How does the system store the visual information contained in the object under consideration?

The chosen feature extractor and descriptor must have the desired property of invariance to a range of factors such as scale, illumination and most importantly, rotation. Since we are using a single mounted camera, it is not possible to capture the complete information content of the object. Thus, the feature descriptor must be robust enough to account for rotation variations when it comes across the same object next. This is extremely critical since, in the real world, we are not at liberty to place the object in the exact way in which it was trained and there will be rotational offsets.

A popular extractor which caters to the above requirements is the **Scale Invariant Feature Transform (SIFT)**. This is an algorithm developed by David Lowe and is widely used in computer vision for applications such as object recognition and tagging. We have chosen the SIFT extractor since it provides the desired invariance to rotation, as shown in Fig. 7.2. A demo version of David Lowe's SIFT keypoint detector is easily accessible [33].

SIFT features are well localized in both the spatial and frequency domains, reducing the probability of disruption by occlusion, clutter, or noise. Also, the features are highly distinctive, which allows a single feature to be correctly matched with high probability against a large database of features, paving the

way for object and scene recognition. An important characteristic of these features is that the relative positions between them in the original scene shouldn't change from one image to another, i.e. the features should be chosen such that their relative positions are not affected by a change in factors such as scale, orientation etc.

It is to be noted that under extreme rotational variations, the matching of objects provided by SIFT diminishes, i.e. the number of successful keypoint matches reduces. In our system, after the object is detected using *morphological operations* and *shadow removal*, it is passed through a SIFT extractor which yields a 128-bit feature descriptor for each keypoint of the object. These descriptors are used to represent the visual information of the object in the database.

How does CLARA recognize new objects?

An important function of the *cognitive engine* is to determine if an object that it encounters is already known to it. It does this by comparing the keypoint descriptors of the object under consideration with those that are already present in the database, as illustrated in Fig. 7.3. The object on the left of each image is the object in question, while the object on the right belongs to the database.

In the first case, we can see a lot of keypoint matches between the objects, and hence, we say that a successful match is obtained and the object is already present

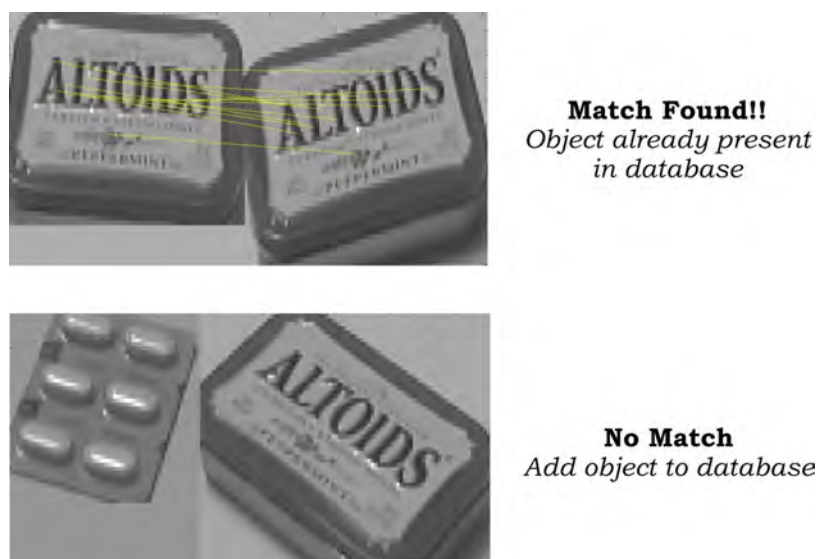


Figure 7.3: Scenario involving an object already present and an unknown object.

in the database. In the second case, there are no keypoint matches since they are two different objects and this implies that the database has to be updated with the new object.

In many situations, even when two different objects are considered, we obtain a low number of keypoint matches due to the similarities of those keypoints. We use a *threshold* for keypoint matches to ensure that we do not falsely decode it as a successful match.

Algorithm

```
flag = LOW;

for NumObjects:
    if Keypoint Matches > KeypointThresh
        flag = HIGH;
        break;
    end
end

if flag == HIGH
    print OBJECT IS IN DATABASE
else
    print ADD OBJECT TO DATABASE
end
```

where *NumObjects* denotes the size of the database, *Keypoint Matches* denotes the number of successful keypoint matches and *KeypointThresh* is a threshold chosen to avoid false matching of objects. The variable *flag* is used to check if the successful keypoint matches are above the threshold or not.

Is the system infallible? Does it never make mistakes?

Most definitely not!! It is practically not possible and also, not desirable to be always right. Since we are trying to model real-world circumstances, the system

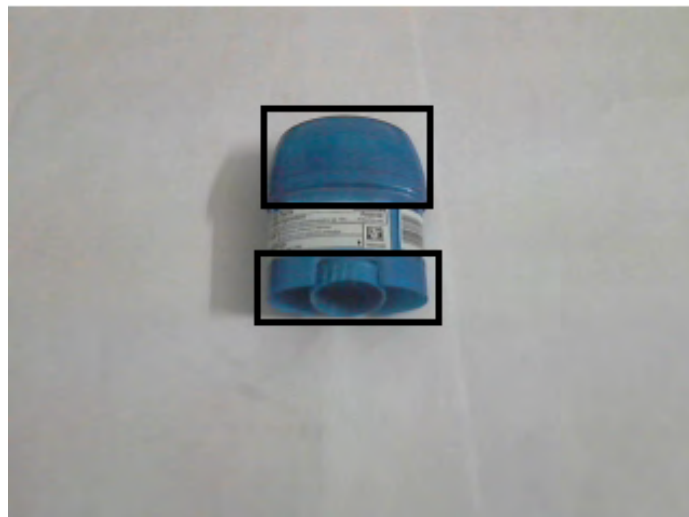


Figure 7.4: Incorrect detection of objects.

is not expected to always be correct in whatever it does. It occasionally makes mistakes, but ensures that it learns from them and this reduces the possibility of them happening again.

One instance of CLARA committing a mistake is shown in Fig. 7.4, which consists of a frame containing a single object in its field of view. On most occasions, the system correctly detects the object and obtains the bounding box around it. However, on a very few occasions, the object is incorrectly segmented into two parts owing to extreme illumination variations. If this is not corrected, CLARA updates its database with two new objects, and this has serious ramifications when it encounters the same object the next time.

We have tried to counter this by increasing the interaction between CLARA and the user, i.e. after the system detects the object, it displays the detected object to the user in order to confirm if the bounding box is correctly placed. On the few occasions that the system is wrong, the user says so and the system can correct itself, thereby ensuring that the database is not updated with false objects.

Is CLARA a walking dictionary? Does it need to have a large collection of words?

We **never** envisioned CLARA as a system with total mastery over the English language, trained with a huge collection of words. Our aim, from the beginning, was

to make it sufficiently robust to carry out rudimentary actions which are expected in practical situations such as picking up an object or pointing to it. Thus, CLARA does **not** need to know many words but just the words that are required for it to perform the desired action. In fact, having a large dictionary increases the complexity involved in deciphering the user's command, and might even lead to incorrect speech recognition.

One way of simplifying things that has been implemented is that, instead of comparing the words spoken by the user with the entire database, we introduce segmented databases which are situation-specific. We make use of a *situationally reduced dictionary* wherein the circumstances dictate the database to be used for comparison. For instance, at times, we are well aware that CLARA would ask a question which demands a **yes** or a **no** from the user. In such a situation, it would make sense to compare the user's speech to only those words rather than the entire dictionary. An argument could be made that this method makes the system rigid, but processing of the user's speech is definitely made easier.

There are basically three databases which we used in our speech recognition algorithms as given below:

i. *Action Database*

This database contains words indicating possible actions or similar words including **pick**, **show**, **this** and **nothing**. It is used for comparison when CLARA observes multiple objects and asks the user for his or her desired action. This part of the database has to be trained offline depending on the actions needed by the particular user.

ii. *Object Database*

The object database is initially empty and gets filled up dynamically by CLARA. The words in this database are chosen by the user and closely represent the object. These may be **ball**, **box**, **gum** etc. and voice samples of these words are added to the database by the user whenever they are encountered. There is no offline training for this database.

iii. *Cognitive Database*

This is the heart of the *cognitive engine* and contains the two pivotal words: **yes** and **no**. This database is also trained with the user's samples before the actual testing begins. Whenever CLARA asks a question, the user's answer is understood by comparing his words with the cognitive database.

When the user speaks to CLARA regarding the action to be performed, how does it know which object is being spoken of?

The user's words, signifying the desired action, triggers a chain of processes which ultimately results in CLARA performing it. All three systems: **ocular system**, **auditory system** and **cognitive engine** are involved in it before an instruction is issued to the robotic arm. The various steps involved in the flow of control are explained below:

- The ocular system detects the objects present in the scene and tags each object with a unique identifier or label, using the appropriate morphological operation.
- The auditory system segments the user's speech and obtains the relevant words. To simplify matters, we have chosen the first word to denote the action and the last word to represent the object, which is usually the case even in practical situations.
- The first word is compared with the *Action Database* to obtain the desired action. If the recognized word is **nothing**, CLARA does just that: nothing.
- If the first word is a valid action such as **show**, then the last word is compared with the *Object Database* to identify the object. Thus, the index of the desired object in the database is obtained.
- Using the *same* index, the visual features of the object are retrieved from the visual database linked with the auditory database.
- The ocular system matches the retrieved features with the objects present in the scene, to see if the object specified by the user is present. If it is indeed present in the image, the system identifies its location in the scene using its label generated in *step 1* of this chain.
- The location information is then sent to the robotic arm to perform the required action on the object at that position.

How is the interactive nature of the system brought about?

It is extremely important to make the system interactive since it has to know when it is making a mistake and can correct itself. Interaction between CLARA and the user has been introduced in the form of speech rather than text, since it is more user-friendly. However, there is a small probability of incorrect speech recognition which would lead to the system performing an action other than the desired one.

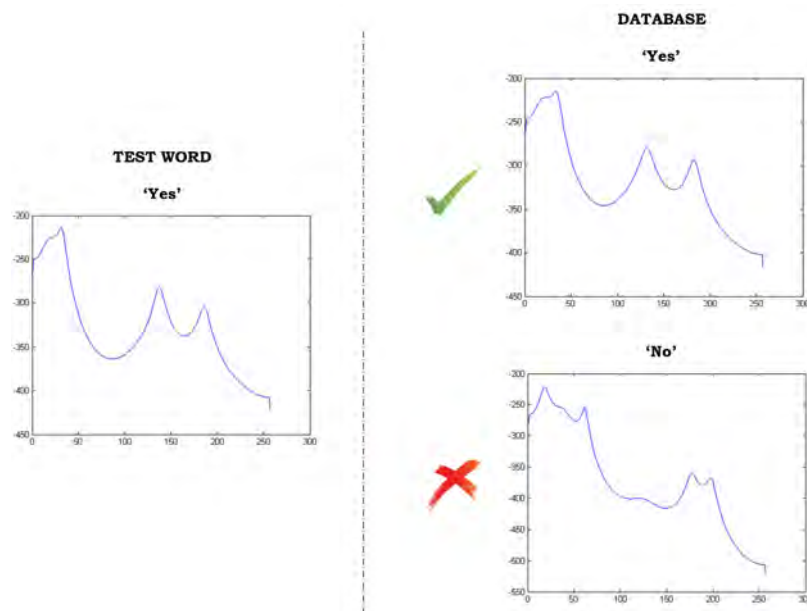


Figure 7.5: Successful recognition of the word ‘yes’.

This places a greater emphasis on maintaining good interaction between the user and CLARA.

At the heart of the system’s interactive nature lies the two words critical to cognition : *yes* and *no*. Whenever the system detects a new object, it immediately interacts with the user and asks him if it has correctly detected it. The user, with a simple *yes* or a *no* can aid the system and prevent errors from creeping in. Also, whenever the user issues the desired command out loud, CLARA immediately asks him or her if it has recognized the word successfully and it is indeed the intended action. *Thus, successful recognition for the other words is not required always but for the words **yes** and **no**, recognition has to be always right, since these form a major part of CLARA’s interaction.*

One instance of successful recognition of the word *yes* is illustrated in Fig. 7.5. The image on the left represents the features of the word *yes* spoken by the user in response to a question from CLARA. The images on the right represent the words already stored in the database. The *cognitive database* of words *yes* and *no* has to be made robust so that they can be successfully recognized almost always.

Chapter 8

Robotic Arm

8.1 Insight

The human arm has evolved over the due course of millions of years to become a very dextrous tool for handling a multitude of tasks of varying complexity. Robots have been designed to mimic the natural processes for carrying out specific tasks [34, 35, 36]. However, the gap lies in building robots that can handle various tasks by adapting themselves. One such robotic device is a robotic arm designed to mimic the human arm functions [37, 38]. The human arm has **seven** Degrees Of Freedom (DOF) and **twenty seven** DOFs for a human hand, as illustrated in Fig. 8.1 and explained in Refs. [39, 40]. For places where we need robotic arm functionality like automobile factories, hospitals or space missions, these many DOFs are usually not required [41]. So an arm with four (Refer Fig. 8.2) or five DOFs is designed, with the hand replaced by the desired type of gripper if holding or picking functionality is desired.

8.2 Need for CLARA

The present robotic arms need to be pre-programmed heavily to perform a specific task. Also, if the task being performed is slightly offset from the ideal positions the arm fails to perform the task. However, we humans are very good at such dynamic adjustments of tasks. We have tried



Figure 8.1: Human Arm.

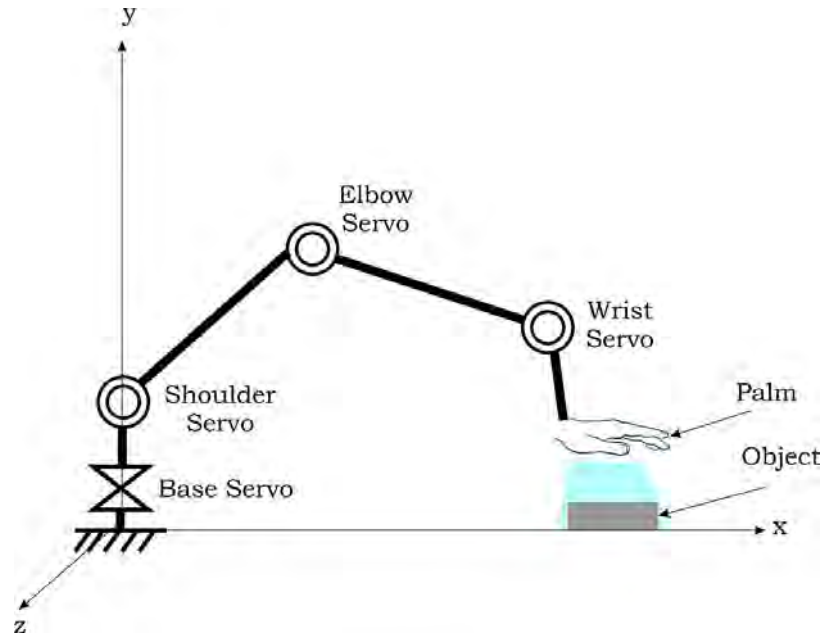


Figure 8.2: A 4 DOF Robotic Arm.

to incorporate this in CLARA. CLARA joins the fine line between completely pre-programmed robots and highly dynamic ability of the human mind. CLARA tries to learn and locate the object just like a human baby. This means that the automobile plant arms will have less training and can dynamically adapt themselves to slight variations.

8.3 Specifications

The specifications of the robotic arm are heavily dependent on the environment and application for which it is used. A higher load-bearing requirement demands servos of higher torque. Table 8.1 contains the specifications of CLARA.

Table 8.1: Robotic Arm Specifications

| | |
|---------------------|---------|
| DOFs | 4 |
| No. of Servos | 5 |
| Servo Specification | 5 Kg-cm |

8.4 Mechanical Design

CLARA is a 4 DOF robotic arm modeled after the basic joints of a human arm. The various rotational movements possible are illustrated in Fig. 8.3, depicted in a color-coded fashion. A 3mm thick transparent acrylic sheet was chosen as the material owing to its low cost, high strength and low weight. These advantages were evident when evaluated against PVC pipes, wood and aluminium. To enhance the structural integrity of the arm, steel spacers were used as beams. The servos were placed on opposite sides to ensure that the center of gravity lies on the vertical axis of the arm. The arm was designed in CorelDRAW X6 and the various parts of the arm are shown in Fig. 8.4.

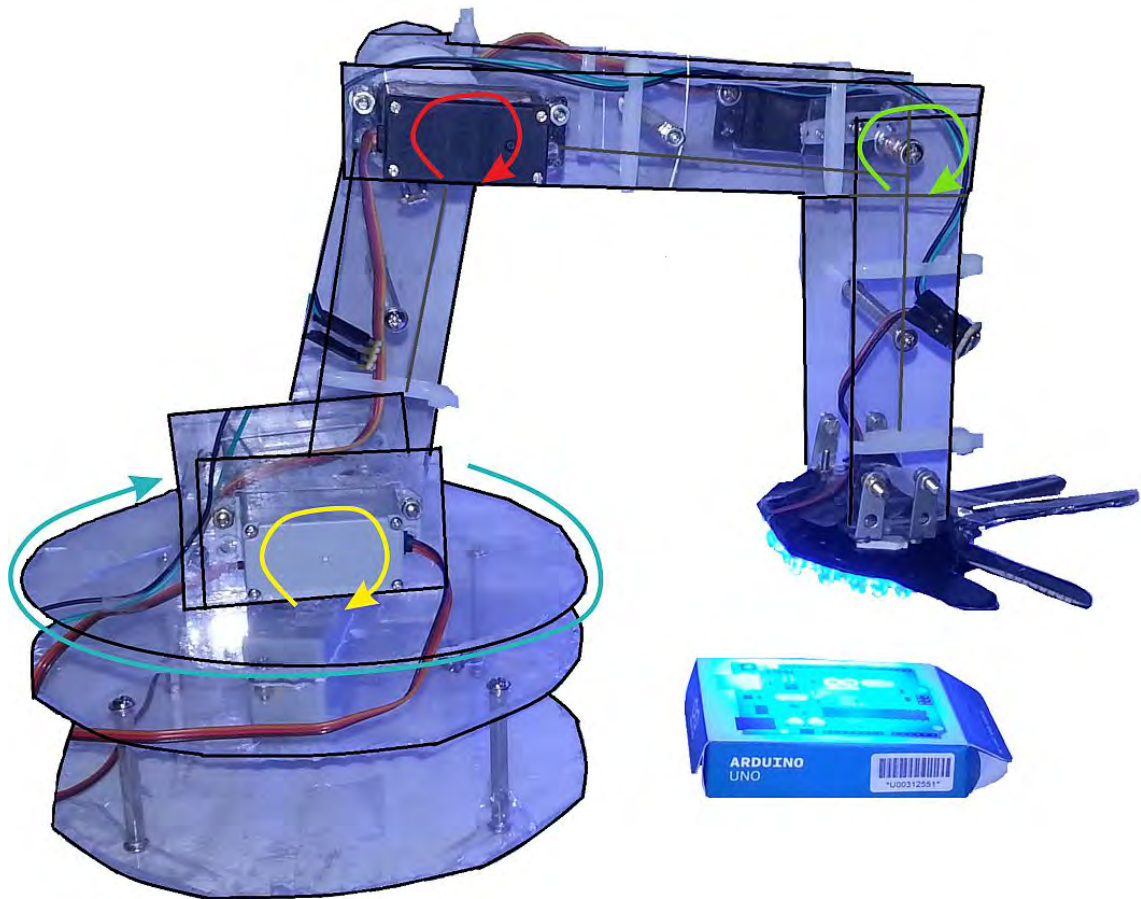
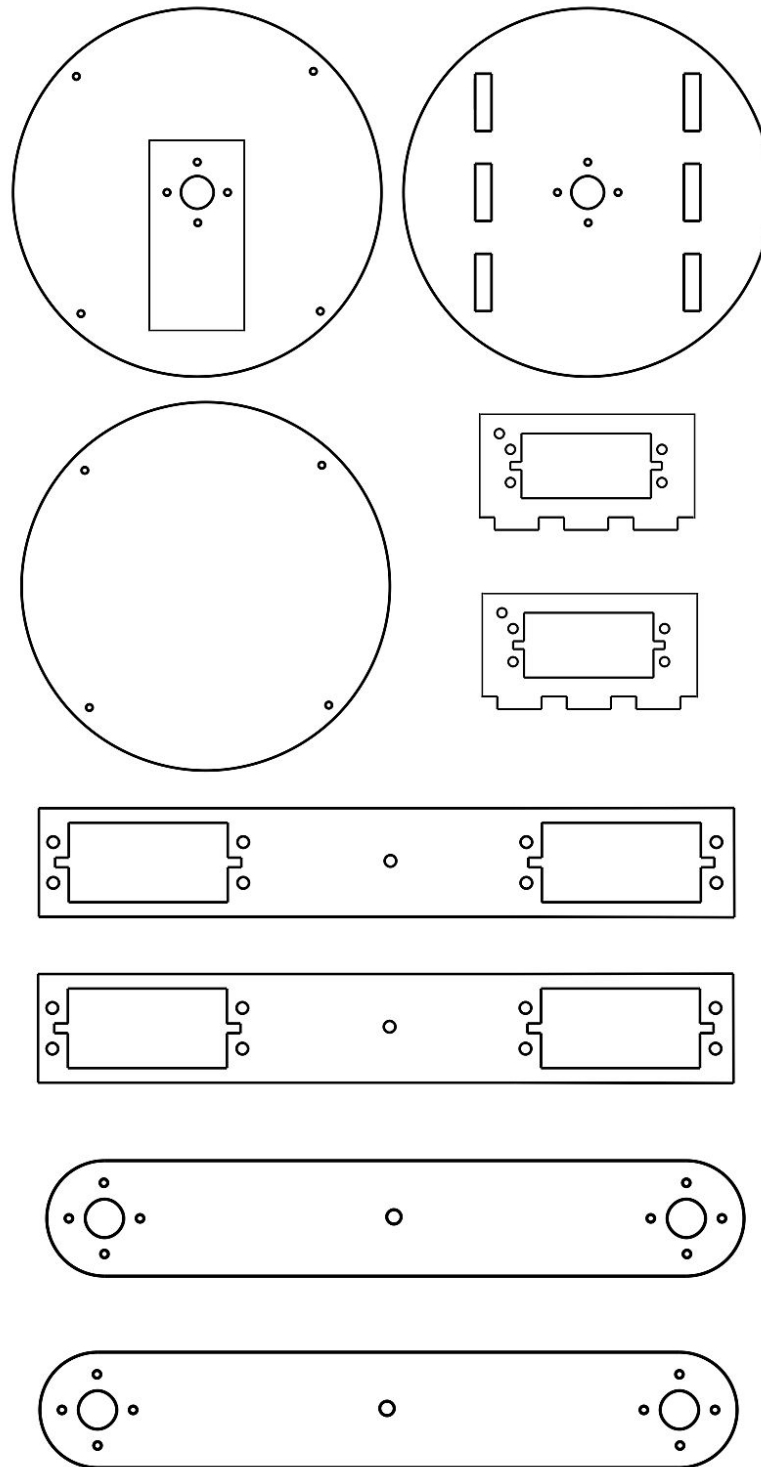


Figure 8.3: Implemented 4 DOF Robotic Arm showing each DOF in a different color; Cyan - Base, Yellow - Shoulder, Red - Elbow, Green - Wrist.



*Sketches not to scale

Figure 8.4: Mechanical designs of the arm.

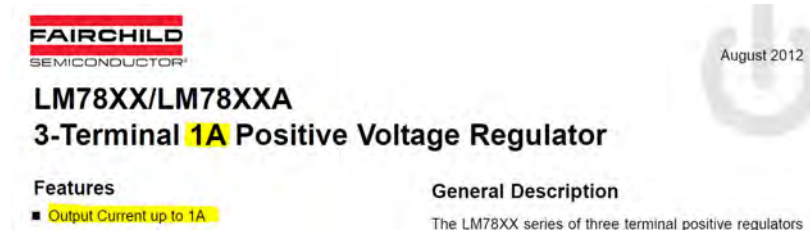


Figure 8.5: Snapshot of LM7805's datasheet.

8.5 Electrical Design

The robotic arm basically consists of five servos which can work in a voltage range of 4.8 to 6V. We have chosen an operating voltage of 5V for all servos. The servos draw a minimum current of about 150mA at no load and can go up to 700mA at full load. At stall, the current drawn can go up to 900mA. In order to cater to the worst case scenario of stall conditions, we designed a 5V power supply capable of supplying up to 900mA. Looking into the LM7805 datasheet [42], which is a voltage regulator, shown in Fig. 8.5, we can see that it can supply up to 1A of current. So we decided to use an LM7805 regulator for each servo individually. Thus, we designed a power board consisting of six regulators, five for the servos and one as an extra power supply (if needed). The regulators heat up a lot when drawing such large amounts of current, and this necessitated the use of a heat sink coupled with a cooling fan to ensure that the regulators do not overheat and catch fire. Being in a TO-220 package, the 7805 is quite compact and hence, we soldered all the regulators and their respective connections onto a perfboard. The regulators regulated 12V from the lead acid battery down to 5V and fed power to the servos, which were controlled using the Arduino PWM pins as indicated in Fig. 8.6. The detailed connections between the power board, Arduino, LCD and the servos are illustrated in Fig. 8.6, which was created using Fritzing, an open-source platform [43]. The implemented power board along with the cooling fan is shown in Fig. 8.7.

To design the front panel, we used a HD44780 blue LCD, a switch for the cooling fan and an Emergency stop (E-stop). The LCD received data from the Arduino pins and displayed the current status of CLARA to aid debugging. As any autonomous system can go wrong anytime, it is essential to have an E-stop. Hence, an E-stop was used by us to cut off the main power to the power board and in turn, the arm. The implemented front panel is shown in Fig. 8.8.



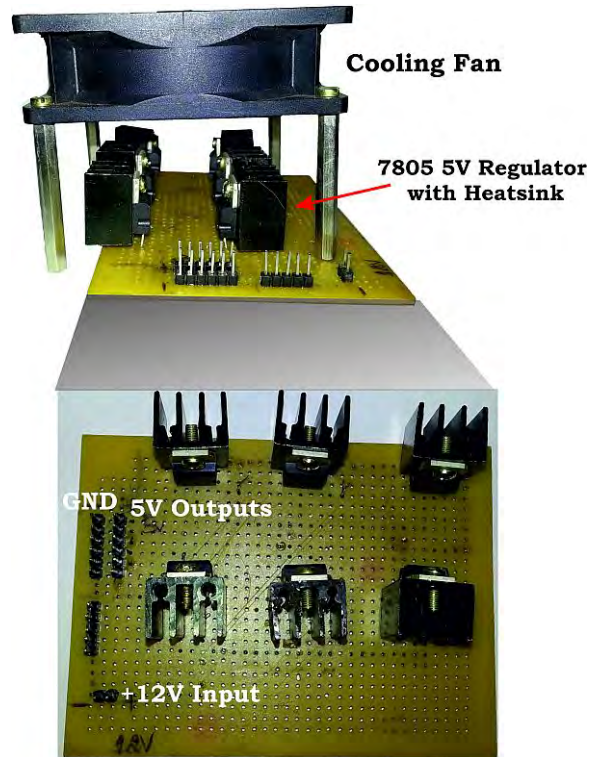


Figure 8.7: Power Supply Board used to power CLARA.

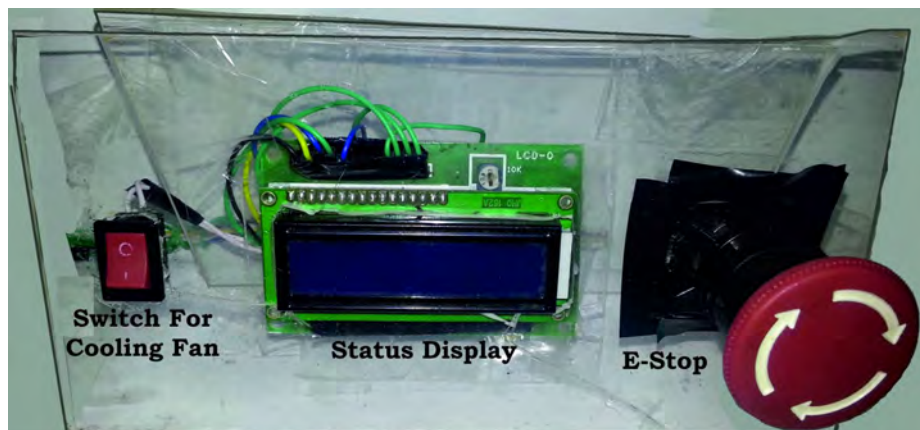


Figure 8.8: Front Panel of CLARA.

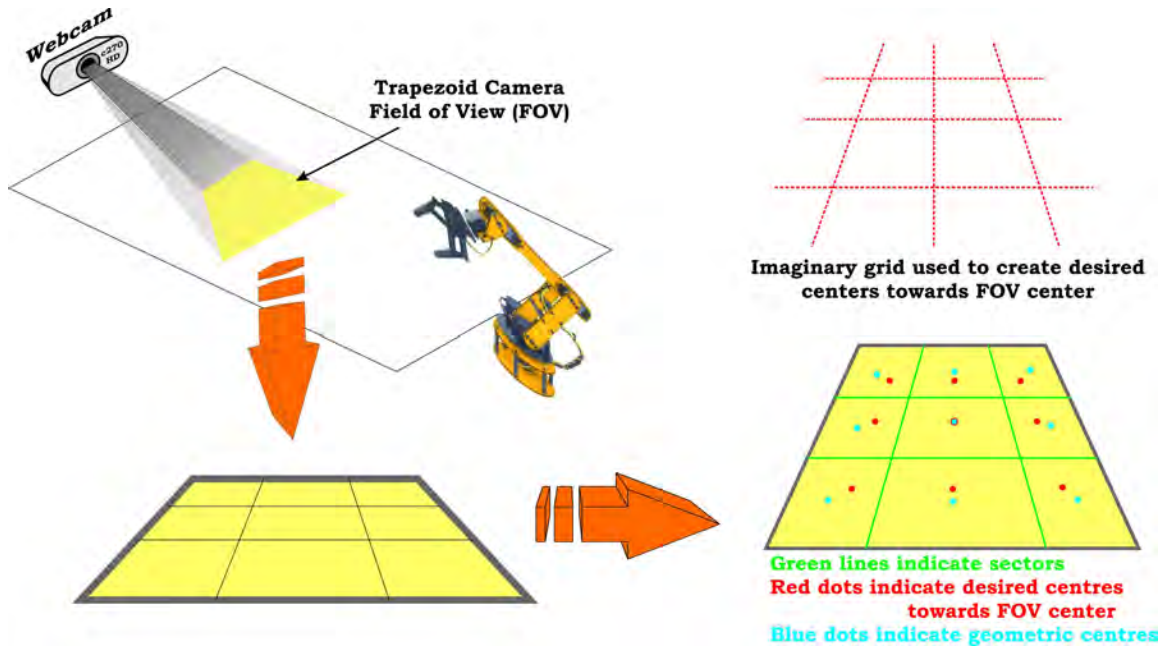


Figure 8.9: Camera field-of-view (FOV) mapped onto a trapezoid.

8.6 Integration with ocular system

The scene viewed by the camera, despite appearing to be rectangular in nature, actually gets mapped onto a trapezoidal field-of-view (FOV) in reality. It is extremely difficult and also impractical to expect the arm to reach every point in the FOV. Hence, we divide the trapezoidal region into discrete regions as shown in Fig. 8.9. We have chosen **nine** regions depending on the object size.

Ideally, the arm has to be manoeuvred onto the exact centers of the regions, as indicated by the blue dot in each region in Fig. 8.9. However, in the regions on the edges of the FOV, the object is more likely to be placed inwards in order to avoid cutting across the edges of the FOV. Thus, the arm is programmed to move to re-aligned centers of the regions, indicated by the red dots. This increases the probability of the arm reaching the exact position of the object wherever it is placed in the FOV. By dividing the FOV into a higher number of regions, we can reach more areas of the scene albeit at an increased cost of complexity.

To aid the integration of the ocular system with the arm, we have used the serial UART interface available on the Arduino platform. The interface between the arm and the ocular system via the Arduino platform is shown in Fig. 8.10. All the processing done by the ocular system of CLARA, using MATLAB, will be

8.6 Integration with ocular system

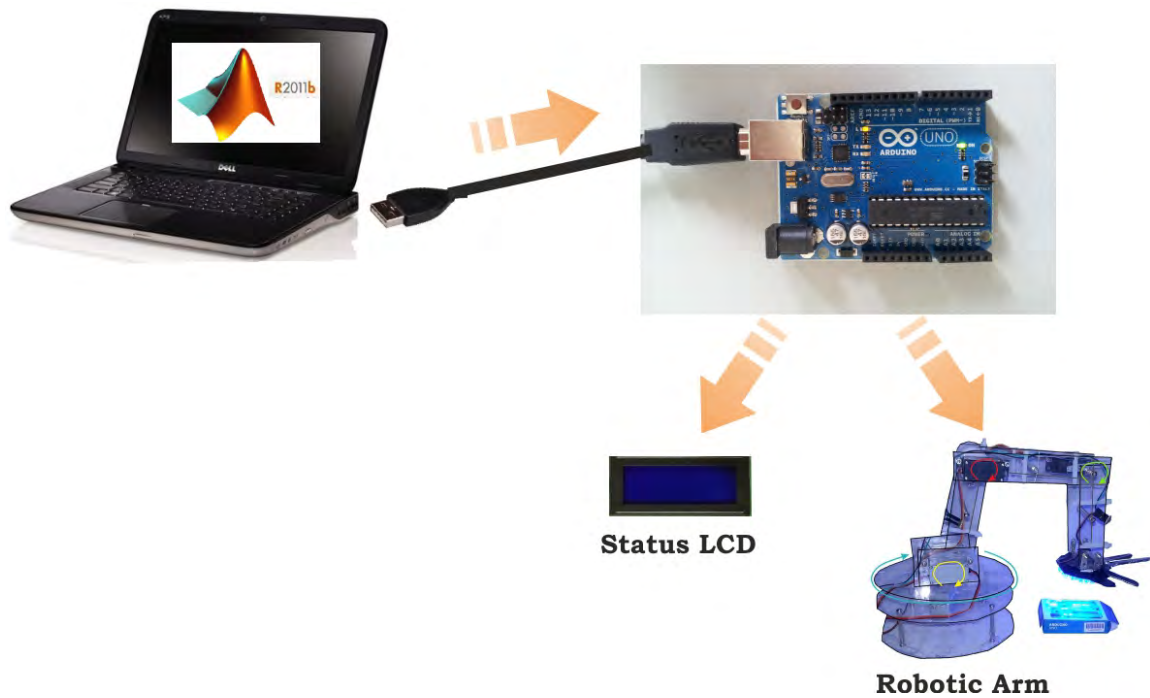


Figure 8.10: Interface of robotic arm with the ocular system.

transformed to a series of commands to be sent to the Arduino board. The series of commands consists of ASCII bytes of numbers from '1' to '9' corresponding to the respective nine areas. The arm moves to the respective region depending upon the character received.

Chapter 9

Alternate Designs

The previous chapters explain the algorithms involved in the visual, auditory and cognitive systems of CLARA. However, before zeroing in on these algorithms, we have tried a lot of other methods and discarded them due to various reasons. A clear explanation is given in this chapter regarding those techniques which we initially tried to incorporate in CLARA.

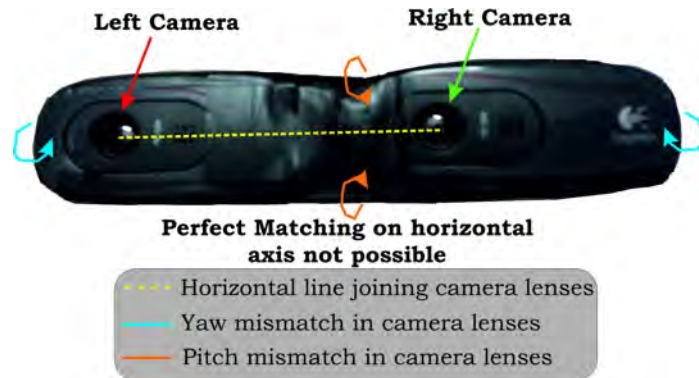


Figure 9.1: Stereoscopic camera design using two webcams.

9.1 Stereoscopic Vision

Stereoscopic Vision system is based on the human visual system [44, 45, 46]. Here, two cameras, perfectly matched and horizontally aligned, are used to create a depth map of the field of view [47, 48]. The depth map is calculated using the horizontal disparity between the left and right images. A stereo vision system was created using two similar webcams as shown in Fig. 9.1. The problem was that, the

cameras, despite being similar, were not exactly matched. Also, the cameras were not perfectly aligned along the horizontal axis since they would be misaligned with respect to pitch and yaw. This led to incorrect disparity maps and hence, erroneous depth calculations. Hence, a single camera system with no depth feedback was used.

9.2 Arm Designs

9.2.1 Conventional Arm

The initial idea was to design a human-like robotic arm. The hand was designed with fingers made of 1.5cm diameter watering pipe. Slots were cut in the pipe to make the flexing of fingers possible as shown in Fig. 9.2. A thread connecting the servo with the tip of the pipe was used to enable the folding and flexing operations. Our manual design of the arm is shown in Fig. 9.3. The hand was heavy and weighed around 200gms, which was a lot. This would mean that we would require other servos to be of rating above 10KG-cm, which are not readily available in India and would also cost more than Rs. 6000 per servo. Hence we looked for alternative designs which would be suitable for our application.

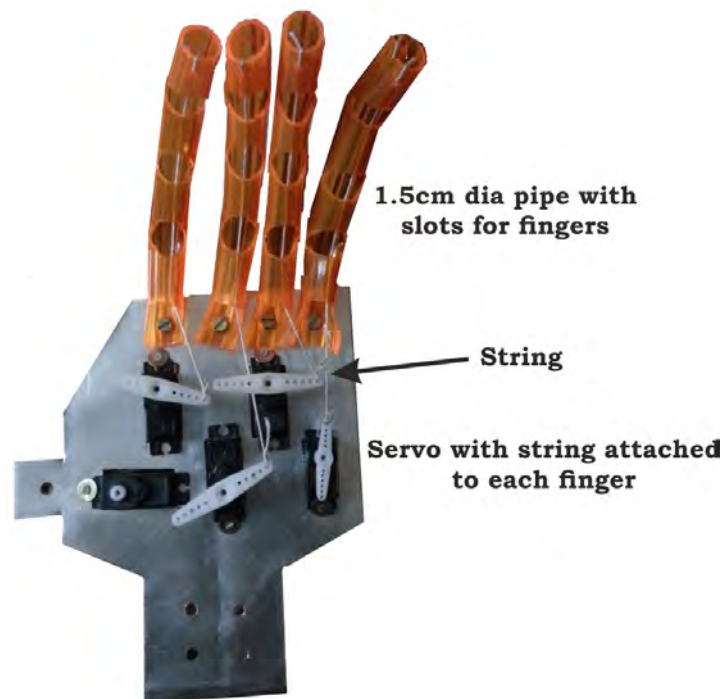


Figure 9.2: Conventional design for fingers of a robotic arm.

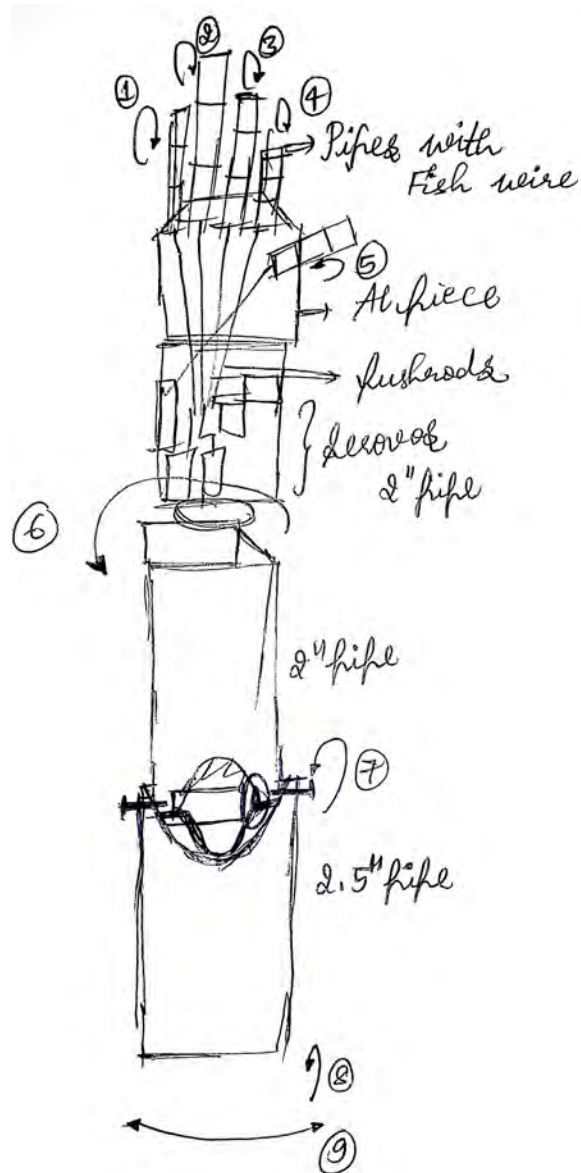


Figure 9.3: Design of a conventional robotic arm.

9.2.2 Universal Robotic Gripper

An alternative design involved the use of a universal robotic gripper based on the jamming of granular material [49], which is shown in Fig. 9.4a. Here, coffee powder is packed in a balloon bladder and a partial vacuum is created in it using a pump [50]. This makes the balloon packed with coffee powder conform to the shape on which the gripper was impressed upon. A rolling pump, shown in Fig. 9.4b, was used

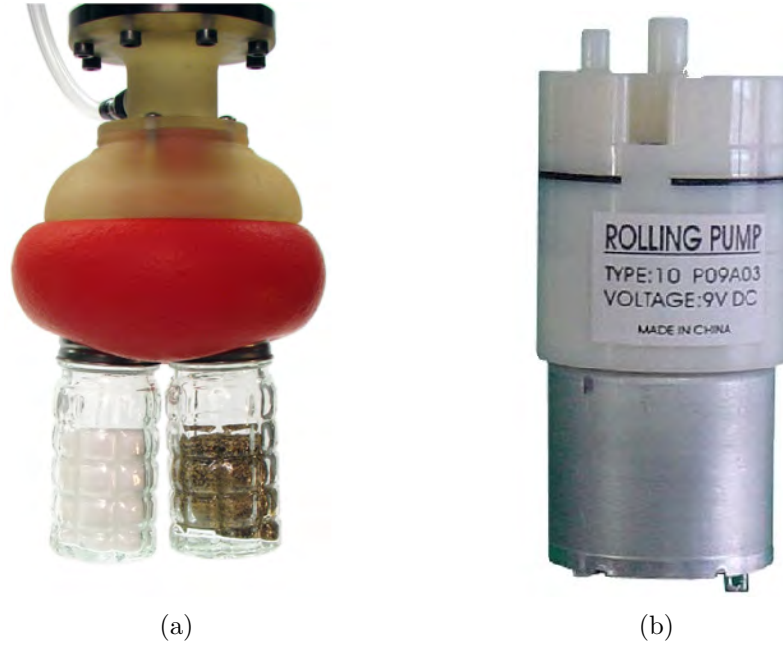


Figure 9.4: (a) Universal Robotic Gripper (b) Rolling pump used to create vacuum.

to create vacuum but it did not provide enough suction for the gripper to work. A home vacuum pump partially worked but was eliminated owing to its heavy weight and AC power requirement. After extensive study and trials, we chose to design an arm which could show objects but not pick them up. This was made necessary by the fact that the action of picking demanded higher servo ratings in order to prevent overloading of the arm.

Chapter 10

Components


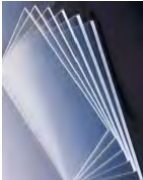



10.1 Arduino UNO

The Arduino Uno is a microcontroller board based on the ATmega328 [51]. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button [52].

10.2 List of Components and Tools

A comprehensive list of the components and tools used by us is given in Tables 10.1 and 10.2 respectively.

Table 10.1: List of Components used with their price and place of availability. Refer Appendix H for details.

| Sl. No. | Component | Qty. | Used For | Cost/Piece | Total Cost | Source |
|--|----------------|-------|--|------------|------------|--------|
|  1 | Arduino UNO R3 | 1 | Arm Control | 1300 | 1300 | A |
|  2 | Acrylic Sheet | 2'X3' | Arm Chassis Material | 65/foot | 390 | B |
|  3 | Perfboard | 2 | Hand, Motor Driver and Servo Extenders | 15 | 30 | C |
|  4 | Berg Wires | 30 | Connections of LCD and Servos | 2 | 60 | A |
|  5 | E-Stop | 1 | Kill switch for Arm | 100 | 100 | C |
|  6 | Switch | 1 | Switch for Cooling Fan | 10 | 10 | C |

10.2 List of Components and Tools

| Sl. No. | Component | Qty. | Used For | Cost/Piece | Total Cost | Source |
|---------|--|------|--------------------------|------------|------------|--------|
| 7 |  Logitech C270 Webcam | 1 | Video Input Device | 1155 | 1155 | D |
| 8 |  Headphones With Mic | 1 | Voice Input Device | 510 | 510 | D |
| 9 |  5KG Servos | 3 | Base and Shoulder Servos | 440 | 1320 | A |
| 10 |  Avionic AV46A Servos | 2 | Elbow and Wrist Servos | 550 | 1100 | E |
| 11 |  7805 5V Voltage Regulators | 6 | Power Board | 7 | 42 | C |
| 12 |  Heatsink | 6 | Heatsink for 7805 | 5 | 30 | C |

10.2 List of Components and Tools








| Sl. No. | Component | Qty. | Used For | Cost/Piece | Total Cost | Source |
|---------|--|---------|--------------------------|------------|------------|--------|
| 13 |  Cooling Fan | 1 | Cooling Power Board | 30 | 30 | F |
| 14 |  5mm Blue LEDs | 40 | Hand | 0.75 | 30 | C |
| 15 |  6cm Spacer | 2 | Arm Strusses | 10 | 20 | G |
| 16 |  5cm Spacer | 1 | Arm Struss | 10 | 10 | G |
| 17 |  4cm Spacer | 4 | Base Spacer | 10 | 40 | G |
| 18 |  Nuts and Bolts | 50 Each | For all screw joints | 1 | 100 | G |
| 19 |  7.2AH Lead Acid Battery | 1 | Powering the Robotic Arm | 700 | 700 | C |

Table 10.2: List of Tools used with their place of availability. Refer Appendix H for details.

| Sl. No. | Tool | Used For | Source |
|--|--|-----------------------------|--------------------|
|  1 | Bosch GSB 10 RE Professional Drill Kit | Drilling Holes | Bosch |
|  2 | Needle File | Filing away excess acrylic | G |
|  3 | Acrylic Cutter | Cutting Acrylic Sheets | B |
|  4 | Screwdrivers | For tightening screws | Any Hardware Store |
|  5 | Soldering Iron and Solder | Soldering Connections | C |
|  6 | Pliers and Wire Stripper | Cutting and Stripping wires | Any Hardware Store |

Chapter 11

Potential Applications

11.1 Industrial Applications

Robotic arms are used extensively in industries where human-like dexterous jobs are performed, such as automobile factories. The Ferrari factory uses two robotic arms called *Romeo* and *Juliet* (Refer Fig. 11.1) to place split rings on the pistons [53]. This is a very precise job and needs to be done perfectly every single time, failing which the engine may fail or seize. To ensure this, Ferrari spends a lot on the routine maintenance of these robotic arms. Also, the factory will stop working even if small tolerances are exceeded. CLARA could be deployed in such a situation and, communicating with the other arm, would ensure that the system handles small faults by itself. Also, this would decrease the routine maintenance costs incurred by the company.

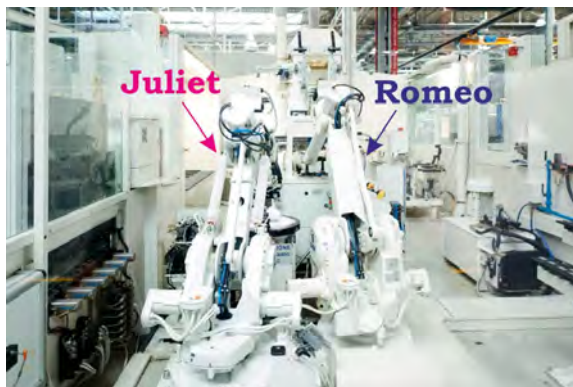


Figure 11.1: Industrial robotic arms *Romeo* and *Juliet*.



Figure 11.2: Miners working in hazardous environments.

11.2 Cognitive Platform

CLARA uses open source hardware called Arduino for its implementation. Hence, it could be used as an open-source hardware platform to educate people about cognitive learning. Owing to its low cost of implementation, anyone can implement CLARA at home and deploy more complex algorithms on it. Since CLARA uses the processor of a desktop computer or a laptop for all its computations, really complex algorithms can be developed. It can also be used by universities which perform cognition-related research without incurring high costs.

11.3 Hazardous Environments

It can be seen from Fig. 11.2 that it is highly difficult for humans to work in mines. It is extremely dangerous for humans to work in coal mines since the roof can collapse anytime and also due to low oxygen levels, extreme stress and poisonous gases. CLARA could be deployed enabled with path planning to perform these hazardous tasks. Other hazardous environments where CLARA could be deployed are nuclear reactors, battlefields and chemical refineries.

11.4 Robotic Surgery

The human body is very complex to understand; even doctors learn from every single operation that they perform. Hence, CLARA could be deployed in robotic surgery like the one shown in Fig. 11.3, to learn from every single procedure



Figure 11.3: Robotic surgery.

and continually improve itself [54, 55]. This could be a breakthrough for complex techniques which are performed by very few doctors in the world and hence, could save many lives.

11.5 Space Exploration

Mars rovers like *Curiosity*, shown in Fig. 11.4, are enabled with robotic arms for picking up and analyzing samples [56]. These robotic arms can be replaced by CLARA to enhance the functionality of the arm and to aid the rover to pick up qualitatively better samples. It could also be trained to perform different actions depending on the objects it encounters.



Figure 11.4: Robotic arm of Mars rover *Curiosity*.



Figure 11.5: Luke Arm - An arm for amputees.

11.6 Arm for an amputee

The most ambitious application would be to give an amputee a robotic arm to aid him in doing everyday chores and bring a semblance of normalcy to his life. One such arm, called the Luke Arm [57], was developed by Dean Kamen and is shown in Fig. 11.5. The Luke Arm is controlled by the human mind via implants. The amputee must undergo rigorous training to be able to use the arm. This could employ the CLARA engine to facilitate easier control. The cognitive nature of CLARA would help in aligning the arm to the amputee's needs.

Chapter 12

Future Work

12.1 Wider range of objects

The algorithm implemented in CLARA uses the **saturation** component for all its ocular processing. While it has its advantages of clearly separating the object from its background, its drawback is that the color content of the object is critical. Thus, the objects shown in Fig. 12.1, which have very little color content and are unsaturated, are not suitable with this algorithm since they cannot be differentiated from the background. An alternate technique could be thought of to incorporate these objects as well.

12.2 Cluttered environment

We have deployed a white background since it provides a stable environment for object detection and recognition. However, the real world contains far more clutter and varied backgrounds than we have used. While the plain background is



Figure 12.1: A few objects which CLARA cannot detect.

a start in achieving successful object detection, the next step would be to deploy a good segmentation technique which would automatically detect new objects even in practical scenarios.

12.3 Increase in resolution of FOV

The FOV of the camera, which is trapezoidal in nature, has been divided into **nine** regions by taking the sizes of the objects used into account. A logical step following this would be to implement a higher number of regions, so that the resolution increases and objects could be reached anywhere in the FOV. A compromise has to be reached between the required arm resolution and the cost of programming the arm for each additional region.

12.4 Auto-calculation of servo values

In the present version of CLARA, we have **nine** regions and, for each region, the values to be given as input to each servo is converted to a Look Up Table (LUT) using the trial and error method. The input to the servo is the angle at which the servo is required to rest. This method proves to be cumbersome if the FOV is divided into a large number of regions. A viable alternative is to model the joints of the arm in the form of *kinematic relations*, and try to deduce the servo values by solving the simultaneous equations for a particular point in the FOV. This technique would introduce unmatched resolution to the arm, since it can decide the required servo values by itself to move to a particular point in the FOV.

Appendix A

MATLAB to Arduino Serial Communication

- Use **s = serial('COMxx');** command to create a serial object corresponding to Arduino [58]. *xx* is the COM port to which Arduino is connected.
- Check which COM port the Arduino is connected to, by looking in Computer Management under Ports (COM & LPT) in Device Manager (Refer Fig. A.1).
- Baud Rate can be changed using **s.BaudRate = Required Baud Rate;**.
- Use **fopen(s);** to start serial communication via the serial port.
- To send a byte to Arduino, use the command **fwrite(s,[data],'uint8');**.
- To close the serial port, use the command **fclose(s);**.

Note: MATLAB to Arduino Serial Communication does not work on FTDI based Arduino boards!!

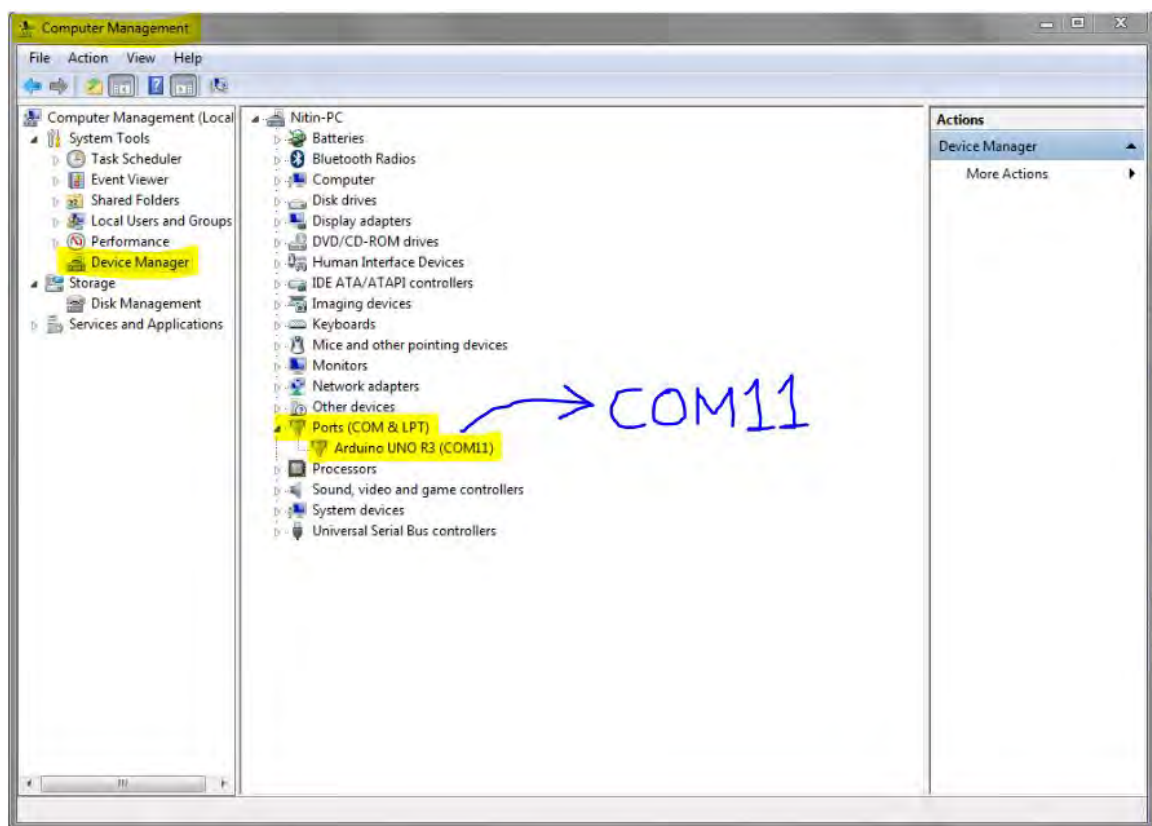


Figure A.1: Serial COM port used by Arduino.

Appendix B

Servo Torque Ratings

The torque rating required for the servos shown in Fig. B.1 can be calculated as follows:

Torque About Joint 1:

$$M1 = \frac{L1}{2} \times W1 + L1 \times W4 + \left(L1 + \frac{L2}{2} \right) \times W2 + (L1 + L3) \times W3$$

Torque about Joint 2:

$$M2 = \frac{L2}{2} \times W2 + L3 \times W3$$

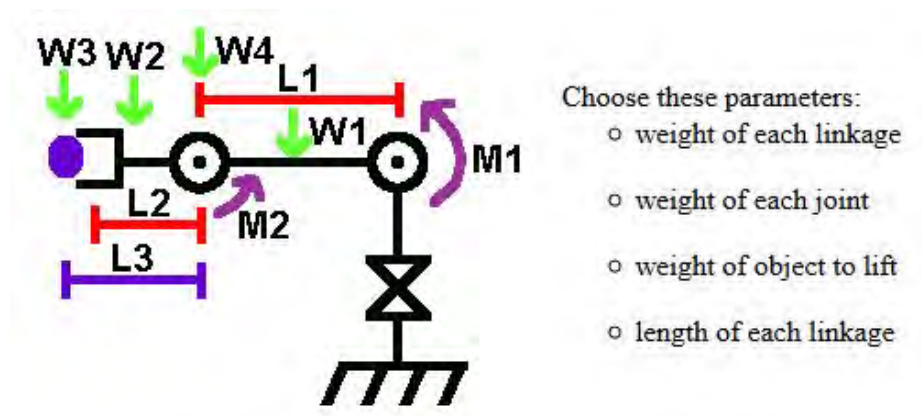


Figure B.1: Determination of servo torque ratings.

Appendix C

A Beginner's Guide to Arduino

The following are the most basic commands required for using Arduino :

pinMode(pin, mode); Configure “pin” to “mode” (INPUT or OUTPUT).

digitalWrite(pin, value); Write “value” (HIGH or LOW) on to “pin”.

analogWrite(pin, value); Write PWM wave “value” varying from 0 to 5V (0 to 255) on to “pin”.

delay(ms); Pauses your program for “ms” milliseconds.

Serial.begin(int BR); Start Serial communication at baud rate “BR”.
(Typically 9600)

int Serial.available(); - Get the number of bytes (characters) available for reading over the serial port.

int Serial.read(); - Reads incoming serial data.

Serial.print(data); - Prints “data” to the serial port.

Serial.println(data); - Prints “data” to the serial port followed by a new line character.

For more information, refer Refs. [59, 60, 61].

Appendix D

Servo Control using Arduino

- Use package `#include <Servo.h>`.
- Create a servo object by using the command `SoftwareServo label;`.
- To assign a servo to a pin, use the command `label.attach(pinNo);`.
- To write a particular value “val” to the attached servo, use the command `label.write(val);`. Here, “val” can vary from 0 to 180 for standard servos.
- Use `label.detach()` to release the pin from servo driving.

Appendix E

HD44780 LCD

| Code (Hex) | Command to LCD Instruction Register |
|---------------|---|
| 1 | Clear Display screen |
| 2 | Return home |
| 4 | Decrement cursor (Shift cursor to left) |
| 6 | Increment cursor (Shift cursor to Right) |
| 5 | Shift display right |
| 7 | Shift display left |
| 8 | Display off, cursor off |
| A | Display on, cursor off |
| C | Display on, cursor off |
| E | Display on, cursor blinking |
| F | Display on, cursor blinking |
| 10 | Shift cursor position to left |
| 14 | Shift cursor position to right |
| 18 | Shift the entire display to the left |
| 1C | Shift the entire display to the right |
| 80 | Force cursor to beginning of 1 st line |
| 0C0 | Force cursor to beginning of 2 nd line |
| 38 | 2 lines and 5x7 Matrix |

Figure E.1: Command set of HD44780 (See Ref. [62]).



Figure E.2: HD44780 LCD.

| Higher 4bit Lower 4bit | 0000 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
|---------------------------------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| xxxx0000 | | | | | | | | | | | | | |
| xxxx0001 | | | | | | | | | | | | | |
| xxxx0010 | | | | | | | | | | | | | |
| xxxx0011 | | | | | | | | | | | | | |
| xxxx0100 | | | | | | | | | | | | | |
| xxxx0101 | | | | | | | | | | | | | |
| xxxx0110 | | | | | | | | | | | | | |
| xxxx0111 | | | | | | | | | | | | | |
| xxxx1000 | | | | | | | | | | | | | |
| xxxx1001 | | | | | | | | | | | | | |
| xxxx1010 | | | | | | | | | | | | | |
| xxxx1011 | | | | | | | | | | | | | |
| xxxx1100 | | | | | | | | | | | | | |
| xxxx1101 | | | | | | | | | | | | | |
| xxxx1110 | | | | | | | | | | | | | |
| xxxx1111 | | | | | | | | | | | | | |

Figure E.3: Character set of HD44780.

Appendix F

LUT of robotic arm

As explained previously, servos take values between 0 and 180 as input, which represents the angular position of the shaft. For each of the **nine** regions of the trapezoidal field-of-view (FOV) shown in Fig. F.1, the values to be given to each servo have to be determined by trial and error, and then stored in the form of a look-up-table (LUT). Table F.1 contains the LUT used in CLARA's working. The terms *base*, *shoulder*, *elbow* and *wrist* refer to the corresponding servos used in the arm. The term *home* refers to the servo values required to maintain the arm at its reference position of minimum stress.

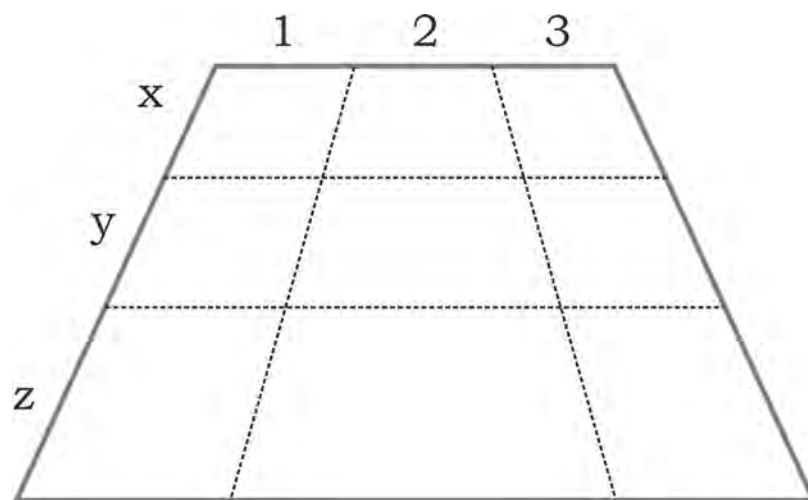


Figure F.1: Trapezoidal FOV as seen by CLARA.

Table F.1: LUT used by CLARA to navigate the trapezoidal FOV.

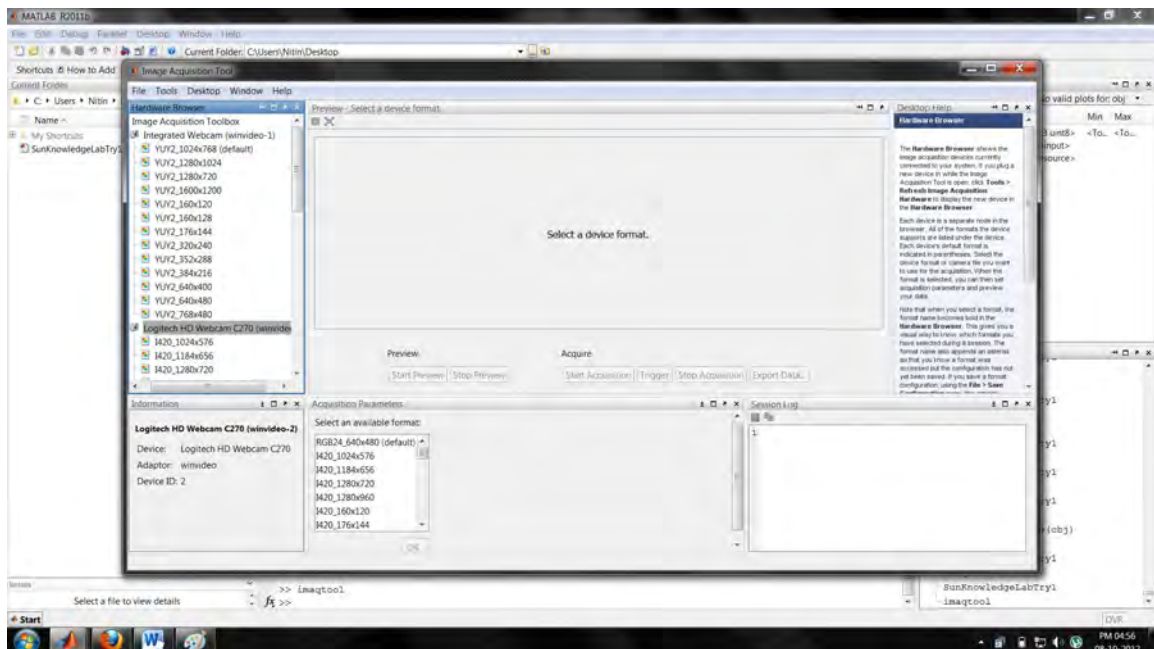
| | Base | Shoulder | Elbow | Wrist |
|------|------|----------|-------|-------|
| Home | 100 | 90 | 75 | 0 |
| x1 | 115 | 145 | 125 | 90 |
| y1 | 118 | 140 | 125 | 70 |
| z1 | 124 | 125 | 100 | 66 |
| x2 | 99 | 145 | 120 | 100 |
| y2 | 104 | 125 | 100 | 86 |
| z2 | 104 | 110 | 80 | 75 |
| x3 | 86 | 145 | 110 | 117 |
| y3 | 88 | 127 | 90 | 97 |
| z3 | 80 | 120 | 80 | 85 |

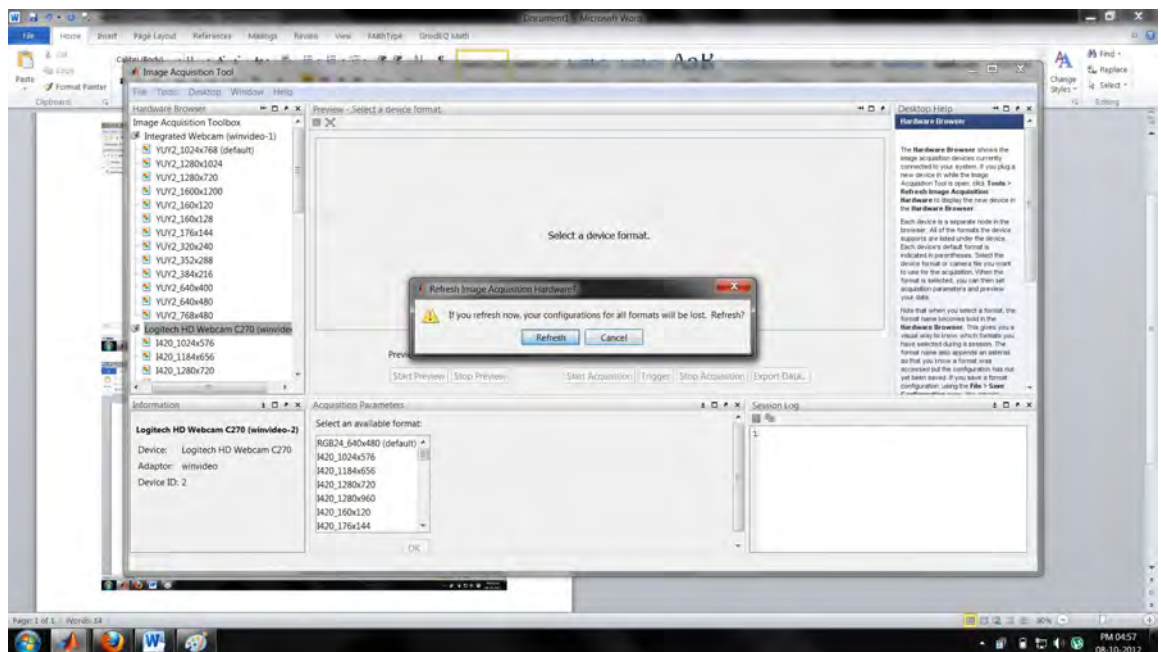
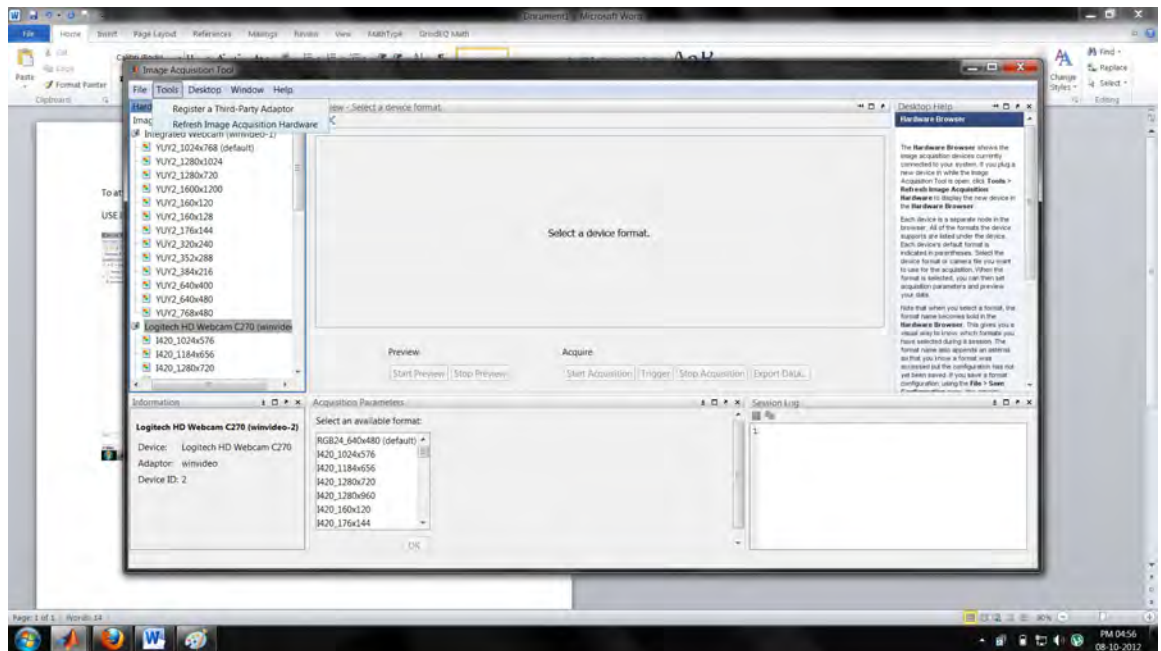
Appendix G

Configuring an external webcam with MATLAB

This is a step-by-step guide to the process involved in making an external webcam work with MATLAB.

Use `imaqtool` command [63] and refresh the list:





Appendix H

Addresses

A - **NSK Electronics**, No. 66/12, Ground Floor, Opp. Vishal Electronics, S. P. Road, Bangalore - 560002, Ph: 9740374373 (Mob.), (080) 22898194, 41575115.

B - **Mahesh Acrylic**, No. 93/1, R. T. Street, B. V. K. Iyengar Road Cross, Bangalore - 560053. Ph: 9448471601 (Mob.), (080) 22352680, 41224051.

C - **Om Electronics**, No. 110, Near Town Hall, S. P. Road, Bangalore - 560002, Ph: (080) 41127727.

D - **Flipkart**, <http://www.flipkart.com/>.

E - **RC Bazaar**, No. 1193, Opp. To Ragi Gudda Temple Arch, 26th Main Road, Jayanagar 9th Block, Bangalore - 560069, Ph: (080) 41738892.

F - **Vishal Electronics**, No. 139/7, Opp. To Karthik Plaza, S. P. Road, Bangalore - 560002, Ph: 9845072496 (Mob.), 9886717462(Mob.), (080) 41517650, 41517640, 41517630, 22233851.

G - **Durga Trading Company**, No. 4, E. V. Lane, S. P. Road Cross, Bangalore - 560002, Ph: 9448827206(Mob.), (080) 41140781, 41247783.

Bibliography

- [1] Ricardo Beira, Manuel Lopes, M Praga, José Santos-Victor, Alexandre Bernardino, Giorgio Metta, Francesco Becchi, and Roque Saltarén. Design of the robot-cub (icub) head. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 94–100. IEEE, 2006. 3.1.1
- [2] Vadim Tikhonoff, Angelo Cangelosi, Paul Fitzpatrick, Giorgio Metta, Lorenzo Natale, and Francesco Nori. An open-source simulator for cognitive robotics research: the prototype of the icub humanoid robot simulator. In *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*, pages 57–61. ACM, 2008. 3.1.1
- [3] Nikolaos G Tsagarakis, Giorgio Metta, Giulio Sandini, David Vernon, Ricardo Beira, Francesco Becchi, Ludovic Righetti, Jose Santos-Victor, Auke J Ijspeert, Maria Chiara Carrozza, et al. icub: the design and realization of an open humanoid platform for cognitive and neuroscience research. *Advanced Robotics*, 21(10):1151–1175, 2007. 3.1.1
- [4] Giorgio Metta, Giulio Sandini, David Vernon, Lorenzo Natale, and Francesco Nori. The icub humanoid robot: an open platform for research in embodied cognition. In *Proceedings of the 8th workshop on performance metrics for intelligent systems*, pages 50–56. ACM, 2008. 3.1.1
- [5] icub. <http://www.icub.org/>. 3.1.1
- [6] Nao. <http://www.aldebaran-robotics.com/en/>. 3.1.2
- [7] Alexander Kain and Michael W Macon. Design and evaluation of a voice conversion algorithm based on spectral envelope mapping and residual prediction. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 2, pages 813–816. IEEE, 2001. 1

- [8] Robert V Shannon, Fan-Gang Zeng, Vivek Kamath, John Wygonski, and Michael Ekelid. Speech recognition with primarily temporal cues. *Science*, 270 (5234):303–304, 1995. 2
- [9] Liang Gu and Kenneth Rose. Perceptual harmonic cepstral coefficients for speech recognition in noisy environment. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 1, pages 125–128. IEEE, 2001. 3
- [10] Mark Gales and Steve Young. The application of hidden markov models in speech recognition. *Foundations and Trends in Signal Processing*, 1(3):195–304, 2008. 4
- [11] Jie Tang, Stephen Miller, Arjun Singh, and Pieter Abbeel. A textured object recognition pipeline for color and depth image data. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3467–3474. IEEE, 2012. 3.3
- [12] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999. 3.3.1
- [13] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. 3.3.1
- [14] Thomas Serre, Minjoon Kouh, Charles Cadieu, Ulf Knoblich, Gabriel Kreiman, and Tomaso Poggio. A theory of object recognition: computations and circuits in the feedforward path of the ventral stream in primate visual cortex. Technical report, DTIC Document, 2005. 3.3.1
- [15] Jeffrey S Beis and David G Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 1000–1006. IEEE, 1997. 3.3.1
- [16] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3): 346–359, 2008. 3.3.1
- [17] An analysis and implementation of the surf method, and its comparison to sift. <http://www.ipol.im/pub/pre/69/>. 3.3.1
- [18] Stephen Se, David Lowe, and Jim Little. Vision-based mobile robot localization and mapping using scale-invariant features. In *Robotics and Automation, 2001*.

- Proceedings 2001 ICRA. IEEE International Conference on*, volume 2, pages 2051–2058. IEEE, 2001. 3.3.1
- [19] Koen EA van de Sande, Theo Gevers, and Cees GM Snoek. Evaluating color descriptors for object and scene recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1582–1596, 2010. 3.3.2
 - [20] Joost Van De Weijer and Cordelia Schmid. Coloring local feature extraction. In *Computer Vision–ECCV 2006*, pages 334–348. Springer, 2006. 3.3.2
 - [21] Hsv color space. http://en.wikipedia.org/wiki/HSL_and_HSV. 5.1
 - [22] Michael J Jones and James M Rehg. Statistical color models with application to skin detection. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 1. IEEE, 1999. 5.2
 - [23] Vladimir Vezhnevets, Vassili Sazonov, and Alla Andreeva. A survey on pixel-based skin color detection techniques. In *Proc. Graphicon*, volume 3, pages 85–92. Moscow, Russia, 2003. 5.2
 - [24] Juan José de Dios and Narciso García. Face detection based on a new color space ycgcr. In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, volume 3, pages III–909. IEEE, 2003. 5.2
 - [25] Mark Nixon and Alberto S Aguado. *Feature extraction & image processing*. Academic Press, 2008. 5.4
 - [26] Daniel Grest, Jan-Michael Frahm, and Reinhard Koch. A color similarity measure for robust shadow removal in real time. In *Vision, Modeling and Visualization*, pages 253–260. Citeseer, 2003. 5.5
 - [27] Cie l*a*b* color space. www.hunterlab.com/appnotes/an07_96a.pdf. 5.5
 - [28] Daryl Ning. Developing an isolated word recognition system in matlab. <http://www.mathworks.in/company/newsletters/articles/developing-an-isolated-word-recognition-system-in-matlab.html>. 6
 - [29] Sharath Kumar K. B., Kiran Ganesh Banakar, Nagaraj N. K., Vishwanath G. Naik, and Sadashiva V. Chakrasali. Technical report on speech pause detection algorithms, 2012. M. S. Ramaiah Institute of Technology. 6.1
 - [30] Vojtech Stejskal, Nikolaos Bourbakis, and Anna Esposito. Empty speech pause detection algorithms comparison. *International Journal of Advanced Intelligence*, 2(1):145–160, 2010. 6.1

- [31] Psd using yule-walker ar method. <http://www.mathworks.in/help/signal/ref/pyulear.html>. 6.3
- [32] Euclidean distance. http://en.wikipedia.org/wiki/Euclidean_distance. 6.4
- [33] Sift keypoint detector demo software download. <http://www.cs.ubc.ca/~lowe/keypoints/>. 7
- [34] A complete robotic arm tutorial, . http://www.societyofrobots.com/robot_arm_tutorial.shtml. 8.1
- [35] A completed robotic arm with grippers, . <http://jjshortcut.wordpress.com/2010/04/19/my-mini-servo-grippers-and-completed-robotic-arm/#more-247>. 8.1
- [36] Robotic arm with 7 servos, . <http://www.thingiverse.com/thing:2433>. 8.1
- [37] Open source diy robotic arm, a call for your skills, . <http://www.gizmowatch.com/entry/open-source-diy-robotic-arm-a-call-for-your-skills/>. 8.1
- [38] Open source robotic arm, . <http://www.thingiverse.com/thing:387>. 8.1
- [39] Seven dofs in the human arm, . <http://www.thetech.org/exhibits/online/robots/arms/7deg.html>. 8.1
- [40] How many degrees of freedom does a human hand have?. http://wiki.answers.com/Q/How_many_degrees_of_freedom_does_a_human_hand_have. 8.1
- [41] Uses of a robotic arm, . <http://www.asc-csa.gc.ca/eng/canadarm/robotic.asp>. 8.1
- [42] Lm7805 datasheet. <https://www.sparkfun.com/datasheets/Components/LM7805.pdf>. 8.5
- [43] Fritzing. <http://fritzing.org/>. 8.5
- [44] David Marr, Tomaso Poggio, Ellen C Hildreth, and W Eric L Grimson. *A computational theory of human stereo vision*. Springer, 1991. 9.1
- [45] Heiko Hirschmuller. Stereo vision in structured environments by consistent semi-global matching. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2386–2393. IEEE, 2006. 9.1

- [46] Heiko Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 807–814. IEEE, 2005. 9.1
- [47] Ashutosh Saxena, Jamie Schulte, and Andrew Y Ng. Depth estimation using monocular and stereo cues. *IJCAI*, 2007. 9.1
- [48] Kohtaro Sabe, Masaki Fukuchi, J-S Gutmann, Takeshi Ohashi, Kenta Kawamoto, and Takayuki Yoshigahara. Obstacle avoidance and path planning for humanoid robots using stereo vision. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 1, pages 592–597. IEEE, 2004. 9.1
- [49] Eric Brown, Nicholas Rodenberg, John Amend, Annan Mozeika, Erik Steltz, Mitchell R Zakin, Hod Lipson, and Heinrich M Jaeger. Universal robotic gripper based on the jamming of granular material. *Proceedings of the National Academy of Sciences*, 107(44):18809–18814, 2010. 9.2.2
- [50] A versatile universal robotic gripper based on the principle of vacuum. <http://www.gizmag.com/universal-robotic-gripper/16729/>. 9.2.2
- [51] Atmega328 specifications. <http://www.atmel.in/devices/ATMEGA328.aspx>. 10.1
- [52] Arduino uno board, . <http://arduino.cc/en/Main/arduinoBoardUno>. 10.1
- [53] Romeo and juliet at the ferrari factory. <http://www.youtube.com/watch?v=tpDnQoqGB0o>. 11.1
- [54] Da vinci surgical system. http://www.intuitivesurgical.com/products/davinci_surgical_system/. 11.4
- [55] William HH Chapman Iii, Robert J Albrecht, Victor B Kim, James A Young, and W Randolph Chitwood Jr. Computer-assisted laparoscopic splenectomy with the da vinci surgical robot. *Journal of Laparoendoscopic & Advanced Surgical Techniques*, 12(3):155–159, 2002. 11.4
- [56] Robotic arm of mars rover curiosity. [http://en.wikipedia.org/wiki/Curiosity_\(rover\)#Robotic_arm](http://en.wikipedia.org/wiki/Curiosity_(rover)#Robotic_arm). 11.5
- [57] Sarah Adee. Dean kamen’s “luke arm” prosthesis readies for clinical trials. *IEEE Spectrum*, 2, 2008. 11.6
- [58] Matlab. <http://www.mathworks.in/>. A

- [59] Official arduino website, . <http://www.arduino.cc/>. C
- [60] Official arduino tutorials, . <http://arduino.cc/en/Tutorial/HomePage>. C
- [61] Official arduino syntax reference, . <http://arduino.cc/en/Reference/HomePage>. C
- [62] Hd44780 datasheet. <https://www.sparkfun.com/datasheets/LCD/HD44780.pdf>. (document), E.1
- [63] Image acquisition tool in matlab. <http://www.mathworks.in/help/imaq/imaqtool.html>. G