Development of an Integrated Control System for a Humanoid Robot

By

Rahid-uz-Zaman	201018042
Miftahur Rahman	201018043
Yeasin Arafat Bhuiyan	201018053

Supervised and Approved by
Dr. Engr. Md. Alamgir Hossain
Associate Professor
ME Dept. (MIST)

Department of Mechanical Engineering
Military Institute of Science and Technology (MIST)
Mirpur Cantonment, Dhaka, Bangladesh.

December 2013

Abstract

Humanoid robot is a huge field for modern robotics research. Many renowned organizations and intellectuals are still working on it. Some outcome of their long term research are ASIMO by Honda, NAO by Aldebaran, HUBO by KAIST, WABIAN by Waseda University etc. For the implementation of an idea, it is essential to have a full engineering system design, simulation and analysis. MISTBOY, is such a dream bot consist of 19 DoF, 45 cm height and 3.5 kg weight. Mechanical Engineering department of Military Institute of Science and Technology aims to design and built a humanoid that is capable of balancing, walking, turning, standing from a prostrate position and finally can play soccer autonomously.

The first purpose, on the of making a soccer playing robot, is to develop a stable and universal humanoid platform on which can be implement various theories and algorithms such as forward kinematics, trajectory planning, dynamic walking, AI, Vision and Image recognition and navigation.

Forward kinematics is helpful for balancing the robot and trajectory planning optimizes the motion of the robot. Forward kinematics calculates the position and orientation of the joint while trajectory planning is one of the fundamental issue in robot design and analysis which relates position, velocity and acceleration for each degree of freedom of manipulator with time. Soccer playing humanoid robot requires a minimum amount of energy to pass the ball to a certain distance which can be determined by analyzing the force and torque produced by the kick. This paper is focused on forward kinematics, trajectory planning, balancing, force and torque calculation and kick analysis.

Machine vision (MV) is developed from the concept of human vision. It works on the basis of pattern and color matching like human. MV is nothing but digital image processing which involves extraction of information from an image. It is the young discipline of modern technology, widely used for industrial purposes, security purposes, medical diagnostics, weapon engineering and so on because of its significant level of accuracy and reliability. Now-a-days, in the field of robotics, MV is expanding its area of research like in ASIMO, HUBO, NAO.

Intelligent vision algorithm is one of the most reliable and effective way to develop the control system of an autonomous robot as it can extracts maximum amount of real time data from the environment. Integration of vision data with other measurement unit, can make the control system more reliable, accurate and stable. Such kind of integrated control system has been implemented on MISTBOY, ongoing humanoid robot project. The vision system of MISTBOY is based on the NI LabVIEW Vision. National Instruments introduced a graphical programming platform LabVIEW for the engineers and scientist to solve problem easily, accurate the productivity and innovate continually. Various approach has been analyzed to find out the best method for faster and accurate tracking system of a robot. In this paper, the integrated vision based control system is also presented with analyzed result through mathematical derivation and simulation.

Acknowledgement

We are thankful to Almighty Allah for his blessings for the successful completion of our thesis. Our heartiest gratitude, profound indebtedness and seep respect go to our supervisor Dr. Engr. Md. Alamgir Hossain, Associate Professor, MIST, Dhaka, Bangladesh, for his constant supervision, affectionate guidance and encouragement and motivation. His keen interest on the topic and valuable advices throughout the study was of great help in completing thesis.

We are especially grateful to the Department of Mechanical Engineering (ME) of Military Institute of Science and Technology (MIST) for providing their all put support during the thesis work.

Finally, we would like to thank our course mates for their appreciable assistance, patience and suggestions during the course of our thesis.

Declaration by Candidates

This is to certify that the work presented in this thesis paper is the outcome of the investigation and research carried out by the following students under the supervision of Dr. Engr. Md. Alamgir Hossain, Associate Professor, MIST, Dhaka, Bangladesh.

It is also declared that neither this thesis paper nor any part thereof has been submitted anywhere for the award of any degree, diploma or other qualifications.

Rahid-uz-zaman
201018042

Miftahur Rahman
201018043

Yeasin Arafat Bhuiyan
201018053

Supervisor certificate

This is to certify that, Rahid-uz-Zaman, student ID: 201018042; Miftahur Rahman, student ID: 201018043; Yeasin Arafat Bhuiyan, student ID: 201018053 have completed their undergraduate project and thesis entitled "Development of an Integrated Control system for a Humanoid Robot". This paper embodies original work done under my supervision.

Dr. Engr. Md. Alamgir Hossain
Associate Professor
Department of Mechanical Engineering
Military Institute of Science and Technology

Nomenclature

DMP	Digital Motion Processing
ICS	Integrated Control System
α	Pan Angle
Θ	Tilt Angle
λ	Focal length of Camera
x,y,z	Camera Axis
X,Y,Z	World Axis
NI	National Instrument
C	Camera point
P	World point
W_h	Homogeneous co-ordinates of world
P_{-1}	Inverse Imaging Transformation
C_h	Camera Homogeneous Transformation
D_e	Euclidian Distance
D_4	City-Block Distance
D_8	Chess-Board Distance
S	Transformed Image
T	Transformation Function
r	Input image
INS	Inertial navigation system
IMU	Inertial measurement unit
DoF	Degree of freedom
MEMs	Micro electro-mechanical system
ARM	Adaptive Resolution Method
AI	Artificial Intelligence

Table of Contents

List of Figures

CHAPTER 1

INTRODUCTION

Chapter 1: Introduction

1.1 Robot and robotics

The meaning of robot is a mechanical man or a more than humanly efficient automation. It is an automatic apparatus or device that performs functions described to human beings or operates with what appears to be almost human intelligence. But this defamation does not give a human shape to the robot. The robot does the work of a human being.

The Robot Institute of America has given a very interesting definition on an Industrial robot. Industrial robots are usually used at industries. "An Industrial robot is a reprogrammable, multifunctional manipulator designed to move materials, parts, tools or special devices through variable programmed motions for the performance of a variety of tasks" [9].

So, a robot is capable of being reprogrammed. This characteristic distinguishes it from a fixed automation. A fixed automation is designed to do one and only one specific task. If the specification of task changes even slightly the fixed automation becomes incapable of performing the task. It is designed to perform according to one fixed specification.

However, a robot can be reprogrammed to perform even when specifications are changed drastically. The original programme is simply erased and the new programme takes care of the changed tasks. This feature that a robot can be reprogrammed makes the robot a flexible device. Hence, manufacturing systems which use robots are called flexible manufacturing system (FMS) due to this flexibility.

Karle Capek was the first person to introduce the word 'robot' and Sir Isaac Asimov coined first time the word 'robotics'. According to Asimov, robotics is the science of dealing with robots. Hence robotics involves a scientific study of robots.

The study includes design, selections of materials of proper quality for the components, fabrication, study of various motors required for moving the components, design of electronic circuits, computers and computer programming and control of robots.

Since robots and robotics are still in the developing stages, a considerable amount of research is being pursued in this line. Depending on the area in which robots are to be used, robotics includes disciplines such as biology, medical science, psychology, agriculture, mining, outer space engineering etc.

Mainly two types of robots are there, i.e. fixed robot and mobile robot. The fixed type robot is fixed to a particular location while doing his work with his hands. A mobile robot moves from place to place. Mobility is given to robots by providing wheels or legs or other crawling mechanisms.

1.2 History of robot and robotics

This segment is deliberated to offer you with a summary of the history of robotics. As you may have presumed, the robotics history is entangled with the history of science, technology and the fundamental principles of progress in technology employed in electronics, computers, even pneumatics & hydraulics can all be measured as a fraction of the robotics history [2]. Robotics at present symbolizes one of the mankind's supreme achievements and is the only best endeavor of mankind to create an artificial, electronic being.

Though robots are regarded as a 20th century discovery, their origins lie in the far history. From the initial time, public have shaped myths regarding automatic beings built-in their individual likeness with extraordinary human powers. The prehistoric age around 270BC Greeks & Egyptians manufactured mechanical machines to execute easy tasks. In modern times, automatic toys amuse and ever more complex machinery was invented. The thought of a realistic motorized humanoid monster named as "Frankenstein" in the year 1818 surveys what occurs when a man-made giant is gifted life by a knowledgeable scientist (Dr. Frankenstein). As the advancement in the computer technology progressed at a great pace, scientists became more fascinated in construction of intellectual machines that can ultimately have some logic to work themselves. At present, robots of all types occupy our globe and are brought into play for diverse applications in space discovery, the armed forces, medication industry, exploration, police work and of course movies.

Though the division of Robotics is new, the making of Robots initiated in the year 1250 when the first man-made automated human (Robot) was developed. In the phase from 1250 to 1950 the Robots were created for entertaining rather than for applications [3].

Here are a number of highlights in the history of robotics in the 20th century**:**

- In the year 1921, the Czech dramatist “Karel Capek” coins the world by using the word robot in his play Rossum’s Universal Robots (R.U.R). This word robot is derived from a Czech word which means “compulsory labor.”
- “Runaround” was composed by Asimov about robots in the year 1942, it held the “Three rules for robots”
  - Robots are not harmful to the humans, or through working, permit a human to come and damage.
  - A robot must follow the commands given by human beings apart from where such instructions would conflict with the First Law of Robotics.
  - A robot must defend its own survival providing such safety does not clash with the First and the Second Law of Robotics.
- In the year 1956, George Devol and Joseph Engelberger established the first robot company.
- In the year 1959, computer assisted manufacturing was verified at MIT.

- UNIMATE- The first industrialized robot was online in a General Motors automobile plant, in the year 1961.
- 1963 was a revolutionary year, first computer controlled robotic arm was designed and it was named as Rancho Arm. The invention was basically for the handicapped peoples.

The inventions in the field of Robotics were never ending and gave human beings a sudden surprising gift as & when launched. After Rancho's Arm various other inventions too were done, but all of the above was the first among all.

1.3 Essential characteristics and components of robot

There are some essential characteristics that a robot must have and this might help you to decide what is and what not a robot is. It will be also helpful to decide what features what will be needed to build into a machine before it can count as a robot [4].

A robot has these essential characteristics:

- **Sensing:** First of all your robot would have to be able to sense its surroundings. It would do this in ways that are not similar to the way that you sense your surroundings. Giving your robot sensors: light sensors (eyes), touch and pressure sensors (hands), chemical sensors, hearing and sonar sensors (ears) and taste sensors (tongue) will give your robot awareness of its environment.
- **Movement:** A robot needs to be able to move around its environment. Whether rolling on wheels, walking on legs or propelling by thrusters a robot needs to be able to move. To count as a robot either the whole robot moves, like the Sojourner or just parts of the robot moves, like the Canada Arm.
- **Energy:** A robot needs to be able to power itself. A robot might be solar powered, electrically powered, battery powered. The way your robot gets its energy will depend on what your robot needs to do.

- **Intelligence:** A robot needs some kind of "smarts." This is where programming enters the pictures. A programmer is the person who gives the robot its 'smarts.' The robot will have to have some way to receive the program so that it knows what it is to do.

Basically robot has five major components such as,

(a) The Manipulator,

(b) The End effectors

(c) The Locomotion Device

(d) The Controller and

(e) The Sensors.

In a robot system, all the above five components are interfaced properly so that each component can work in a co-coordinated fashion for the effective and efficient functioning of the robot. In an industrial robot system, a mini computer is being used as the controller. Sensors are measuring instruments that measure quantities such as position, velocity, force, torque, proximity, temperature etc.

1.4 Applications of Robots

Currently, robots perform a number of different jobs in numerous fields and the amount of tasks delegated to robots is rising progressively. The best way to split robots into types is a partition by their application.

1. Industrial robots – These robots bring into play in an industrialized manufacturing atmosphere. Typically these are articulated arms particularly created for applications like- material handling, painting, welding and others. If we evaluate merely by application then this sort of robots can also consist of some automatically guided automobiles and other robots.

2. Domestic or household robots – Robots which are used at home. This sort of robots consists of numerous different gears for example- robotic pool cleaners, robotic sweepers, robotic vacuum cleaners, robotic sewer cleaners and other robots that can perform different household tasks. Also, a number of scrutiny and tele-presence robots can also be considered as domestic robots if brought into play in that sort of environment.

3. Medical robots – Robots employed in medicine and medicinal institutes. First & foremost surgical treatment robots. Also, a number of robotic directed automobiles and perhaps lifting supporters.

4. Service robots – Robots that cannot be classed into any other types by practice. These could be various data collecting robots, robots prepared to exhibit technologies, robots employed for research, etc.

5. Military robots – Robots brought into play in military & armed forces. This sort of robots consist of bomb discarding robots, various shipping robots, exploration drones. Often robots at the start produced for military and armed forces purposes can be employed in law enforcement, exploration and salvage and other associated fields.

6. Entertainment robots – These types of robots are employed for entertainment. This is an extremely wide-ranging category. It begins with model robots such as robosapien or the running photo frames and concludes with real heavy weights like articulated robot arms employed as movement simulators.

7. Space robots – I would like to distinct out robots employed in space as a split apart type. This type of robots would consist of the robots employed on Canadarm that was brought into play in space Shuttles, the International Space Station, together with Mars explorers and other robots employed in space exploration & other activities.

8. Hobby and competition robots – Robots that is created by students. Sumo-bots, Line followers, robots prepared merely for learning, fun and robots prepared for contests. Now, as you can observe that there are a number of examples that fit well into one or more of these types.

CHAPTER 2

LITERATURE REVIEW

Chapter 2: Literature Overview

2.1 What is humanoid robot?

Humanoid Robotics includes a rich diversity of projects where perception, processing and action are embodied in a recognizably anthropomorphic form in order to emulate some subset of the physical, cognitive and social dimensions of the human body and experience. Humanoid Robotics is not an attempt to recreate humans. The goal is not, nor should it ever be, to make machines that can be mistaken for or used interchangeably with real human beings. Rather, the goal is to create a new kind of tool, fundamentally different from any we have yet seen because it is designed to work with humans as well as for them. Humanoids will interact socially with people in typical, everyday environments.

At present, Humanoid Robotics is not a well-defined field, but rather an underlying impulse driving collaborative efforts that crosscut many disciplines. Mechanical, electrical and computer engineers, roboticists, computer scientists, artificial intelligence researchers, psychologists, physicists, biologists, cognitive scientists, neurobiologists, philosophers, linguists and artists all contribute and lay claim to the diverse humanoid projects around the world. Inevitably, some projects choose to emphasize the form and mechanical function of the humanoid body. Others may focus on the software to animate these bodies. There are projects that use humanoid robots to model the cognitive or physical aspects of humans. Other projects are more concerned with developing useful applications for commercial use in service or entertainment industries. At times, there are deep ideological and methodological differences. For example, some researchers are most interested in using the human form as a platform for machine learning and online adaptation, while others claim that machine intelligence is not necessary.

2.2 Vision system in humanoid robot

Robots can see the world in three dimensions with vision systems already used at least on a limited scale in robotics. A more accurate vision means high details for the environment and this could be reached with stereo vision camera and the new entry in 3D interior mapping the MatterPort mapping camera.

A vision system that allows robots to see the world in 3D will be primarily used in advanced robotic application in the future and this is a good reason to invest time and money in developing new advanced 3D vision systems. The optical vision system was designed to be able to detect obstacles in the path of the robot. The vision system also tracks moving objects. Each eye consists of a lens in front of a 4×4 array of light dependent resistors. The resistance values are sampled at 30 frames per second. A single servo motor enables the two eyes to converge. The left eye analyses the image for verticals, horizontals, top right to bottom left edges and top left to bottom right edges. The right eye detects movement of dark to bright edges and can pan and tilt, with the left eye, to make such edges fall on the center of the sensor array. Compensation is provided for low and high levels of ambient lighting giving an automatic iris effect. The system reliably tracks moving objects but is currently confused by very high contrast lighting such as spotlights. The system is however working more reliably than an ultrasonic system which was tried on the earlier prototypes. Difficulties encountered with that system were multiple reflections resulting from low angular resolution (a consequence of the wide beam width due to using small diameter transducers), specular reflection from smooth objects, absorption from soft objects and motor interference (sound and electrical). Some of the difficulties are experienced. The difficulties are greater with electromagnetic and sonar range measurement systems in low cost autonomous mobile robots where lightness and low current consumption is particularly important. It is hoped that implementing stereoscopic vision using convergence of the two 16 sensor arrays will give adequate range resolution for obstacle avoidance.

2.3 Control system in humanoid robot

The on board control system uses interconnected microprocessors communicating in a hierarchical structure The master processor controls all the robot functions. The sensor and drive signals are processed in parallel. Each eye, arm and leg and has its own dedicated processor. The three balance sensor groups share two processors. Communication between processors is serial at 9600 Baud. Each servo motor is driven to a resolution of 256 steps. Each microprocessor has an individual resonator to control its clock speed and each processor is run at a speed appropriate to its task. The only significant restraint was the minimum of 4 MHz which is required to run the 9600 Baud serial communications. The range of clock speeds used is from 8 MHz to 50 MHz. As much processing as possible, especially for the basic functions, is done close to the sensors. There have been difficulties associated with the inter-processor communications, electrical interference, the time taken to process the input sensor signals and generate command signals, and difficulties setting up a synchronized command structure. The objective was to produce a scale humanoid at near minimal cost.

2.4 Current research Projects

The research of autonomous robots is one of the most important challenges in recent years. Among the numerous robot researches, the humanoid robot soccer competition is very popular. The robot soccer players rely on their vision systems very intensively when they are in the unpredictable and dynamic environments. This vision system can help the robot to collect various environment information as the terminal data to complete the functions of robot localization, robot tactic, barrier avoiding, etc. It can reduce the computing complexity by using our proposed approach, adaptive resolution method (ARM), to recognize the critical objects in the contest field by object features which can be obtained easily. The experimental results indicate that the proposed approach can increase the real-time and accurate recognition efficiency.

Humans interact with continuously flowing, diverse stimulation. Likewise, humanoids must have multi-modal perceptual systems that can seamlessly integrate sensors. One way to do this is to allow sensors to continually compete for dominance. At the Electro Technical Laboratory in Japan, G.Cheng and Y. Kuniyoshi have developed a humanoid with 24 degrees of freedom, joint receptors with encoders and temperature sensing. The humanoid uses 6 PCs for control of hearing, vision, motor output and integration. The robot itself is lightweight and flexible, allowing it to interact comfortably and safely with humans. Throughout a visual and auditory tracking task, the robot tracks a person by sight and/or sound while mimicking the upper body motion of a person. The focus of the work was in showing that the robot can track people using a multiple sensory approach that is not task-specific and does not need to switch between sensor modalities. The key is that perceptual subsystems necessary for mimicry, tracking, vision and auditory processing should not be thought of as separate tasks and pursued separately, but as essential capabilities that must together contribute to high-utility humanlike behavior.

This said, humanoid roboticists agree that vision is the most crucial sensing modality for enabling rich, humanlike interactions with the environment. Of course, computer vision has long been a hard problem and an essential study in and of itself. The first main problem is that many factors are confounded into image data in a many-to-one mapping. Another problem is the amazing amount of data to be processed. For a long time, computer vision research assumed that the goal was to acquire as much data about the environment as possible. This approach proved computationally intractable.

KHR

KHR-3 [6] is our latest humanoid robot. Its height and weight are 125cm and 55Kg. The robot has been upgraded from KHR-2. Its mechanical stiffness of links and reduction gear capacity of joints have been modified and improved. Increased stiffness makes the robot have low uncertainty in joint angle and link position, which can affect its stability Joint and link shape are seriously designed to fit with its art design concept. The joint controller, motor drive, battery, sensors and main controller (PC) are designed to be installed in the robot itself.

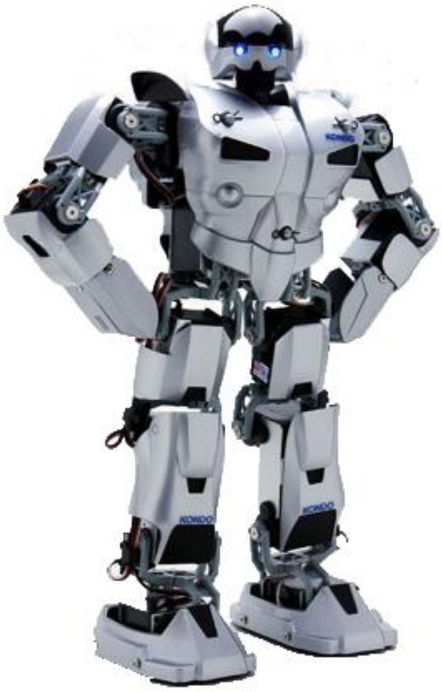


Figure 2.1 KHR humanoid robot

HUBO

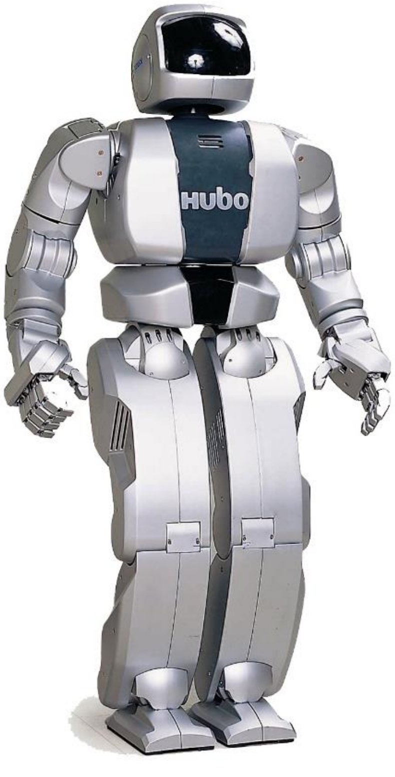


Figure 2.2 HUBO humanoid robot

HUBO [3] is a biped walking humanoid robot developed by the Humanoid Robot Research Centre at KAIST. It is 125cm tall and weighs 55kg. The inside frame is composed of Aluminum alloy and its exterior is composite plastic. A lithium polymer battery located inside of HUBO allows the robot to be run for nearly 90 minutes without external power source. All electrical and mechanical parts are located in the body, and the operator can access HUBO using wireless communications. HUBO can walk forward, backward, sideways, and it can turn around. Its maximum walking speed is 1.25km/h and it can walk on even ground or on slightly slanted ground. HUBO has enough degrees of freedom (DOF) to imitate human motions. In particular, with five independently moving fingers, it can imitate difficult human motions such as sign language for deaf people. Additionally, with its many sensors HUBO can dance with humans. It has two CCD cameras in its head that approximate human eyes, giving it the ability to recognize human facial expressions and objects. It can also understand human conversation, allowing it to talk with humans. . HUBO is an upgraded version of KHR-2. The mechanical stiffness in the links was improved through modifications and the gear capacity of the joints was readjusted. The increased stiffness improves the stability of the robot by minimizing the uncertainty of the joint positions and the link vibration control. In the design stage, features of the exterior, such as the wiring path, the exterior case design and assembly, and the movable joint range were critically reconsidered.

HRP

Kawada Industries, Inc., a subsidiary of Kawada Technologies, Inc., has developed HRP-4 [9], a new research and development platform for working humanoid robots, in collaboration with the National Institute of Advanced Industrial Science and Technology. In this joint project, Kawada Industries developed the humanoid robot hardware, while Fumio Kaneshiro (Senior Research Scientist), Humanoid Research Group (Leader: Kazuhiro Yokoi), the Intelligent Systems Research Institute of AIST and other members developed the motion control software.

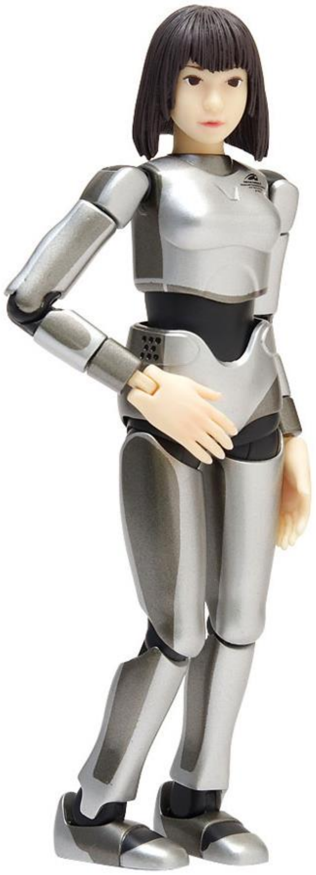


Figure 2.3 HRP-4 humanoid robot

There high-density implementation technology used for HRP-4C, the cybernetic human developed by AIST, is applied to HRP-4 [w]. HRP-4 has a total of 34 degrees of freedom, including 7 degrees of freedom for each arm to facilitate object handling and has a slim, lightweight body with a height of 151 cm and weight of 39 kg. Furthermore, the HRP-4 control system adopts a software platform OpenRTM-aist and the Linux kernel with the RT-Preempt patch. Therefore, many domestic and international software assets for robot systems, including OpenHRP3, an open-source robot simulator, can be utilized. HRP-4 is expected to accelerate the research and development of next-generation robot systems necessary for the future robot industry, such as human-cooperative robots capable of operating under various environments.

NAO

NAO [8] is a humanoid robot created by French company Aldebaran Robotics. It is also an official robot for RoboCup Standard Platform League. NAO is about 58 cm tall, and has a variety of sensors: 2 axis gyrometer, 3 axis accelerometer, Hall Effect sensors, tactile sensor, bumpers, channel sonars, I/R, but also microphone, loudspeakers, CMOS cameras, and Ethernet and Wi- connection. The "heart" of the robot is 500MHz x86 AMD GEODE processor, with 256MB SDRAM and 2GB memory. It runs an embedded Linux OS, and can be programmed with C, C++, Urbi, Python, and .NET. NAO is the most widely used humanoid robot for academic purposes worldwide.

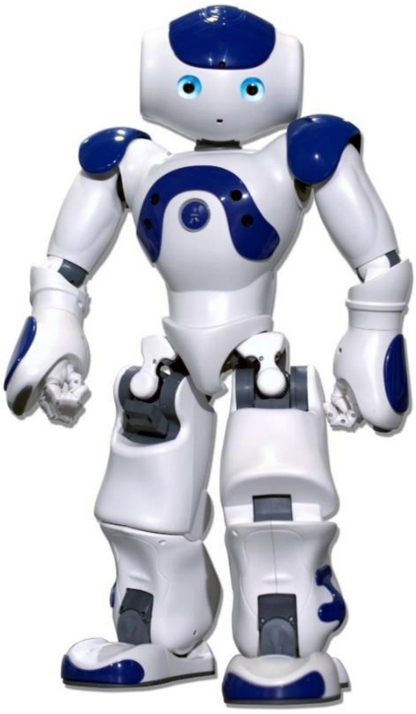


Figure 2.4 NAO humanoid robot

It is a versatile platform used to explore a wide variety of research topics in robotics as well as computer science, human-machine interaction, and the social sciences. NAO’s many sensors and actuators, convenient size, and attractive appearance, combined with sophisticated embedded software, makes it a unique humanoid robot ideal for many research fields.

ASIMO

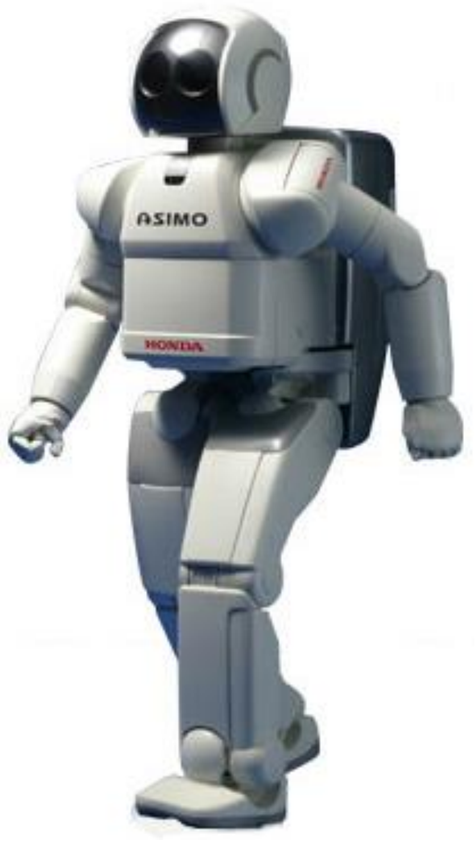


Figure 2.5 ASIMO humanoid robot

ASIMO [9] is the world's most advanced humanoid robot, developed by the Japanese company Honda. The first ASIMO was completed after 15 years of research, and it was officially unveiled on October 31, 2000. The robot resembles a small astronaut wearing a backpack, and is capable of performing a variety of tasks, including running, kicking a ball, walking up and down stairs, and recognizing people by their appearance and voice. The name is short for "Advanced Step in Innovative Mobility" and is also known as a abbreviation of ashita no mobility, meaning 'mobility in the future.' It was named in reference to Isaac Asimov, an American professor and science fiction writer who is credited with coining the term robotics and proposing the three laws of robotics. Based on this concept, ASIMO's design concerns three main elements, which are human-friendliness, adaptability to the human environment, and engineering feasibility.

The robot's height was set at 120 cm (or 130 cm in the case of second-generation ASIMO), which is similar to a child's, as this would be practical both on the engineering aspect (since a smaller and lighter robot is less challenging than an adult-sized robot such as the P2 prototype) and the question of operability in the environment, where light switches are normally located 110 cm from the floor. With less bulk, the robot would be able to move more efficiently in handling obstacles and narrow passages, and it would also be less overwhelming presence to humans and, in case of accidents, less hazardous.

2.5 Ethical considerations

The world's population of real humans continues to steadily grow. One might ask why it is necessary to make a machine that looks, thinks and emotes like a human when there are of humans already, many of whom do not have jobs or good places to live. It is important to re-emphasize that humanoids cannot and will not ever replace humans. Computers and humans are good at fundamentally different things. Calculators did not replace mathematicians. They did change drastically the way mathematics was taught. For example, the ability to mentally multiply large numbers, although impressive, is no longer a highly valued human capability. Calculators have not stolen from us part of what it means to be human, but rather, free our minds for more worthy efforts. As humanoids change the contours of our workforce, economy and society, they will not encroach on our sovereignty, but rather enable us to explore and further realize the very aspects of our nature we hold most dear.

Speaking in purely utilitarian terms, emotion is the implementation of a motivational system that propels us to work, improve, reproduce and survive. In reality, many of our human "weaknesses" actually serve powerful biological purposes. Thus, if it is wanted to be useful, human-like robots, some motivational system can be given in robots.

Most likely, two distinct species of humanoids will arise: those that respond to and illicit human emotions and those are wished to simply to do work, day in and day out, without stirring our feelings. Some ethicists believe this may be a difficult distinction to maintain. On the other hand, many consider ethical concerns regarding robot emotion or intelligence to be moot. According to this line of reasoning, no robot really feels or knows anything that have been not told to feel or know. From this perspective, it seems unnecessary to give a second thought to our treatment of humanoids. They are not 'real.' They are merely machines.

CHAPTER 3

Balancing with Sensors

Chapter 3: Balancing with Sensors

3.1 Inertia measurement unit

The creation of autonomous robots is a task that many have undertaken for military, medical or other reasons. But no matter the reason, measurements of the position of the system relative to its environments are needed for it to act accordingly without an external intervention. A technique that humans have used to navigate through their environment is dead reckoning [10]. This approach consists of the calculation of current position with the use of the following: knowledge of an initial position, and measurements of speed and direction. The INS uses an equivalent approach with the help of Newton's laws.

A control loop systems is used that take readings of the IMU to allow the platform to return to its original position. Some application for this type of system could be a platform that holds a camera so that in case of undesired movements the camera continues filming in the same direction.

3.2 Accelerometer

Accelerometers allow the measurement of the acceleration on a single axis to be ascertainable thanks to Newtown's second law, F=ma where a could be represented as the sum of g (gravitational forces) and f, the acceleration produced by external forces of the object, thus a=g+f. Typically in NS there are three accelerometers orthogonal to each other to provide the acceleration reading in three different directions. It is not practical to use the entire mass of a system to determine its acceleration. Instead, a proof mass connected to a set of springs is used to measure the acceleration is found to be more efficient in discerning the acceleration. With the help of the Hook law, F= x.k where k is a constant, a property of the spring and x it's the stretch displacement of the spring the acceleration of the object on a single axis can be measured. This is the simplest type of accelerometers there far more accurate and expensive accelerometers that involve technology such as Solid-state ferroelectric accelerometer and Solution electrolytic accelerometer.

MMA7455L Accelerometer:

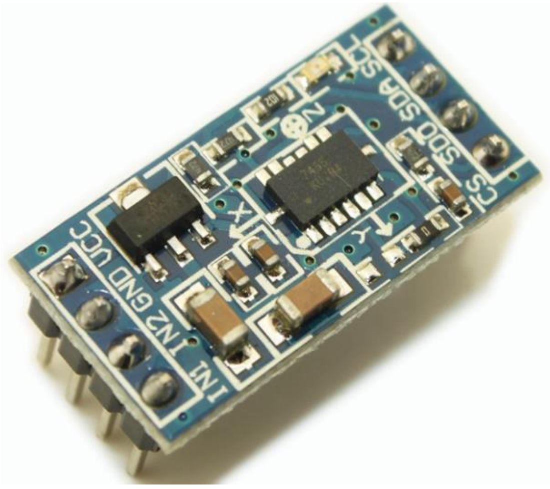


Figure 3.1 Accelerometer IC MMA7455L

The MMA7455L [11] is a Digital Output, low power, low profile capacitive micro machined accelerometer featuring signal conditioning, a low pass filter, temperature compensation, self-test, configurable to detect 0g through interrupt pins (INT1 or INT2), and pulse detect for quick motion detection. 0g offset and sensitivity are factory set and require no external devices. The 0g offset can be customer calibrated using assigned 0g registers and g-Select which allows for command selection for 3 acceleration ranges (2g/4g/8g). The MMA7455L includes a Standby Mode that makes it ideal for handheld battery powered electronics.

3.3 Gyro

A gyroscope is a device used primarily for navigation and measurement of angular velocity. Gyroscopes are available that can measure rotational velocity in 1, 2, or 3 directions. 3-axis gyroscopes are often implemented with a 3-axis accelerometer to provide a full 6 degree-of-freedom (DoF) motion tracking system.

Technically, a gyroscope is any device that can measure angular velocity. As early as the 1700ís, spinning devices were being used for sea navigation in foggy conditions. The more traditional spinning gyroscope was invented in the early 1800ís, and the French scientist Jean Bernard Leon Foucault coined the term gyroscope in 1852. In the late 1800ís and early 1900 ís gyroscopes were patented for use on ships. Around 1916, the gyroscope found use in aircraft where it is still commonly used today.

Throughout the 20th century improvements were made on the spinning gyroscope. In the 1960's, optical gyroscopes using lasers were first introduced and soon found commercial success in aeronautics and military applications. In the last ten to fifteen years, MEMS gyroscopes have been introduced and advancements have been made to create mass-produced successful products with several advantages over traditional macro-scale devices.

Traditional Gyroscope Function

Gyroscopes function differently depending on their type. Traditional pinning gyroscopes work on the basis that a spinning object that is tilted perpendicularly to the direction of the spin will have a precession. The precession keeps the device oriented in a vertical direction so the angle relative to the reference surface can be measured. Optical gyroscopes are most commonly ring laser gyroscopes. These devices send two lasers around a circular path in opposite directions. If the path spins, a phase shift can be detected since the speed of light always remain constant. Usually the rings are triangles or rectangles with mirrors at each corner.

Optical gyroscopes are a great improvement to the spinning mass gyroscopes because there is no ear, greater reliability and smaller size and weight.

MEMS Gyroscope

Even after the introduction of laser ring gyroscopes, a lot of properties were desired. MEMS vibrating mass gyroscopes aimed to create smaller, more sensitive devices. The two main types of MEMS gyroscope, discussed in Micro machined Vibrating Gyroscopes: Design and Fabrication, are the tuning fork gyroscope and the vibrating ring gyroscope.

Draper Tuning Fork Gyroscope

One of the most widely used micro-machined gyroscopes is the tuning fork design from the Charles Stark Draper Lab. The design consists of two tines connected to a junction bar.

Which resonate at certain amplitude. When the tines rotate, Coriolis force causes a force perpendicular to the tines of the fork. The force is then detected as bending of the tuning fork or a torsional force. These forces are proportional to the applied angular rate, from which the displacements can be measured in a capacitive fashion. Electrostatic, electromagnetic, or piezoelectric mechanisms can be used to detect the force.

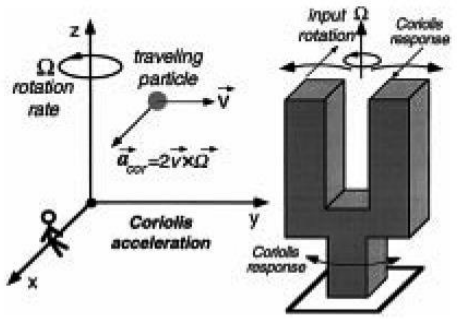


Figure 3.2 Draper tuning fork gyroscope

Since the development of their first tuning fork gyroscope in 1993, the Draper [12] Laboratory has made significant improvements to the device. Their first gyroscope was developed for the automobile industry. The gyroscope had command of 1 degree/hr drift, and possessed 4000 deg/hr resolution. These devices eventually functioned as the yaw rate sensor for skid control in. The first working prototype of the Draper Lab comb drive tuning fork anti-lock braking applications. Tests run on these sensors involve the examining the change in bias and error of such over a number of variables. Proper data could be retrieved in 0.8 s and sent to the necessary actuator to cause proper breaking in due time. These systems need to operate in a range of temperatures, specifically from -40 to 80 degrees Celsius. Over this range, both the bias error and the scale factor error are both quite stable. The bias error is approximately 2200 deg/h. Scale factor error was approximately 0.08%.

3.4 Accelerometer and Gyro combination

To measure the acceleration of a body, inertial sensors are used: accelerometers and gyroscopes. The accelerometer is used to measure acceleration in a single direction. By placing three accelerometers orthogonally to each other it is possible to sense the acceleration in any direction. From the gyroscope, the angular velocity can be obtained on a single axis. These two inertial sensors together can help to identify the change of position of a system. Inertial sensors are often attached to a fixed part of the vehicle or system so that the measurements that it senses are due to the system movement and not the movement of the sensor itself.

3-Axis Gyro/Accelerometer IC - MPU-6050

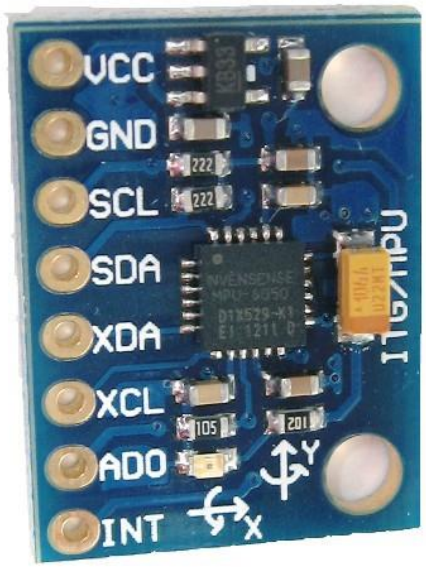


Figure 3.3 Gyroscope IC MPU 6050

The MPU-6050 [11] is a serious little piece of motion processing tech. By combining a MEMS 3-axis gyroscope and a 3-axis accelerometer on the same silicon die together with an onboard Digital Motion Processor™ (DMP™) capable of processing complex 9-axis Motion Fusion algorithms, the MPU-6050 does away with the cross-axis alignment problems that can creep up on discrete parts. The parts' integrated 9-axis Motion Fusion algorithms can even access external magnetometers or other sensors through an auxiliary master I2C bus, allowing the devices to gather a full set of sensor data without intervention from the system processor.

For precision tracking of both fast and slow motions, the MPU-6050 features a user-programmable gyro full-scale range of ±250, ±500, ±1000, and ±2000°/sec (dps) and a user-programmable accelerometer full-scale range of ±2g, ±4g, ±8g, and ±16g.

Features:

- ✓ Digital-output of 6 or 9-axis Motion Fusion data in rotation matrix, quaternion, Euler Angle, or raw data format

- ✓ Tri-Axis angular rate sensor (gyro) with a sensitivity up to 131 LSBs/dps and a full-scale range of ±250, ±500, ±1000, and ±2000dps
- ✓ Tri-Axis accelerometer with a programmable full scale range of ±2g, ±4g, ±8g and ±16g
- ✓ Reduced settling effects and sensor drift by elimination of board-level cross-axis alignment errors between accelerometers and gyroscopes
- ✓ Digital Motion Processing™ (DMP™) engine offloads complex Motion Fusion, sensor timing synchronization and gesture detection
- ✓ Embedded algorithms for run-time bias and compass calibration. No user intervention required
- ✓ Digital-output temperature sensor
- ✓ Digital input on FSYNC pin to support video Electronic Image Stabilization and GPS
- ✓ Programmable interrupt supports gesture recognition, panning, zooming, scrolling, free fall interrupt, high-G interrupt, zero-motion detection, tap detection, and shake detection
- ✓ VDD Supply voltage range of 2.375V–3.46V; VLOGIC at 1.8V±5% or VDD
- ✓ On-chip timing generator with ±1% frequency variation over full temperature range

3.5 Filter

Measured sensor data usually consists of noise which reduces accuracy. In order to get accurate estimates of the true value, the sensor data needs to be filtered. A good filtering [13] algorithm can eliminate noise from the data and retain useful information. Many types of filter used for IMU MPU 6050. Kalman filter and complementary filter are most popular and useful.

Kalman Filter:

It is an optimal recursive data processing algorithm depends on the criteria chosen to evaluate the performance. Since it does not require storage of all previous data and also does not require the previous data to be reprocessed, it is called a recursive sensor. Kalman Filter [14] is used in many system estimation applications like state estimation, digital signal processing, sensor integration, Navigational Systems, etc. Kalman Filter is frequently used for the purpose of filtering accelerometer data to give position and velocity coordinates.

Kalman filter has two phases:

- PREDICT: Predicts the state estimate of the current time using the state estimate of the previous time. The current measured values are not considered.
- UPDATE: Update phase combines the current time state estimate with the current measured values and updates the current state estimate.

The algorithm works by using a weighted average model on the predicted value and the current value. The more certain measurement is given more weight. The filter works in the discrete time domain.

The underlying concept of Kalman filter is a discrete time linear dynamic system [15]. This system depends on the following equations:

$$x_k = Ax_{k-1} \tag{3.1}$$

$$Z_k = Bx_k \tag{3.2}$$

Here,

- X_k is the state of the system at time k. It is based on the state of the system at time k-1. It is simply defined by its position, velocity and acceleration.

$$x_k = \begin{vmatrix} x \\ dx \\ d^2x \end{vmatrix} \tag{3.3}$$

- A is in the form of a matrix. It is an operation used to calculate the current state of the system from the previous state with assumption of constant acceleration.

- Z_k is the measured value of the system and it relates to the calculated value x_k. In perfect world, $Z_k = x_k$. But, in real life this may not happen because of noise.

Complementary filter:

Complementary filter [16], is in fact a combination of filters which manage both high-pass and low-pass filters simultaneously. The low pass filter filters high frequency signals (such as the accelerometer in the case of vibration) and low pass filters that filter low frequency signals (such as the drift of the gyroscope). This filter is widely used for inertial measurement sensor. The formula used for this filter is:

$$angle = 0.98 * (angle + gyro * dt) + (0.02 * accelerometer) \qquad (3.4)$$

Here, first portion of the right side (angle+gyro*dt) resembles a high pass filter on the gyro integrated angle estimation. It will have the same time constant as the low pass filter. Second portion of the right side (0.02*accelerometer) is the low pass filter acting on the accelerometer. The function "Complementary Filter" has to be used in a infinite loop. Every iteration the pitch and roll angle values are updated with the new gyroscope values by means of integration over time. The filter then checks if the magnitude of the force seen by the accelerometer has a reasonable value that could be the real g-force vector. If the value is too small or too big, it is surely known that it is a disturbance and don't need to take into account. Afterwards, it will update the pitch and roll angles with the accelerometer data by taking 98% of the current value, and adding 2% of the angle calculated by the accelerometer. This will ensure that the measurement won't drift, but that it will be very accurate on the short term.

CHAPTER 4

VISION

Chapter 4: Vision

4.1 Imaging geometry

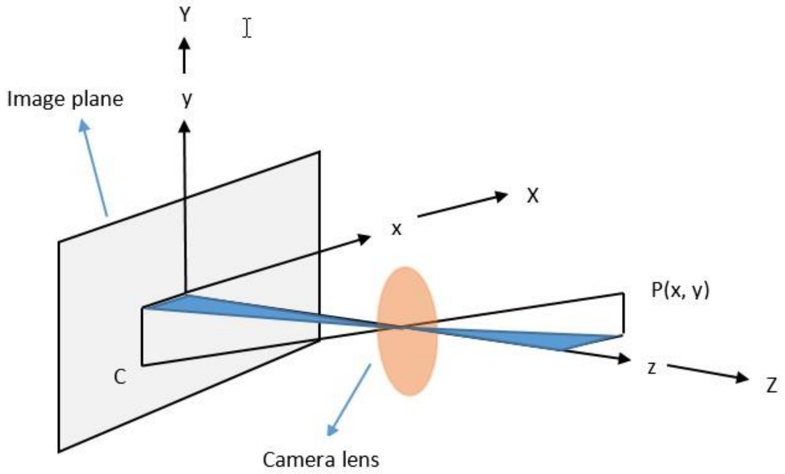


Figure 4.1 Imaging geometry

z = axis of lens or camera axis

x, y = image plane

P(x, y, z) = world point

C= camera point

C (x, y, z) is point on the image plane

X, Y, Z = world co-ordinate

x, y, z = camera co-ordinate

$$\frac{x}{\lambda} = \frac{X}{Z-\lambda}(-1) \tag{4.1}$$

$$\frac{x}{\lambda} = \frac{X}{\lambda-Z} \tag{4.2}$$

$$\frac{y}{\lambda} = \frac{Y}{\lambda-Z}(-1) \tag{4.3}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = P \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{4.4}$$

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = P^{-1} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{4.5}$$

$$W_h = P^{-1} C_h \tag{4.6}$$

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} kx \\ ky \\ kz \\ k \end{bmatrix} \tag{4.7}$$

To get real co-ordinates divided by the last vector

$$x = \frac{\lambda X}{\lambda - Z} \tag{4.8}$$

$$y = \frac{\lambda Y}{\lambda - Z} \tag{4.9}$$

$$z = \frac{\lambda Z}{\lambda - Z} \tag{4.10}$$

But z = 0 because as only the image plane are taken

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{\lambda} & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ \frac{\lambda - Z}{\lambda} \end{bmatrix} \tag{4.11}$$

Inverse Imaging Transformation:

$$P^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{\lambda} & 1 \end{bmatrix} \tag{4.12}$$

$$W_h = P^{-1} C_h \tag{4.13}$$

Here, W_h = homogeneous co-ordinates of world

P^{-1} = inverse imaging transformation and

C_h = camera homogeneous transformation

Let us consider,

$$P^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{\lambda} & 1 \end{bmatrix} \tag{4.14}$$

$$C_h = \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix} \tag{4.15}$$

So,

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{\lambda} & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix} = W_h \tag{4.17}$$

$$W_h = C_h \tag{4.18}$$

But, world coordinate and camera coordinate are co-incident, which are not very useful.

Without taking zero, consider z has a value.

$$\mathrm{C_h^*} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{4.19}$$

$$\mathrm{W_h^*} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{\lambda} & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ \frac{z}{\lambda} + 1 \end{bmatrix} \tag{4.20}$$

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} \frac{\lambda x}{\lambda + z} \\ \frac{\lambda y}{\lambda + z} \\ \frac{\lambda z}{\lambda + z} \end{bmatrix} \tag{4.21}$$

So,

$$Z = \frac{\lambda z}{\lambda + z} \; or, Z(\lambda + z) = \lambda z \;\; or, Z\lambda = z(\lambda - Z) \; or, z = \; \frac{\lambda Z}{\lambda - Z} \tag{4.22}$$

Now,

$$X = \frac{\lambda x}{\lambda + z} = \frac{\lambda x}{\lambda + \frac{\lambda - Z}{\lambda - Z}} = \frac{\lambda x(\lambda - Z)}{\lambda^2 - \lambda Z + \lambda Z} \tag{4.23}$$

Finally,

$$X = \frac{x}{\lambda}(\lambda - Z) \tag{4.24}$$

And

$$Y = \frac{y}{\lambda}(\lambda - Z) \tag{4.25}$$

So, if Z of world coordinate is known, can know other two world coordinates.

Z is necessary.

World coordinate may not co-inside with camera coordinate system, camera plane may not also be the same.

Now, consider a point in global coordinate W_h is given and the camera is mounted on a gimbal located at $W_0(x_0, y_0, z_0)$ having an offset, $\bar{r}$, with pan angle θ and tilt angle α.

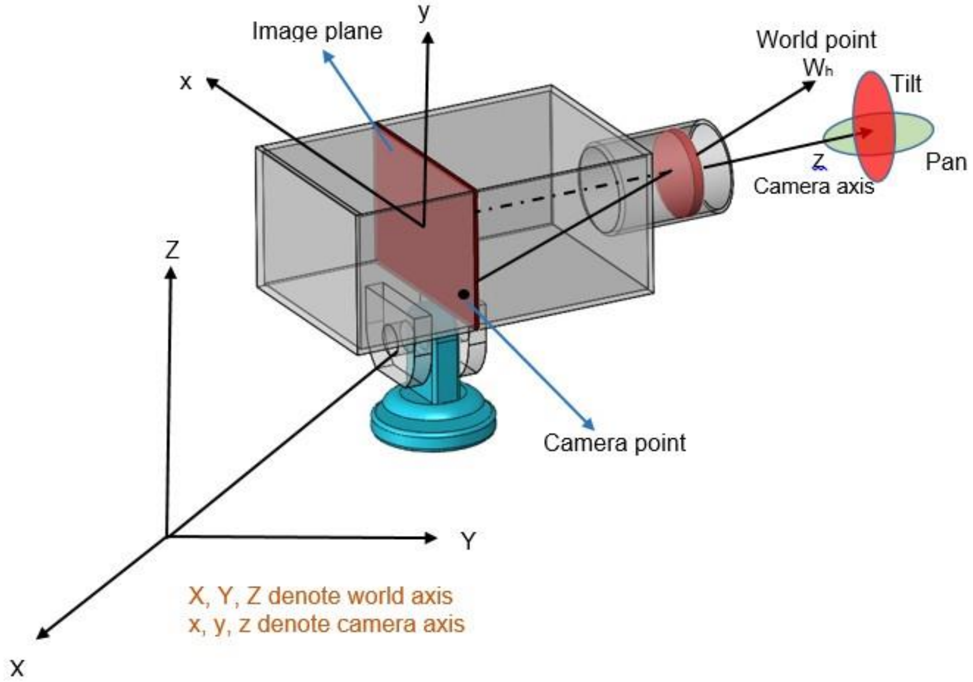


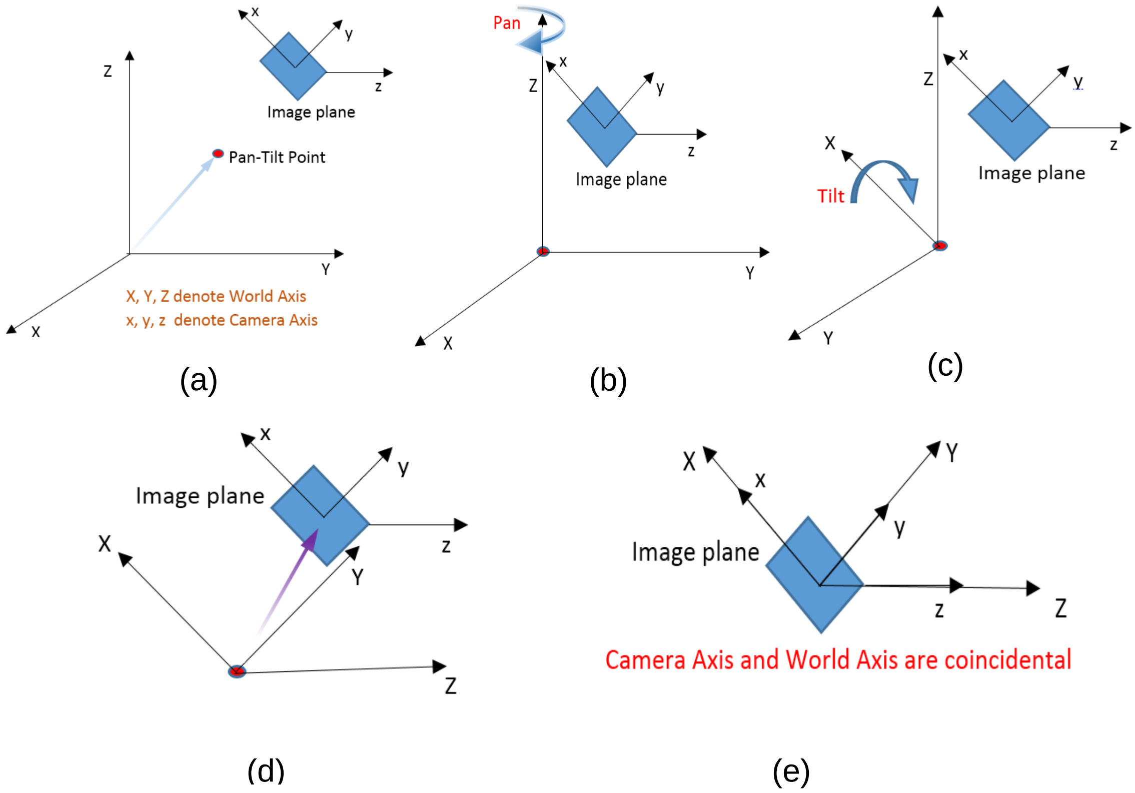
Figure 4.2 Relation between camera axis and world axis

Camera axis transforms in the following sequence like in Figure 11,

a) Original co-ordinates, b) rotation of world axis along Z, c) rotation of world axis along X axis, d) translation of world axis to camera point, f) coincidental camera and world axis.

To find the image coordinate, it can be assumed that G.F.C = C.F.C because of pan tilt camera for arbitrary rotation it is not possible.

$$\text{Translation}, T = \begin{bmatrix} 1 & 1 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$


x
y
z
Z
Image plane
Pan-Tilt Point
Y
X
X, Y, Z denote World Axis
x, y, z denote Camera Axis
Pan
Tilt
Image plane
Camera Axis and World Axis are coincidental

(a) (b) (c)

(d) (e)

Figure 4.3 Transformation of camera axis.

$$\text{Pan rotation (rotation with respect to Z)}, R_\theta = \begin{bmatrix} cos\theta & sin\theta & 0 & 0 \\ -sin\theta & cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{Translation}, T = \begin{bmatrix} 1 & 1 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{Pan rotation (rotation with respect to Z)}, R_\theta = \begin{bmatrix} cos\theta & sin\theta & 0 & 0 \\ -sin\theta & cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{Tilt rotation (rotation wt. X)}, R_\alpha = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\alpha & sin\alpha & 0 \\ 0 & -sin\alpha & cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{Translation of offset}, C = \begin{bmatrix} 1 & 0 & 0 & -r_1 \\ 0 & 1 & 0 & -r_2 \\ 0 & 0 & 1 & -r_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

So, the final transformation,

$$C_h = P.C.R_\alpha.R_\theta.T \qquad (4.26)$$

If camera coordinate and axis are not in line with real world axis:

$$C_h = (PGRT)W_h \qquad (4.27)$$

C_h = camera coordinate system

P = perspective transformation

G = gamble offset

R = rotation (pan and tilt)

T = translation of global axis to the gamble center

W_h = world homogeneous coordinates

$$x = \frac{\lambda(X-X_0)cos\theta+(Y-Y_0)sin\theta-r_1}{-(X-X_0)sin\theta sin\alpha+(Y-Y_0)cos\theta sin\alpha-(Z-Z_0)cos\alpha+r_2+\lambda} \qquad (4.28)$$

Here,

X, Y, Z = global axis

x = position of a point in x axis

λ = focal length of camera

$(X-X_0)$, $(Y-Y_0)$, $(Z-Z_0)$ defines the offset of gimbal center

Θ = pan angle

α = tilt angle

r_1, r_2, r_3 = offset of the gimbal of the image plane center wt. to the gimbal center

$$y = \frac{\lambda\{-(X-X_0)sin\theta cos\alpha+(Y-Y_0)cos\theta cos\alpha+(Z-Z_0)sin\alpha-r_2\}}{\{-(X-X_0)sin\theta sin\alpha+(Y-Y_0)cos\theta sin\alpha-(Z-Z_0)cos\alpha+r_2+\lambda\}} \qquad (4.29)$$

For 2 camera:

If 2 camera is used instead of 1 then one camera has to be coincidental.

Let, 1st camera co-ordinates are coincidental (coincidental means that camera axis and global axis are same)

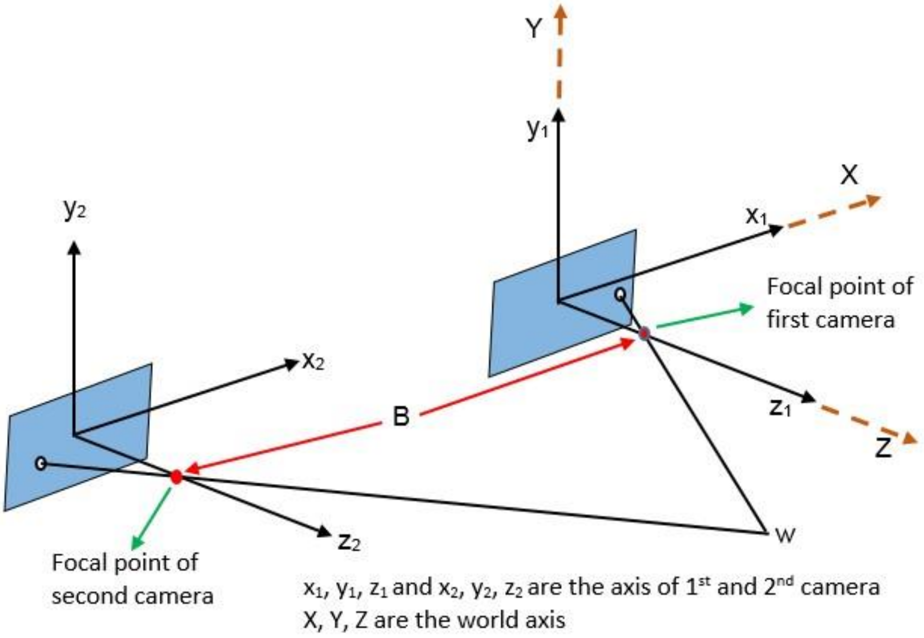

Y
y1
X
x1
y2
Focal point of first camera
x2
B
z1
Z
Focal point of second camera
z2
w
x1, y1, z1 and x2, y2, z2 are the axis of 1st and 2nd camera
X, Y, Z are the world axis

Figure 4.4 Relation of 2 camera axis with world axis.

$$X = \frac{x_1}{\lambda}(\lambda - z_1) \tag{4.30}$$

And $X_0 = -B$

$$X_2 = X_1 + B = \frac{x_2}{\lambda}(\lambda - z_2) \tag{4.30}$$

Here, $z_1 = z_2 = Z$

$$X_2 - X_1 = B \tag{4.31}$$

Or, $$\frac{x_2}{\lambda}(\lambda - Z) - \frac{x_1}{\lambda}(\lambda - Z) = B \tag{4.32}$$

Or, $$\lambda x_2 - Zx_2 - \lambda x_1 + Zx_1 = \lambda B \tag{4.33}$$

Or, $$\lambda(x_2 - x_1) + Z(x_1 - x_2) = \lambda B \tag{4.34}$$

Or, $$Z(x_1 - x_2) = \lambda B - \lambda(x_2 - x_1) \tag{4.35}$$

Or, $$Z = \frac{\lambda B}{-(x_2 - x_1)} - \frac{\lambda(x_2 - x_1)}{-(x_2 - x_1)} \tag{4.36}$$

So, $$Z = \lambda - \frac{\lambda B}{(x_2 - x_1)} \tag{4.37}$$

Now, $$X_1 = \frac{x_1}{\lambda}\left(\lambda - \lambda + \frac{\lambda B}{(x_2 - x_1)}\right) \tag{4.38}$$

$$= \frac{x_1}{\lambda} . \frac{\lambda B}{(x_2 - x_1)} = \frac{x_1 B}{x_2 - x_1}$$

Similarly, $$Y_1 = \frac{y_1 B}{x_2 - x_1} \tag{4.39}$$

So, global location of the point will be (X1, Y1 and Z1) if 2 camera are used instead of 1 camera.

4.2 Automatic camera calibration

In geometrical camera calibration, the objective is to determine a set of camera parameters that describe the mapping between 3-D reference coordinates and 2-D image coordinates. Camera calibration in the context of three-dimensional machine vision is the process of determining the internal camera geometric and optical characteristics (intrinsic parameters) or the 3-D position and orientation of the camera frame relative to a certain world coordinate system (extrinsic parameters) [9]. Sometimes, both the parameters can be determined from the calibration. In many cases, the overall performance of the machine vision system strongly depends on the accuracy of the camera calibration.

Here, calibration of the camera using a 3D object is discussed. This technique is applicable if and only if the following conditions are satisfied:

a. Calibration target: 2 planes are at right angle
b. positions of pattern corners only with respect to a coordinate system of the target is known

$$\overline{C_h} = (PCRT)W_h = AW_h \tag{4.40}$$

$$W_h = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \tag{4.41}$$

$$\overline{C_h} = \begin{bmatrix} C_{h1} \\ C_{h2} \\ C_{h3} \\ C_{h4} \end{bmatrix} = AxW_h = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \times \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{4.42}$$

And
$$\begin{bmatrix} C_{h1} \\ C_{h2} \\ C_{h3} \\ C_{h4} \end{bmatrix} = \begin{bmatrix} xk \\ yk \\ zk \\ k \end{bmatrix} \tag{4.43}$$

So,
$$x = \frac{xk}{k} = \frac{C_{h1}}{C_{h4}} \; and \; y = \frac{C_{h2}}{C_{h4}} \tag{4.44}$$

Now,
$$xk = xC_{h4} = a_{11}X + a_{12}Y + a_{13}Z + a_{14} \tag{4.45}$$

$$yC_{h4} = a_{21}X + a_{22}Y + a_{23}Z + a_{24} \tag{4.46}$$

$$C_{h4} = a_{41}X + a_{42}Y + a_{43}Z + a_{44} \tag{4.47}$$

Putting value of C_{h4} in Equation 4.45,

$$a_{11}X + a_{12}Y + a_{13}Z + a_{14} = x(a_{41}X + a_{42}Y + a_{43}Z + a_{44}) \tag{4.48}$$

Or,

$$a_{11}X + a_{12}Y + a_{13}Z + a_{14} - x(a_{41}X + a_{42}Y + a_{43}Z + a_{44}) = 0$$
(4.49)

For Equation 4.46

$$a_{21}X + a_{22}Y + a_{23}Z + a_{24} - y(a_{41}X + a_{42}Y + a_{43}Z + a_{44}) = 0$$
(4.50)

All the co-efficient are unknown in Equations 4.49 and 4.50.

X, Y, Z value for the 6 points of box are known from the camera.

Writing Equations 4.49 and 4.50 for the 6 points, it is possible to get 12 equations which are linear. By solving these 12 linear equations, easily the 12 unknown constant values can be found.

4.3 Image processing

Image processing is a method to convert an image into digital form and perform some operations on it, in order to get an enhanced image or to extract some useful information from it. It is a type of signal dispensation in which input is image, like video frame or photograph and output may be image or characteristics associated with that image. Usually Image Processing system includes treating images as two dimensional signals while applying already set signal processing methods to them.

The purpose of image processing is divided into 5 groups. They are:

a. Visualization - Observe the objects that are not visible
b. Image sharpening and restoration - To create a better image
c. Image retrieval - Seek for the image of interest
d. Measurement of pattern – Measures various objects in an image
e. Image Recognition – Distinguish the objects in an image

Image processing consists of following steps:

a. Image acquisition (sensing)
b. Preprocessing (reduction in noise and enhancement)
c. Segmentation (separating region)
d. Description (characteristics features)
e. Recognition (identify regions)
f. Interpolation (Assign meaning)

4.3.1 Basic concepts

4.3.1.1 Neighborhood of a pixel

4-neighbors:

Any pixel P(x, y) has two vertical and two horizontal neighbors, given by (x+1, y), (x-1, y), (x, y+1), (x, y-1). This set of pixels are called the 4-neighbors of P, and is denoted by N_4(P). Each of them are at a unit distance from P.

Diagonal Neighbors:

The four diagonal neighbors of P(x, y) are given by, (x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1). This set is denoted by N_D(P). Each of them are at Euclidean distance of 1.414 from P.

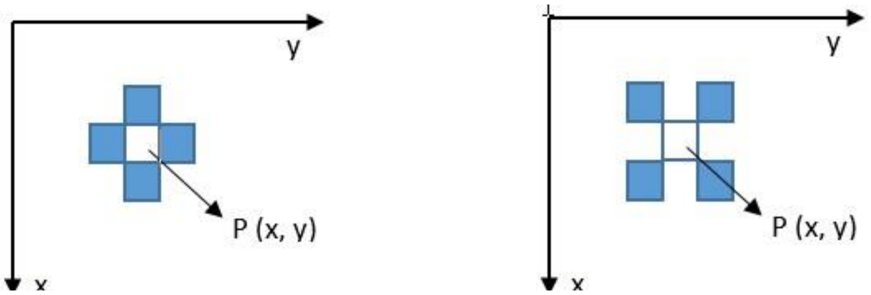


Figure 4.5 Neighborhood pixels

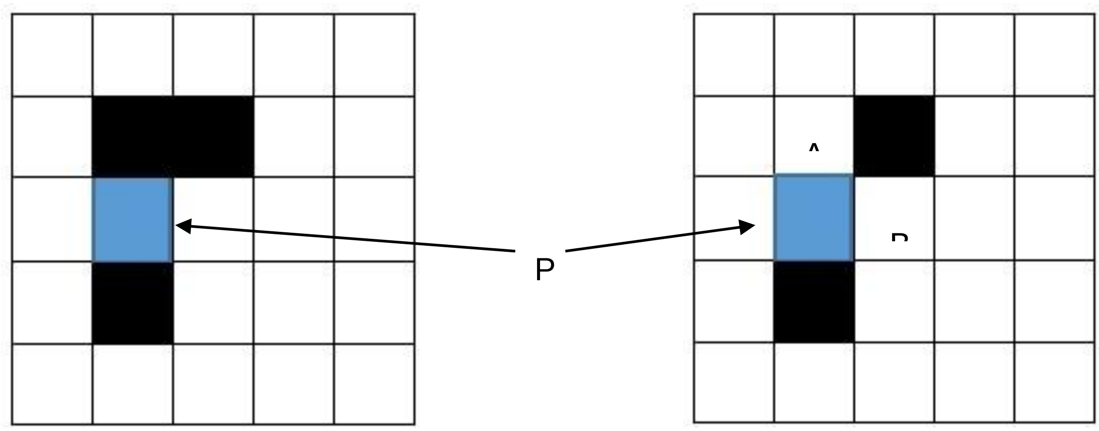
The points $N_4(P)$ and $N_D(P)$ are together known as 8-neighbors of the point P, denoted by $N_8(P)$. Some of the points in the N_4, N_D and N_8 may fall outside image when P lies on the border of image.

4.3.1.2 Connectivity

Two pixels are connected if they are neighbors and their gray levels satisfy some specified criterion of similarity. For example, in a binary image two pixels are connected if they are 4-neighbors and have same value (0/1).To determine the connectivity, a set of gray levels values or intensity levels value is defined.

3 types of connectivity:

- 4-connectivity: Two pixels P and Q with values from V are 4-adjacent if Q is in the set $N_4(P)$.
- 8-Connectivity: Two pixels P and Q with values from V are 8-adjacent if Q is in the set $N_8(P)$.
- m-adjacency: Two pixels P and Q with values from V are m adjacent if and only if the following conditions are satisfied:
  a. Q is in $N_4(P)$,
  b. Q is in $N_D(P)$,
  c. No common N_4 of both P and Q.



(a

(b)

Figure 4.6 Neighborhood pixels

In Fig (a) and (b), the black pixels are connected with P. But, in Fig 2, A and B pixels are not m-connected as they have common N_4.

4.3.1.3 Distance

- Euclidean Distance: The Euclidean Distance between p and q is defined as:

$$De\ (p,q) = \frac{[(x - s)2 + (y - t)2]1}{2} \tag{4.51}$$

Pixels having a distance less than or equal to some value r from (x, y) are the points contained in a disk of radius r centered at (x, y).

- City-block distance: The D_4 distance (also called city-block distance) between p and q is defined as:

$$D_4\ (p,q) = |\, x - s \,| + |\, y - t \,| \tag{4.52}$$

Pixels having a D4 distance from (x, y), less than or equal to some value r form a diamond centered at (x, y).

- *chessboard distance:* The (also called *D_8 distance*) between *p* and *q* is defined as:

$$D_8\ (p,q) = max(\ |x - s\,|, |y - t\,|\ ) \tag{4.53}$$

Pixels having a *D_8* distance from (x, y), less than or equal to some value r form a square centered at (x, y).

- **D_m distance:** It is defined as the shortest m-path between the points. In this case, the distance between two pixels will depend on the values of the pixels along the path, as well as the values of their neighbors.

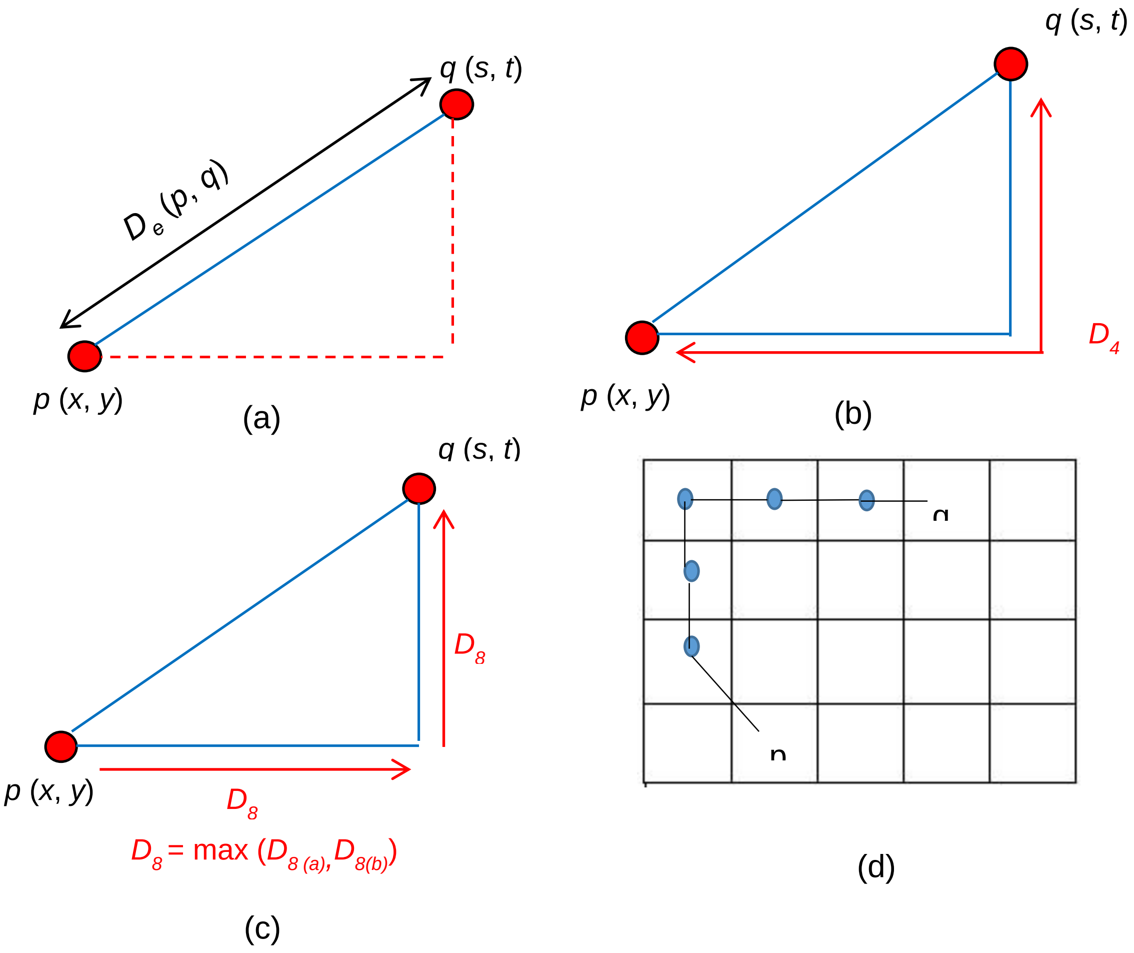


Figure 4.7 Distance between pixels.

Figure 14, (a) represents Euclidean distance, (b) represents City block distance, (c) represents Chessboard distance (d) represents the m-distance.

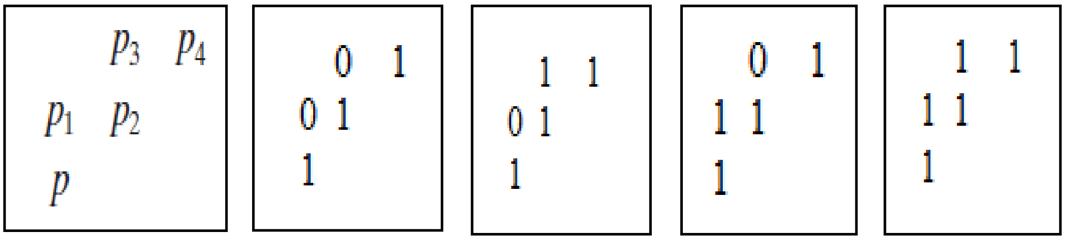
Example: Consider the following arrangement of pixels and assume that *p*, p_2, and p_4 have value 1 and that p_1 and p_3 can have can have a value of 0 or 1. Suppose that the adjacency of pixels values is 1 (i.e. $V = \{1\}$).

Now, to compute the D_m between point's *p* and p_4*, there are* 4 cases:

Case1: If $p_1 = 0$ and $p_3 = 0$

The length of the shortest m-path (the D_m *distance) is 2 (p, p_2, p_4)*

Case2: If p_1 =1 and p_3 = 0



(a) Initial Orientation *(b)* Case *1* *(c)* Case *2* *(d) Case 3* *(e) Case 4*

Figure 4.8 Mixed connectivity pixels distance

Now, p_1 *and p will no longer be adjacent (see m-adjacency definition)* then, the length of the shortest path will be 3 (*p*, p_1, p_2, p_4).

Case3: If p_1 =0 and p_3 = 1

The same applies here, and the shortest –m-path will be 3 (*p*, p_2, p_3, p_4)

Case4: If p_1 =1 and p_3 = 1

The length of the shortest m-path will be 4 (*p*, p_1, p_2, p_3, p_4).

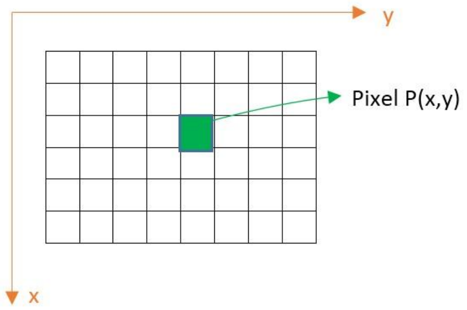
4.3.2 Techniques of Image processing

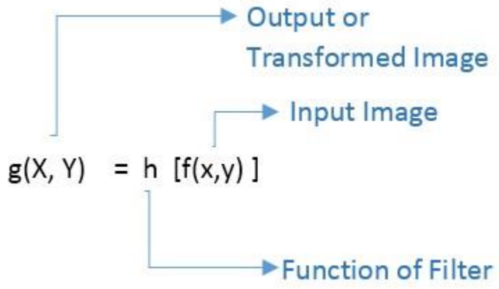
There are mainly two types of Image processing techniques.

a) Spatial Domain Processing
b) Frequency Domain Processing

Some basic things of Image processing are explained here.

For all different values of x and y belonging to the image-





Function or Filter depends on the properties of P, as well as neighboring pixels.

Pre-processing steps:

$$\sum_{\substack{j=-1,0,1\\k=-1,0,1}} Wjk\ f(x+j\,,y+k)$$

$=W_{-1-1}\ f(x-1\,,y-1)+$ $W_{-1\,0}\,f(x-1\,,y)+W_{-1\,1}\,f(x-1,y+1)+\cdots$

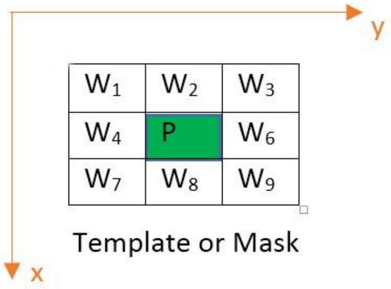


Figure 4.9 Template of an image

$$\begin{aligned}= W_1\,f(x-1,y-1)+W_2\,f(x-1,y)\,W_3\ f(x-1,y+1)+W_4 f(x,y-1)\\ +W_5\ f(x,y)+W_6\ f(x,y+1)+W_7\,f(x+1,y+1)\\ +W_8\,f(x+1,y)+W_9\,f(x+1,y+1)\end{aligned}$$

The sum is the new pixel value of P. This process will be continued to the whole image and will not be replaced by the value until the calculations of all pixels are done.

Special case:

If the template is 1*1 then the transformation of input pixel does not depend on the neighboring pixels.

$$S=T(r) \tag{4.54}$$

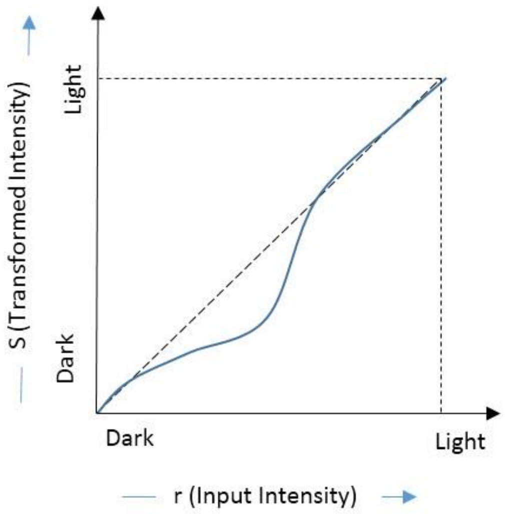


Figure 4.10 Intensity transformation

Where,

S= Transformed Image

T= transformation function

r = Input image

This is called Image Intensity Transformation (Mapping)

If the line in Fig 20 is straight then the all pixels variation of input and output will be same. As a result there will not be any significant change. Deviation of the graph line change the image like contrast increasing or decreasing.

Frequency Domain Approach:

Frequency Domain approach works based on the Fourier Transformation. It transformed a function to another function.

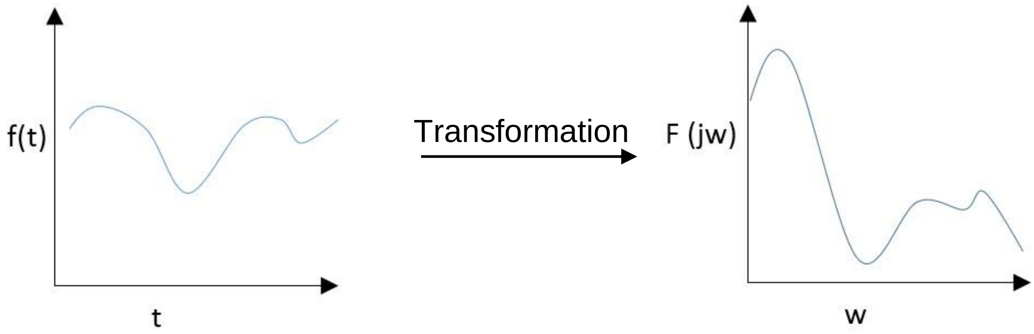


Figure 4.11 Transformed function after Fourier transform

Discrete Fourier Transformation:

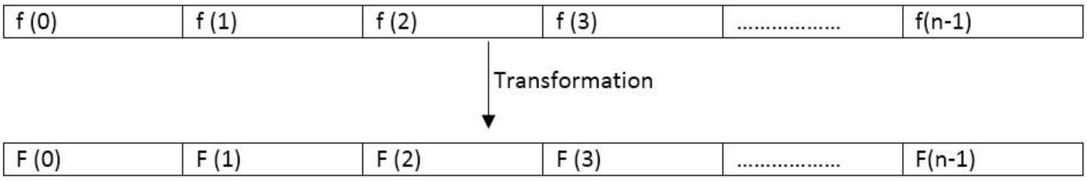
If F(u) is a function of Frequency and f(x) is a function of x then,

$$F(u) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)\; e^{\frac{-2\pi jux}{N}} \tag{4.55}$$

And the Inverse Transformation is

$$f(x) = \sum_{x=0}^{N-1} F(u)\, e^{\frac{2\pi jux}{N}} \qquad (4.56)$$

2-D discrete Fourier Transformation:



The 2-D transformation is according to x and y .So,

$$F(u, v) = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y)\, e^{\frac{-2\pi j(ux+vy)}{N}} \qquad (4.57)$$

And the Inverse Transformation is

$$f(x, y) = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} F(u, v)\, e^{\frac{2\pi j(ux+vy)}{N}} \qquad (4.58)$$

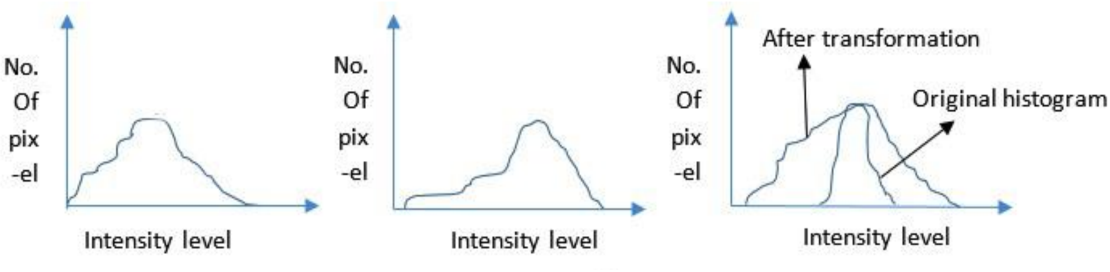
4.4 Enhancement

The principal objective of image enhancement is to process a given image so that the result is more suitable than the original image for a specific application. It accentuates or sharpens image features such as edges, boundaries, or contrast to make a graphic display more helpful for display and analysis. The enhancement doesn't increase the inherent information content of the data, but it increases the dynamic range of the chosen features so that they can be detected easily.

The greatest difficulty in image enhancement is quantifying the criterion for enhancement and, therefore, a large number of image enhancement techniques are empirical and require interactive procedures to obtain satisfactory results.

4.4.1 Histogram

The histogram of a digital image with grey levels in a certain range, is a discrete function [17]. It is the basic image enhancement process. The shape of the histogram of an image gives us useful information about the possibility for contrast enhancement. A histogram of a narrow shape indicates little dynamic range and thus corresponds to an image having low contrast.



(a) (b) (c)

Figure 4.12 Histogram (a) Original, (b) transformed, (c) comparison

Histogram transformation is nothing but intensity transformation. For continuous intensity level, histogram will be a probability density function.

Intensity transformation: S=T(r)

Here, S is the new intensity, T is the transformation function and r is original pixel intensity.

To normalize this, let us assume that,

$0 \leq r \leq 1$ and $0 \leq s \leq 1$

This two condition states that, if maximum intensity is 256 then divide 0 to 1 area into 255 portions so that highest level remain 1.

T^{-1} will be possible if and only if:

a. T is a monotonically increasing function and
b. Single valued function

Histogram equalization:

Equalization is done to get the final image intensity as intensity.

From Equation 4.54, r can be evaluated and the value of r will be T^{-1}(s).

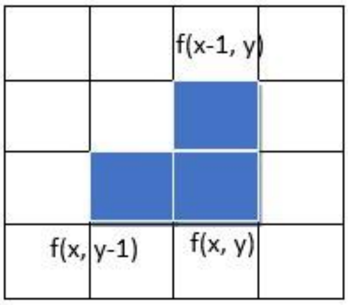
After transformation image will take different shape.

4.4.2 Making edges more prominent

Edge is the place where intensity changes suddenly.

$$G_x = \frac{\partial y}{\partial x} \sim f(x,y) - f(x-1,y)$$

$$G_y = \frac{\partial y}{\partial x} \sim f(x,y) - f(x,y-1)$$



To determine the edges only the gradient of the previous pixel is consider. The best way to determine the edge is to use SOBEL MASK or SOBEL OPERATOR [18]. If the image or pixels are uniform then it will be very near to zero. If the intensity changes in x direction, SOBEL MASK will give high value in G_x but G_y will not give high value [19]. If the intensity changes in oblique direction then the mask will give value in G_x and G_y.

$$G_y = \frac{\partial y}{\partial x} \sim f(x-1,y+1) - f(x-1,y-1) + 2f(x,y+1) - 2f(x,y-1) + f(x+1,y+1) - f(x+1,y-1)$$

Similarly, G_x can also be determined.

-1	-2	-1
0	0	0
-1	2	-1

G_x

-1	0	1
-2	0	2
-1	0	1

G_y

4.5 Methods of matching

Matching means comparison between two objects. For matching, a standard or precedent object has to be set which is compared with the later one. In image processing, the precedent image portion is called template. The template [20] selection is the most important part in image processing because it works as memory. A template has to be extracted to compare it with the real-time images. The image chosen for template extraction must have high quality, proper edges and have to be captured by a high resolution camera. As the processed image lost its quality, it is preferable to use high quality digital camera to get maximum accuracy. According to template matching, 3 types of matching is possible:

1. Color matching: Color matching compares the color content of an image or regions in an image to existing color information. The color information in the image may consist of one or more colors. To use color matching, regions have to be defined in an image that contain the color information for using as a reference. Machine vision functions then learn the 3D color information in the image and represents it as a 1D color spectrum [21]. MV application compares the color information in the entire image or regions in the image to the learned color spectrum, calculating a score for each region. This score relates how closely the color information in the image region matches the information represented by the color spectrum.

2. Pattern matching: Pattern matching locates regions of grayscale image that match a predetermined template. Pattern matching finds template matches regardless of poor lighting, blur, noise, shifting of the template or rotation of the template. It is used to quickly locate known reference pattern in an image. With pattern matching, a model or template is created that represents the object for which it is being search. Then, machine vision application searches for the model in each acquired image calculating a score for each match. The score relates how closely the model matches the pattern form. Pattern matching works based on the Cross Correlation and pyramid matching method. Mathematical process of image correlation is a simple method. Actually the template is laid over the source image and the intensity values for each corresponding pixel are individually multiplied. Then all of them are summed to produce a single correlation value. The template is then moved one pixel and the process is repeated until the whole source image has been covered and also based on it a matrices of correlation values has been created. The cross correlation [22] matrices is populated using the following equation:

Cross Correlation Matrix, j=

$$\sum_{x=0}^{b-1}\sum_{y=0}^{a-1}(Template_{x,y})(Source_{(i+x),(j+y)}) \quad (4.59)$$

If the images has not been normalized then it must normalized by dividing each element in the perspective images by the square root of the sum of its square:

Cross Correlation Matrix, i =

$$\frac{\sum_{x=0}^{b-1}\sum_{y=0}^{a-1}(Template_{x,y})(Source_{(i+x),(j+y)})}{\sqrt{(\sum_{x=0}^{b-1}\sum_{y=0}^{a-1}(Template_{x,y})^2)((\sum_{x=0}^{b-1}\sum_{y=0}^{a-1}(Source_{(i+x),(j+y)}))}} \quad (4.60)$$

3. Color pattern matching: Color pattern matching is a unique approach that combines color and spatial information to quickly find color patterns in an image. Color pattern matching tool locates the reference pattern in an image even when the pattern in the image is rotated and slightly scaled. When a pattern is rotated or scaled in the image, the color pattern matching tool detects the following features of an image:

- The pattern in the image
- The position of the pattern in the image
- The orientation of the pattern
- Multiple instances of the pattern in the image, if applicable

When color pattern matching is used to search for a template, the software uses the color information in the template to look for occurrences of the template in the image. The software then applies grayscale pattern matching in a region around each of these occurrences to find the exact position of the template in the image. The following figure illustrates the general flow of the color pattern matching algorithm. The size of the searchable region depends on the provided inputs, such as search strategy and color sensitivity.

4.6 Interfacing and mapping with Arduino

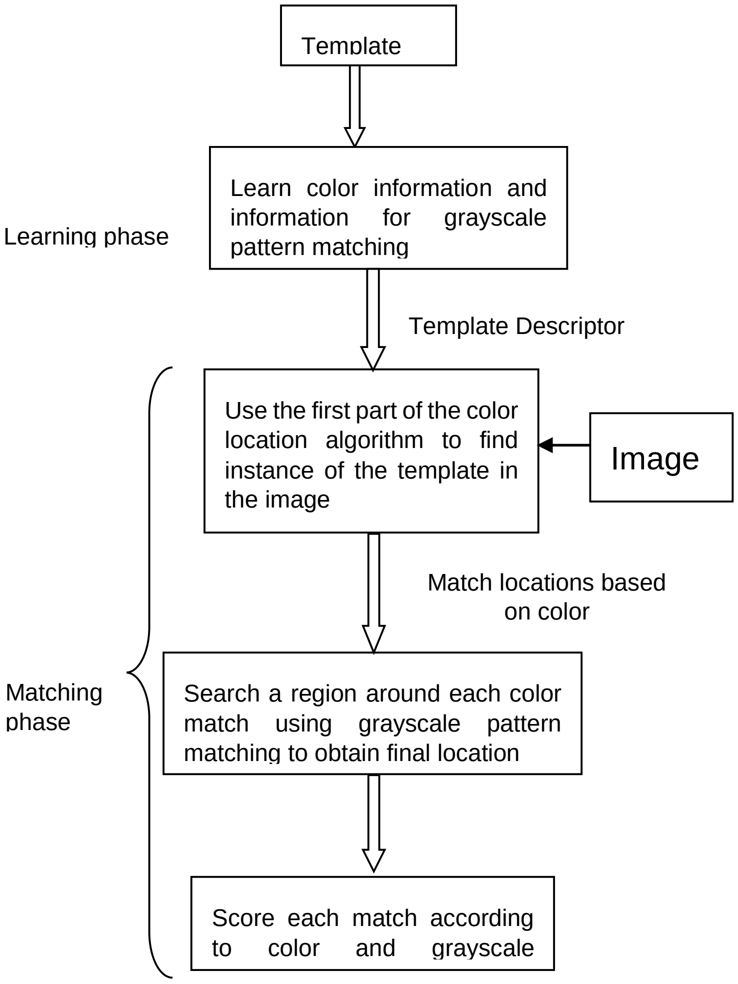

Template
Learn color information and information for grayscale pattern matching
Learning phase
Template Descriptor
Use the first part of the color location algorithm to find instance of the template in the image
Image
Match locations based on color
Matching phase
Search a region around each color match using grayscale pattern matching to obtain final location
Score each match according to color and grayscale

Figure 4.13 Algorithm for color matching

MISTBOY vision system consists of 2 high definition 8 megapixels camera of Genius brand which is capable of 640x480 video capture at 30 frames per second. In this paper only the ball detection algorithm using a single camera is discussed. The response time of a soccer playing robot should be as fast as possible to cope up with the speed of the ball. To decrease the response time it is needed to process the continuous inline images expeditiously. As the processing needs high definition images instantly, it is necessary to use very high configured processor instantly. But, it is not possible to use an onboard CPU because of the size and weight problem in balancing. So, the idea is to transfer the images through Wi-Fi to an external CPU for processing. After processing, the external CPU send the decisions to the onboard Arduino for actions.

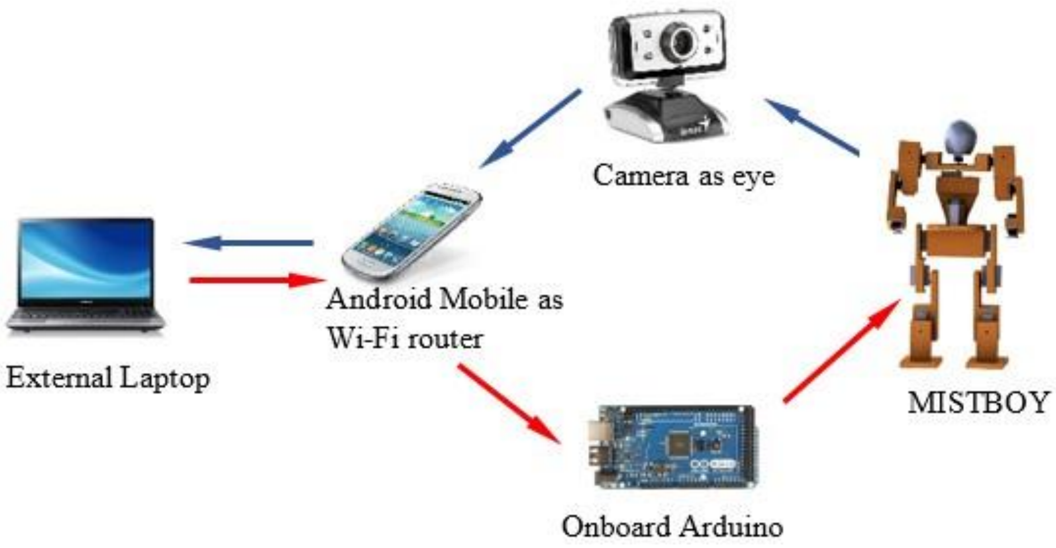


Figure 4.14 Integrated Vision system.

CHAPTER 5

Distance Measurements

5.1 Vision

5.2 Sonar

5.3 IR (Infrared Ray)

Chapter 5: Distance Measurement

5.1 Vision system

Distance measurement is one desired capability for an intelligent robot to understand its working environment. Among existing distance-measurement techniques, one category imitates the human vision and evaluates the distance using the spatial disparity of an object point in two images. The measurement system typically consists of a pair of cameras. The distance is computed using the disparity of two corresponding pixels with the triangulation. The two cameras must be carefully aligned and well calibrated to minimize the measurement inaccuracy. If the characteristics of two cameras are not identical due to a difference in fabrication, an impact or aging, a significant measurement error could be hard to avoid. Some researchers elaborated on the monocular vision for possibly overcoming the shortcomings of the stereo-vision measurement system. With two images taken at two different positions by a single camera, the distance information can be computed in the similar way as that with the stereo-vision. The robotic eye-in-hand system, which has a camera moved by a robot arm, is an example. Since the movement of the camera on the robot arm is Omni-directional, finding the matching points on images could be computationally costly. Hazard on the camera is more likely due to the frequent movement and impact. Physically moving the robot arm also causes a significant amount of delay on distance measurement. Other researchers suggested a measurement system with a camera and two fixed plane mirrors. Stereo images reflected from the two mirrors are acquired by the single camera. With two fixed mirrors, the field of view is reduced and becomes narrower. Convex mirrors had been suggested to replace plane mirrors to increase the viewable and measurable area. However, image distortion caused by the convex mirrors becomes a major problem. Robot vision system and image processing is described details with illustration in chapter 4, sub-section 4.2.1.3.

5.2 Sonar

Sound waves are everywhere around us, even when we cannot hear them. Human hearing responds to sound frequencies in the range between 20 Hz and 20,000 Hz. It is important to make sure we first understand how to describe sound waves. The frequency of a wave is defined as the number of cycles the wave completes in a unit of time. More specifically, frequency of 1Hz, or one hertz, indicates that the wave oscillates one cycle over a time period of 1 second. Look at what happens to a sine wave when its frequency is increased from 1Hz to 5Hz. The sine wave on the left completes 1 full cycle within 1 second or, in other words, has a frequency of 1 Hz. The wave on the right oscillates 5 times in 1 second time or has a frequency of 5 Hz. For humans, the impact of the frequency limits means our ears cannot process sounds that complete less than 20, or more than 20000, oscillations per second [23].

Ships scanning the ocean floor for sunken submarines, planes or wrecks send ultrasonic pings, or ultrasound waves, that reflect off the surfaces and return back to the sensor.

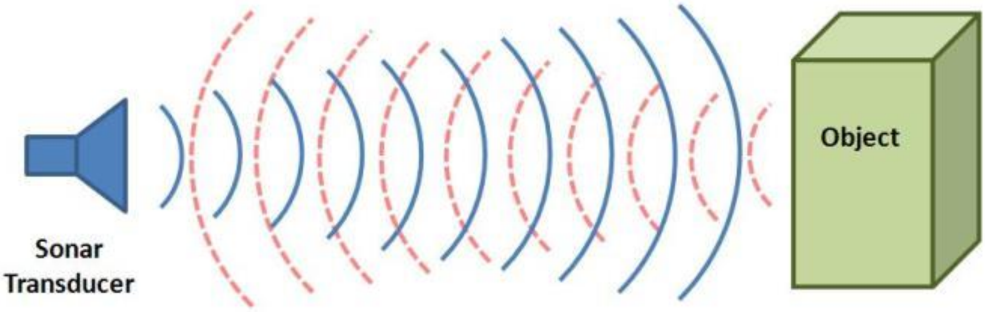


Figure 5.1 Basic principle of sonar sensor

Working principal:

The speed of sound in air varies as a function of temperature by the relationship:

$$c = 13044\sqrt{1 + \frac{T}{273}}$$

The wavelength of sound changes as a function of both the speed of sound and the frequency, as shown by the expression:

$$\lambda = c/f$$

Where:

λ = wave length, c= sound of wave and f= frequency

5.3 IR (Infrared Ray)

INFRARED (IR) sensors are extensively used for measuring distances. Therefore, they can be used in robotics for obstacle avoidance. They are cheaper in cost and faster in response time than US. However, they have non-linear characteristics and they depend on the reflectance properties of the object surfaces. So knowledge of the surface properties must be known prior. In other words, the nature in which a surface scatters, reflects, and absorbs infrared energy is needed to interpret the sensor output as distance measure. IR sensors using reflected light intensity to estimate the distance from an object are reported in the bibliography. Their inherently fast response is attractive for enhancing the real-time response of a mobile robot. Some IR sensors described in the bibliography are based on the measurement of the phase shift, and offer medium resolution from 5 cm to 10 m, but these are very expensive.

In an unknown environment, it is important to know about the nature of surface properties in order to interpret IR sensor output as a distance measurement. Here, US sensor can play an important role in determining the surface properties. The co-operation between the US and IR sensors are utilized to create a complementary system that is able to give reliable distance measurement. They can be used together where the advantages of one compensate for the disadvantages of the other. The integration of the information supplied by the multiple US and IR sensors can be a means to cope with the spatial uncertainty of unknown, unstructured environments in several applications of advanced robotics, such as flexible industrial automation, service robotics, and autonomous mobility.

METHODOLOGY

The process of measuring distance to an obstacle by using IR sensors can be divided into three steps [24]. First, the properties of the surface of the obstacle are determined. Secondly, the angle or orientation of the surface relative to the sensor is determined. Finally, the distance is calculated by using the information obtained in first two steps.

A. Determination of Surface Properties:

As light energy hits a surface, some portion of it scattered or absorbed and rest of the energy is reflected. Different surfaces scatter, absorb and reflect light in different portions. It is obvious that black surface will absorb more light than a white surface, and a shiny smooth surface will reflect more energy than a rough surface.

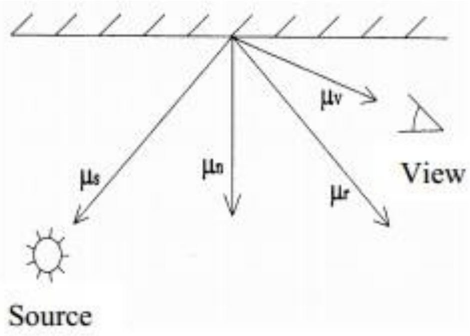


Figure 5.2 Phong Model

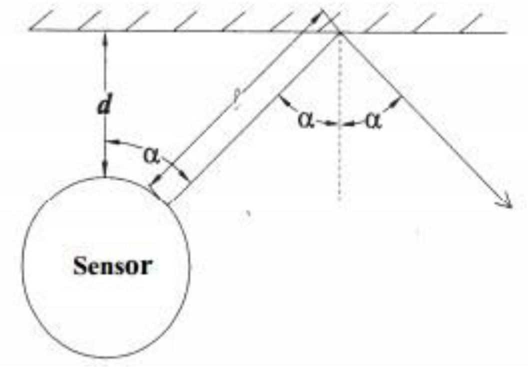


Figure 5.3 Emission and reflection of an infrared signal by sensor

.

The Phong Model can provide a simplified description of these effects into four constants: C0, C1, C2, and n. The Phong equation for intensity of energy, I, reflected from a surface is:

$$I = C_0(\mu_s.\mu_n) + C_1(\mu_r.\mu_v)^n + C_2$$

Where, μs, μn, μr and μv are the light source, surface normal, reflected and viewing vector, respectively.

The angle between the source vector and the normal vector of the surface is α. Also, if one assumes that the emitter and receiver are in the same position, then the angle between the viewing vector and the reflected vector is 2α.

Therefore, Equation (1) becomes:

$$I = C_0 \cos(\alpha) + C_1 \cos^n(2\alpha) + C_2$$

Again, the energy absorbed by the phototransistors is a function of Intensity (I), distance traveled (2l), and the area (A) of the sensor.

$$E = \frac{IA}{(2l)^2}$$

l Can be expressed in terms of d, a and the radius of the sensor (r):

$$l = \frac{d}{\cos(\alpha)} + r(\frac{1}{\cos(\alpha)} - 1)$$

Finally, the energy absorbed by the sensor can be expressed as:

$$E = \frac{C_0 \cos(\alpha) + C_1 \cos(2\alpha)}{[\frac{d}{\cos(\alpha)} + r(\frac{1}{\cos(\alpha)} - 1)]}$$

B. Determination of the Angle of a Surface

The relative angle of the sensor to the surface must be determined to simplify the calculating the surface properties and the distance of an obstacle. The maximum reading of the sensor will occur at α= 0. In Fig. 6, the spike occurs where the direction of the IR signal corresponds to the surface normal (α = 0).

C. Calculating the Distance to an Object

After obtaining the properties of a surface and the relative angle of the surface, it becomes easier to calculate the distance. From (5), the distance (d) can be expressed as:

$$d = r(\cos(\alpha) - 1) + \cos(\alpha)\sqrt{\frac{C_0 \cos(\alpha) + C_1 \cos(2\alpha)}{E}}$$

Thus the infrared readings can be interpreted to distance between the obstacle and the sensor.

CHAPTER 6

Experimental Procedure and Methodology Assessment

Chapter 6: Experimental Procedure for Methodology Assessment

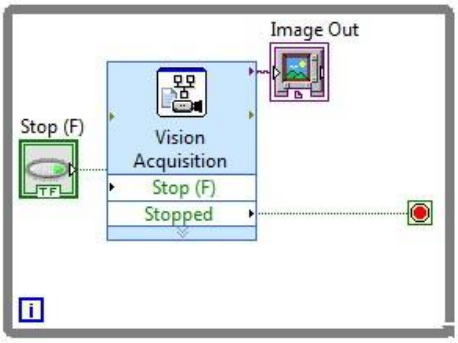
6.1 LabVIEW vision procedure

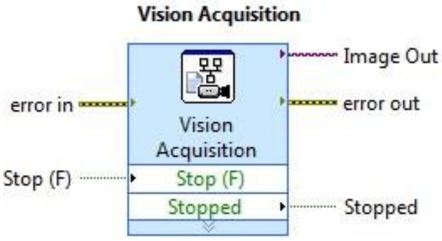
LabVIEW Vision is a powerful tool for image processing. It is used for Industrial, laboratory as well as education purposes. The main benefit of LabVIEW Vision is time saving because it is a wire programming. There is no need of typing. If anyone know the main mechanism of LabVIEW, he or she can complete the programming very quickly.

1. Vision Acquisition:

Creates, and edits acquisitions using the NI Vision Acquisition Express VI. The NI Vision Acquisition Wizard is launched by placing the Express VI on the block diagram. Select an acquisition source and configure an acquisition using NI-IMAQ, NI-IMAQdx, or simulate an acquisition by reading an AVI or image files from a folder. After an acquisition is configured, select controls and indicators to be able to programmatically set in LabVlEW. Double-click the Vision Acquisition Express VI to edit the acquisition.

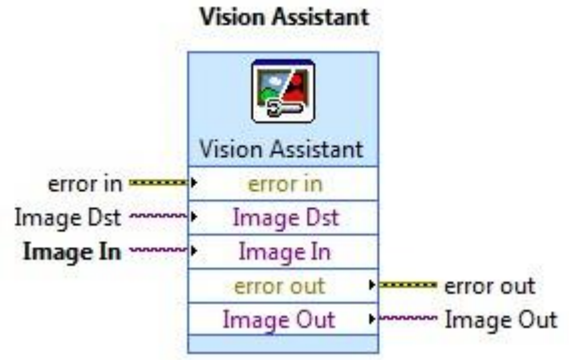
Note: Any images created by the Express VI need to be disposed after use. Use the IMAQ Dispose.vi to clean up the images output by the Express VI when they are no longer needed.

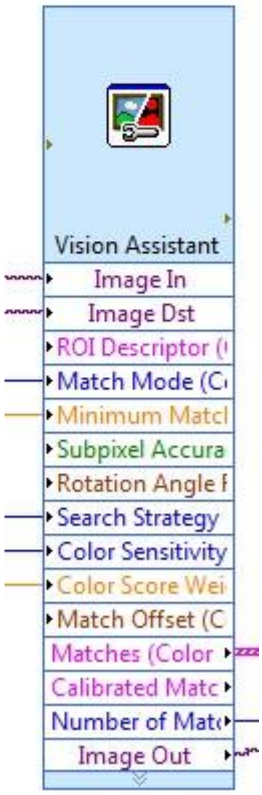




2. Vision Assistance:

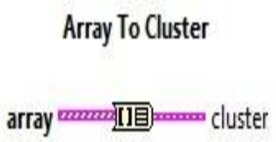
Creates, edits, and runs vision algorithms using NI Vision Assistant. When you place this Express VI on the block diagram, NI Vision Assistant launches. Create an algorithm using the Vision Assistant processing functions. After you create an algorithm, you can select the controls and indicators that you want to be able to programmatically set in LabVIEW. Double-click the Vision Assistant Express VI to edit the algorithm.





When various functions or matching of Image Processing is activated, Vision Assistance input and output configuration increase like Figure 2(a) to 2(b)

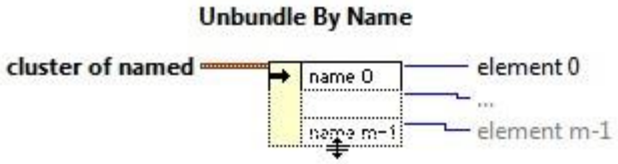
a. Array to Cluster:



Converts a 1D array to a cluster of elements of the same type as the array elements. Right-click the function and select Cluster Size from the shortcut menu to set the number of elements in the cluster.

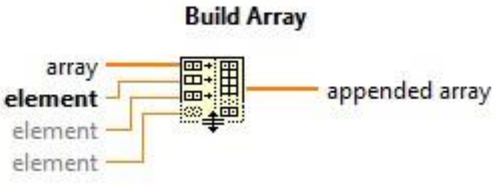
b. Unbundle by Name:

Returns the cluster element whose names are specified.



3. Build Array:

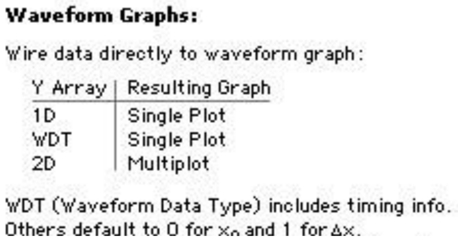
Concatenates multiple arrays or appends elements to an n-dimensional array.



4. Waveform Graph:

Waveform Graph

Waveform Graph is used to make a graph or chart from input information. Waveform graphs mainly works based on array.

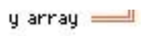


Waveform Graphs:

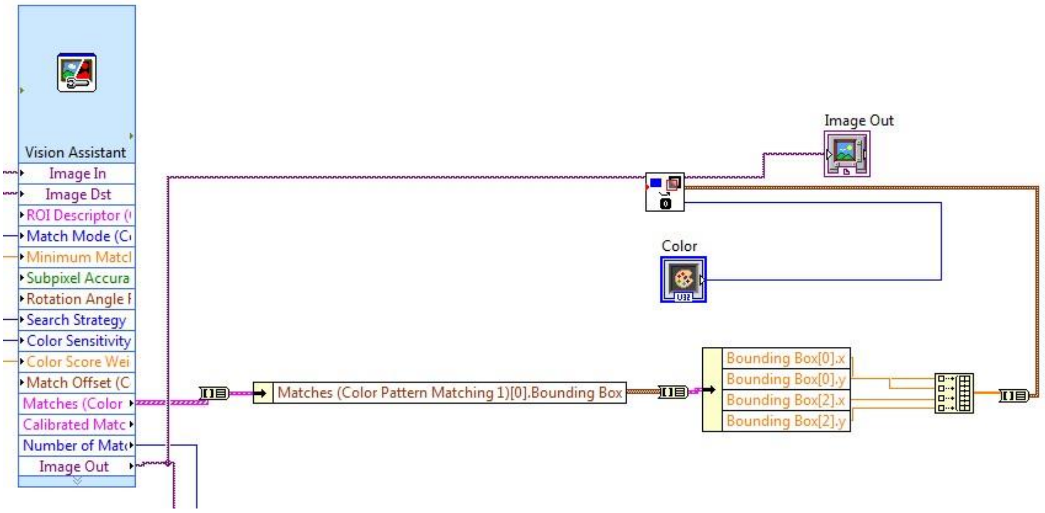
Wire data directly to waveform graph:

Y Array	Resulting Graph
1D	Single Plot
WDT	Single Plot
2D	Multiplot

WDT (Waveform Data Type) includes timing info. Others default to 0 for x_0 and 1 for Δx.



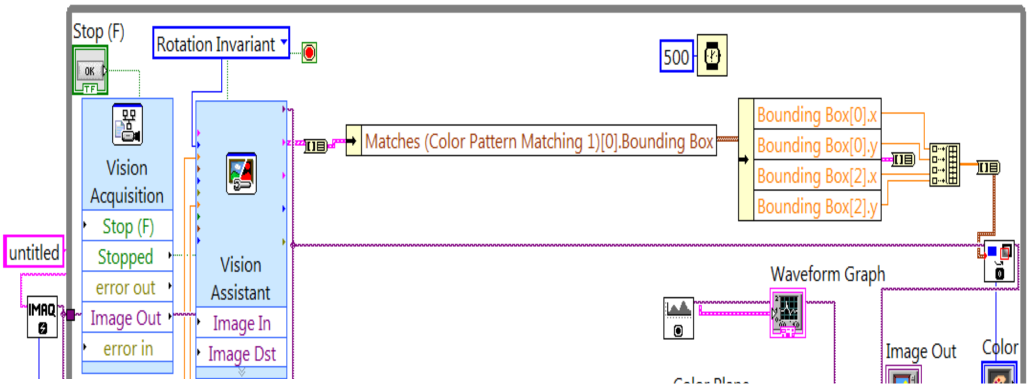
5. Image to Matching To display the matched images, first the input images are converted into arrays and then arrays to clusters. Overly



is applied on the Cluster to display the matching point. Then all the Clusters are combined together to make it again array. Thus the matching point has been shown.

6. Combination of whole program:

Whole program combines all the information coming from Vision Acquisition to Vision Assistance. Then it converts the image to appropriate data to apply Overlay and Matching on it. Again, the program combine all the data to present the result as an real-time image.



Experimental procedure for methodology assessment

Power calculation

6.2 Power calculation

Required Given;

In this project, 7 adapters are used. Each adapter provides 2 amp and 5V. So, total power supplied is –

P = (Number of motors*Voltage *Current) that means P= 7*2*5 = 70W

In this project 17 motors are used. Current drawn by each motor depends on the load. Load is maximum on the leg. Normally, 500mA per motor is enough. But, as the le motors carry a lot of load, it will carry more current too. For this reason, 1A per motor is given for the leg motors. About 10 motors are used for leg motion. So total power absorb by the leg is, P = 10 *5*1 =50W

As other motors draw only 500mA per motor, total power absorbed by the top side is P = 7*5*0.5 = 17W. So, about 3W is for extra storage. Arduino and sensors need some power which is given from there.

CHAPTER 7

EXPERIMENTAL RESULTS AND OBSERVATION

7.1 Vision analysis from LabVIEW

7.2 Histogram analysis

7.3 Accelerometer and gyro results

7.4 Integrated control system (ICS)

7.5 Walking mechanism

Chapter 7: Experimental Results and Observation

7.1 Vision analysis from LabVIEW

7.1.1 Color Matching:

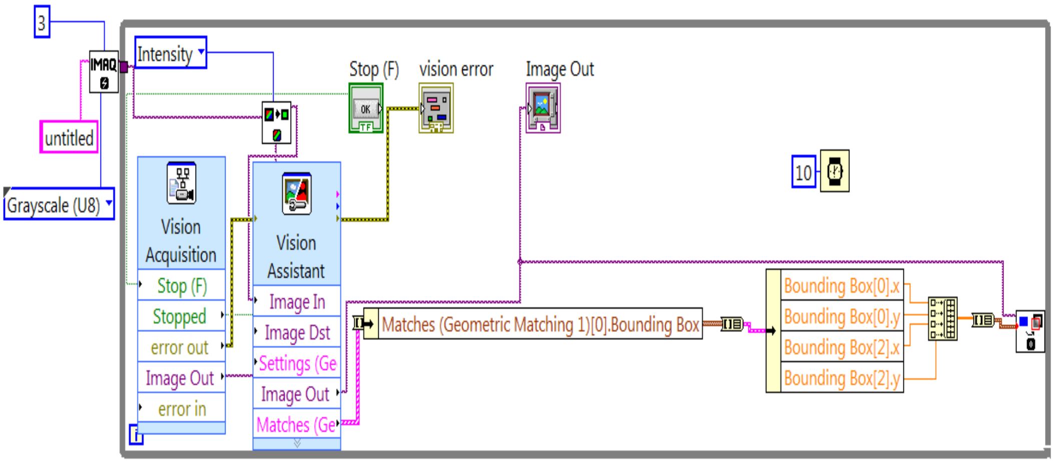
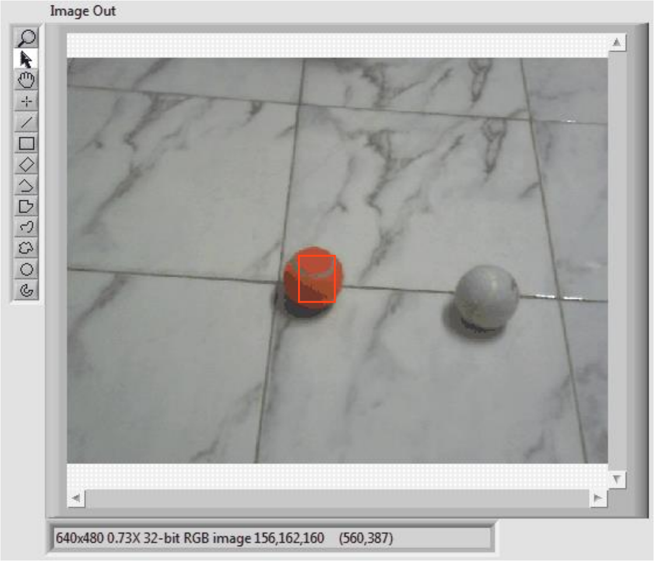


Figure 7.1 Color matching algorithm

Accuracy of color matching depends on target matching score and difference between the color threshold values of background and desired object.



640x480 0.73X 32-bit RGB image 156,162,160 (560,387)

Figure 7.2 NI Color Matching can detect the desired red object.

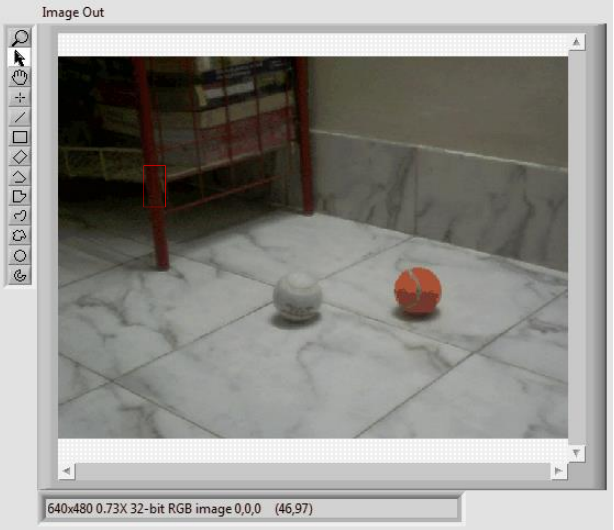


Figure 7.3 NI Color Matching wrong detection.

In Figure 7.2, the robot can easily detect the red ball without any interference. But the problem arises when a little bit similarity occurs in color matching, even a small amount of background color may affect the decision like in Figure 7.3.

7.1.2 Pattern matching

NI Vision pattern matching accurately locates objects in conditions where they vary in size (±5%) and orientation (between 0° and 360°) and when their appearance is degraded. Pattern matching is a fast and affective technique for object detecting. But pattern matching works only on gray scale images.

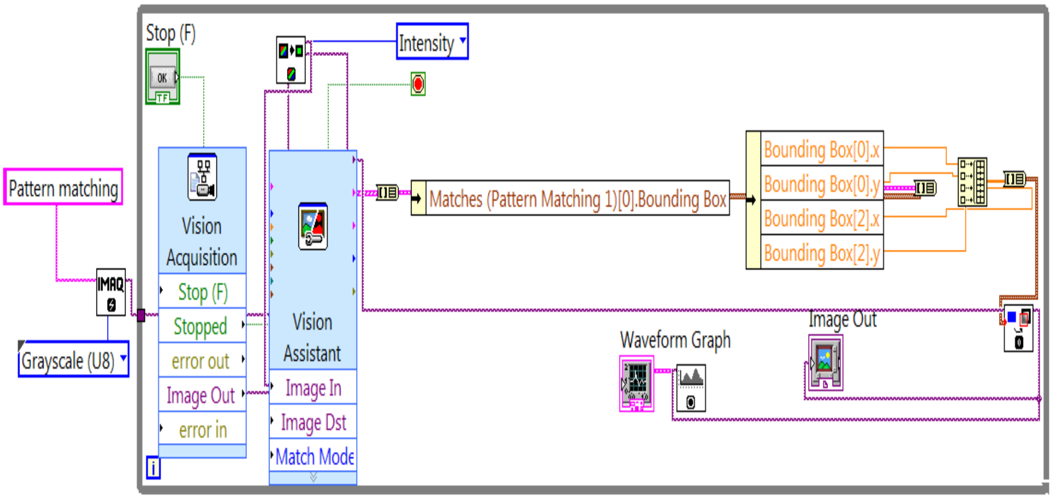
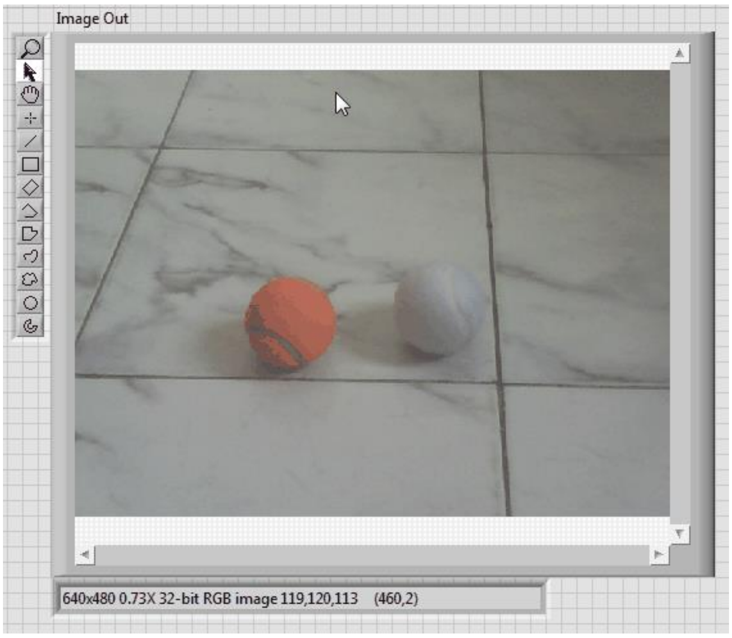
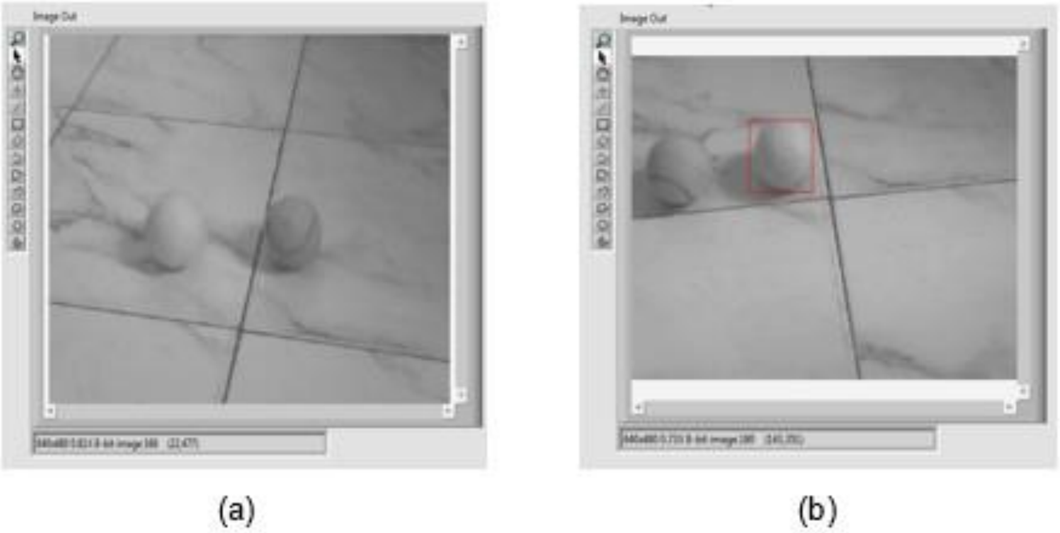


Figure 7.4 Pattern matching algorithm.



640x480 0.73X 32-bit RGB image 119,120,113 (460,2)

Figure 7.5 Original real time RGB image from NI Image Processing.



(a)

(b)

Figure 7.6 Gray scale image.

As the gray scale image lost its color properties, so the pattern matching cannot identify the deviations in color. As a result, two problem occurs. First one is, due to the gray scale affect if the background and the object become similar, it is difficult for the vision system to identify the object like Figure 7.6 (a). Second one is, pattern matching cannot differentiate among similar patterned different colored objects like in Figure 7.6 (b).

7.1.3 Color pattern matching:

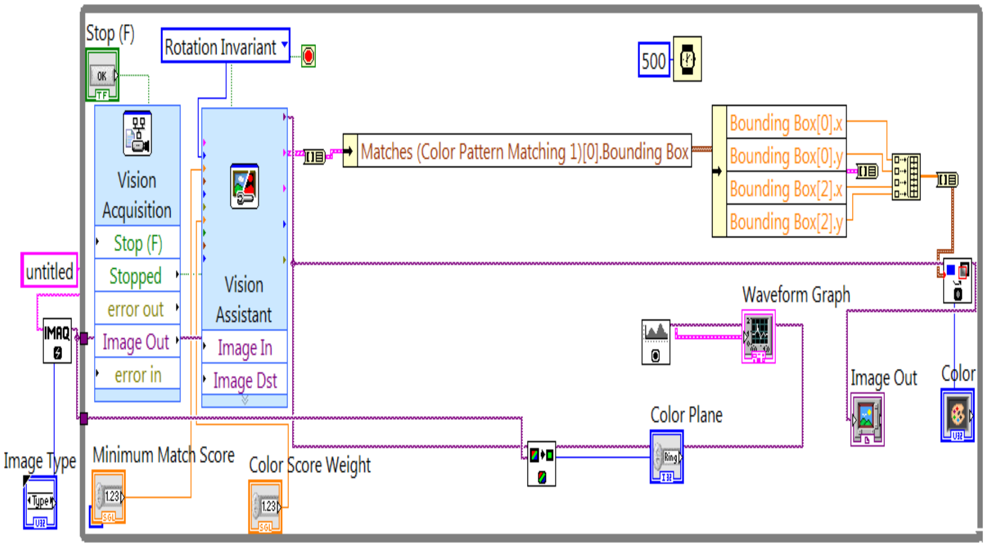
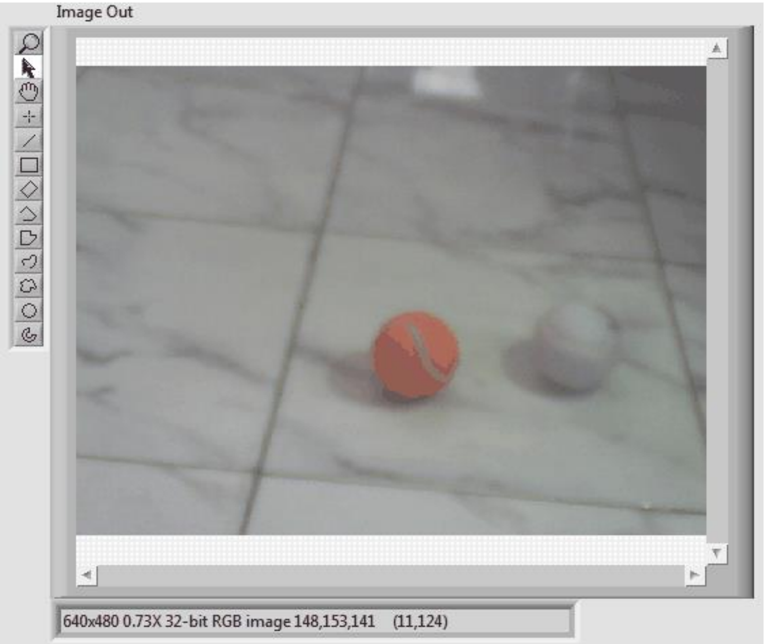


Figure 7.7 Color pattern matching algorithm



640x480 0.73X 32-bit RGB image 148,153,141 (11,124)

Figure 7.8 Original RGB real time image

In Figure 7.8, NI Color Pattern Matching cannot detect the red ball when the minimum match score is 750 and Color score weight is 500.As the color matching score weight is low and pattern matching cannot also give a good score on the white background.

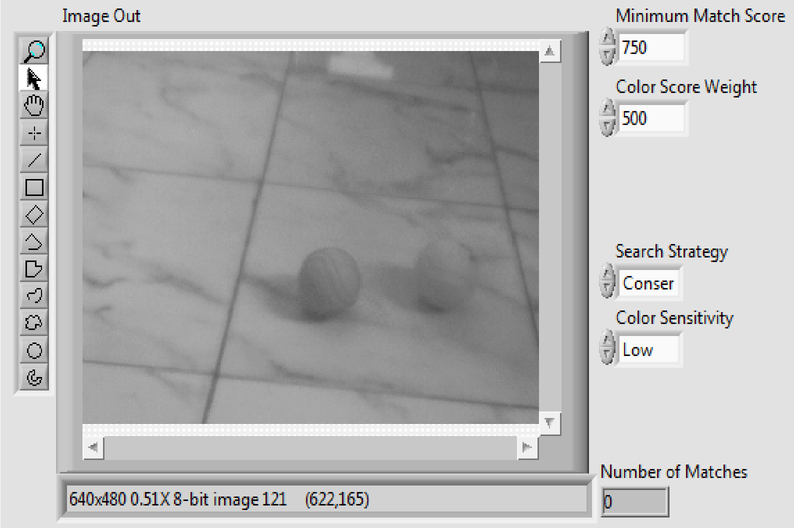


Image Out

Minimum Match Score 750

Color Score Weight 500

Search Strategy Conser

Color Sensitivity Low

Number of Matches 0

640x480 0.51X 8-bit image 121 (622,165)

Figure 7.9 NI Color Pattern Matching cannot detect the red ball.

Color Weight calibration is very important in Color Pattern matching because the matching is based on both the color matching score and the pattern matching score.

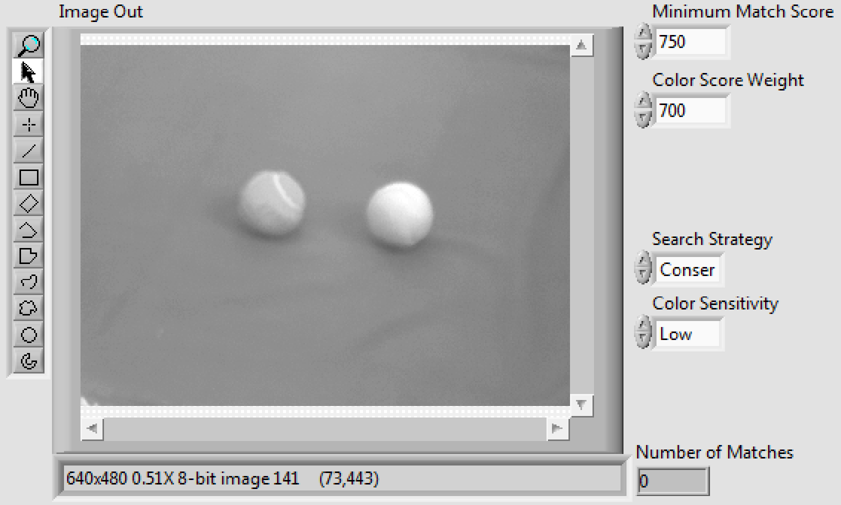


Image Out

Minimum Match Score 750

Color Score Weight 700

Search Strategy Conser

Color Sensitivity Low

Number of Matches 0

640x480 0.51X 8-bit image 141 (73,443)

Figure 7.10 Calibration for white background.

Color weight matching score has to be increased when the colors of ball and the background become similar after greyscale imaging because pattern matching works based on the greyscale image. Otherwise, it is not possible to track the ball as the pattern matching is giving very small amount of score like Figure 7.10.

Color Weight matching score has to be reduced when the object color and the background color match with each other. In this the total score remains below the Minimum Match Score .As a result NI vision cannot detect the desired ball like Figure 7.10. Calibration for white ground may not work on green or black background. As a result NI Color Pattern Matching cannot detect the desired object.

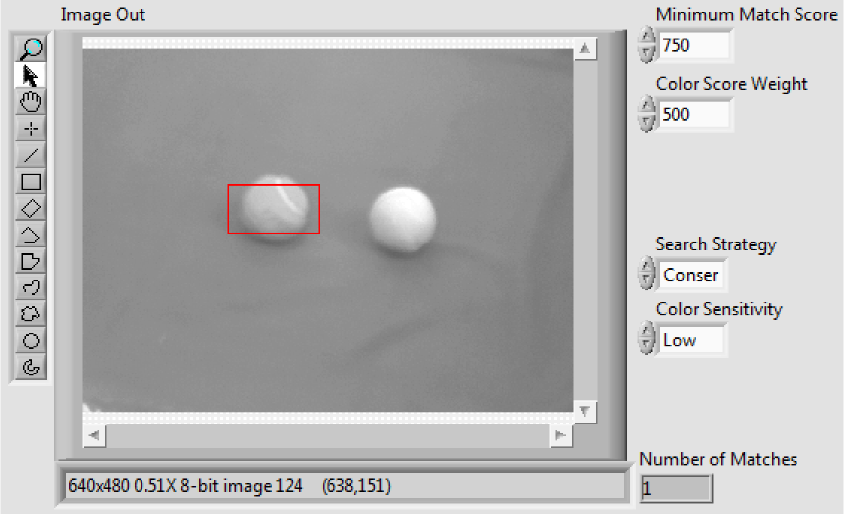


Image Out

Minimum Match Score
750

Color Score Weight
500

Search Strategy
Conser

Color Sensitivity
Low

Number of Matches
1

640x480 0.51X 8-bit image 124 (638,151)

Figure 7.11 NI Color pattern matching correct detection

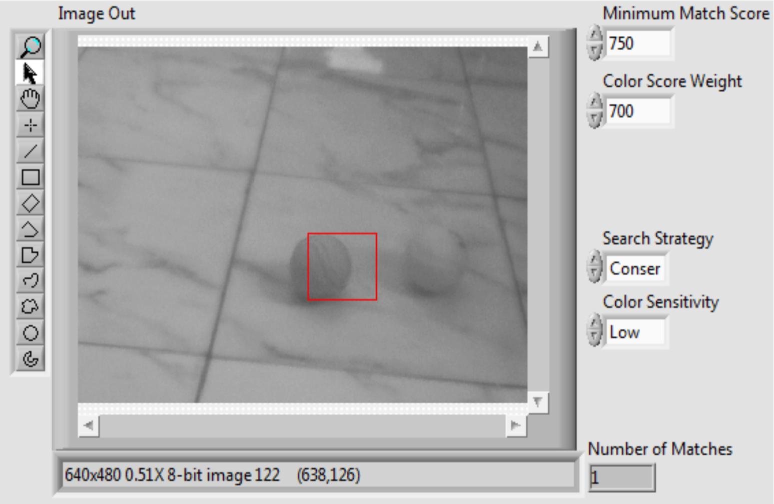


Image Out

Minimum Match Score
750

Color Score Weight
700

Search Strategy
Conser

Color Sensitivity
Low

Number of Matches
1

640x480 0.51X 8-bit image 122 (638,126)

Figure 7.12 Correct detection of red ball

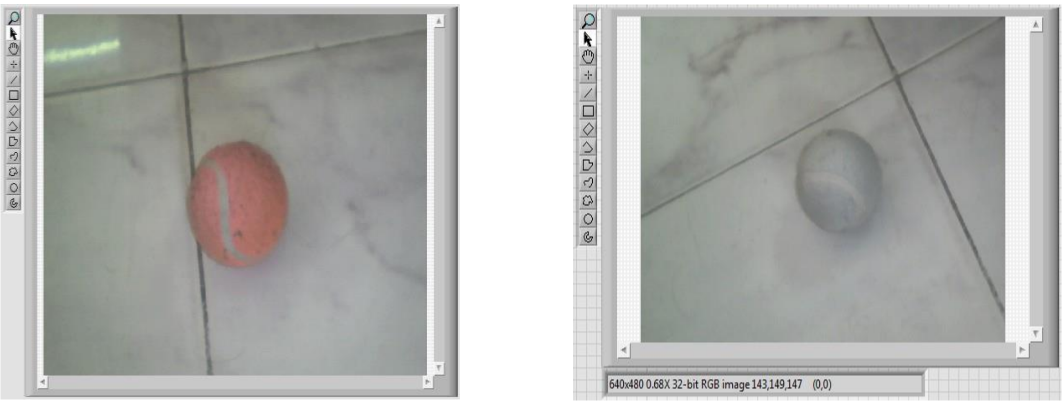
As the Color Score Weight is reduced, again the vision system start to track the ball like in Figure 7.11.

In Figure 7.12, NI Color Pattern Matching cannot detect the red ball when the minimum match score is 750 and Color score weight is 500.As the color matching score weight is low and pattern matching cannot also give a good score on the white background , total received matching score remains below 750.

7.2 Histogram

Histogram is a powerful tool to represent whether the image is suitable for analysis or not.Another benefit is that the pixels intensity of particular color can be easily understood. These informations are used for MISTBOY vision system to analyze the image and to detect the red ball more accurately.

Though Red ball has already been detect by color pattern matching algorithm,it's accuracy increases by combining it with histogram.



(a)

(b)

Figure 7.13 Image of (a) Red ball, (b) White ball.

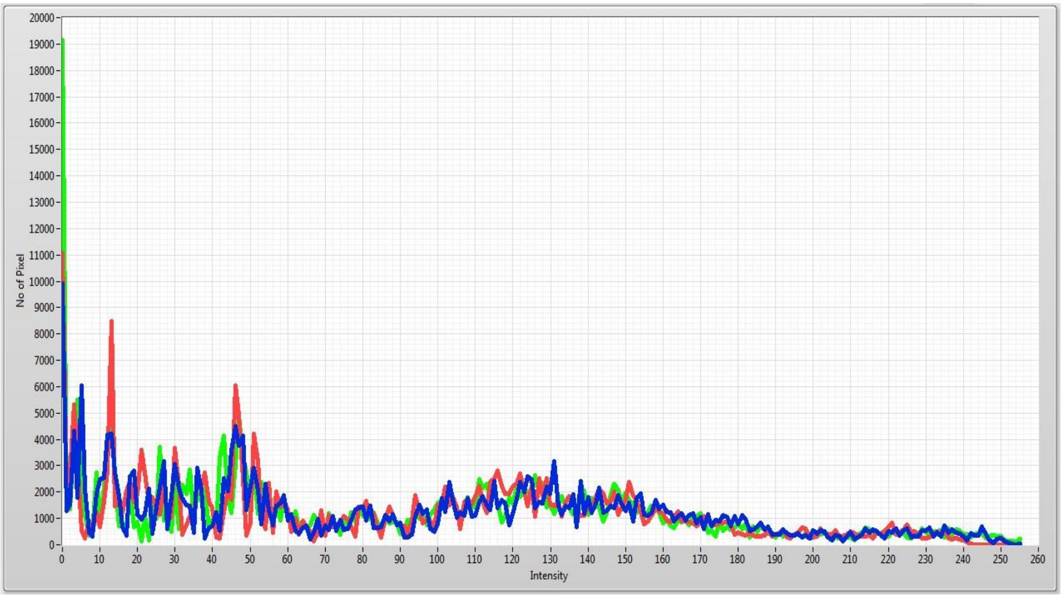


Figure 7.14 Histogram of white ball where intensity is decreased.

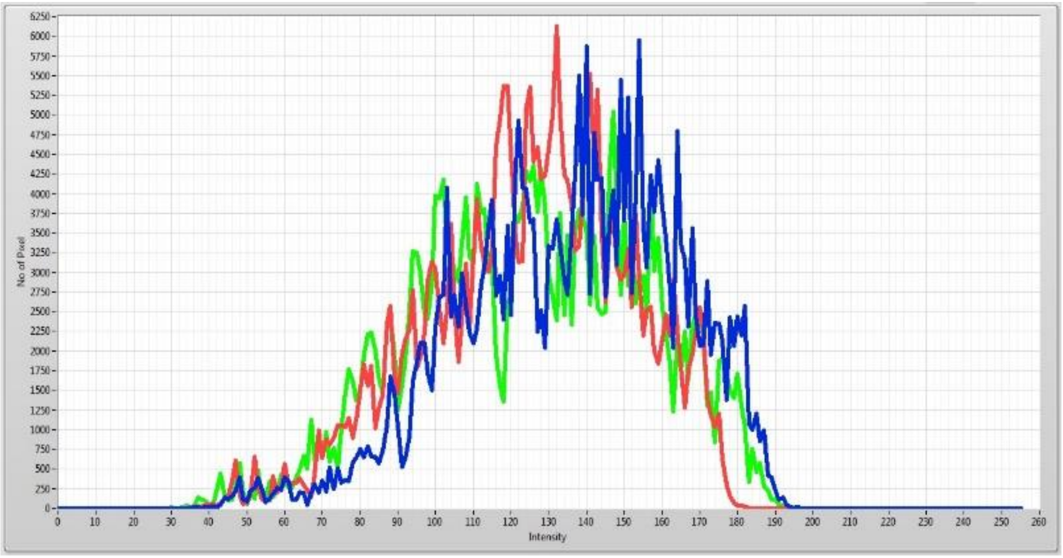


Figure 7.15 Histogram of red ball where intensity is increased.

In Figure 7.15 and 7.13 (a), when the red ball appears infront of the camera ,the intensisty as well as the red pixel value increses.On the other hand, in Figure 7.14 and 7.13 (b), when the white ball or white things appears ,the pixel values beome very low.By observing the histogram MISTBOY vision system can easily detects the color of the ball and makes the color pattern matching algorithm more accurate. This technique can be used to detect other colors also.

7.3 Accelerometer and Gyro results

IMU MPU 6050 is used in MISTBOY which is the combination of accelerometer, gyroscope. Combination of accelerometer and gyro, gives MISTBoy a better stability. Without filter, accelerometer and gyro works independently. To combine the data, complementary filter is used.

Before Filter

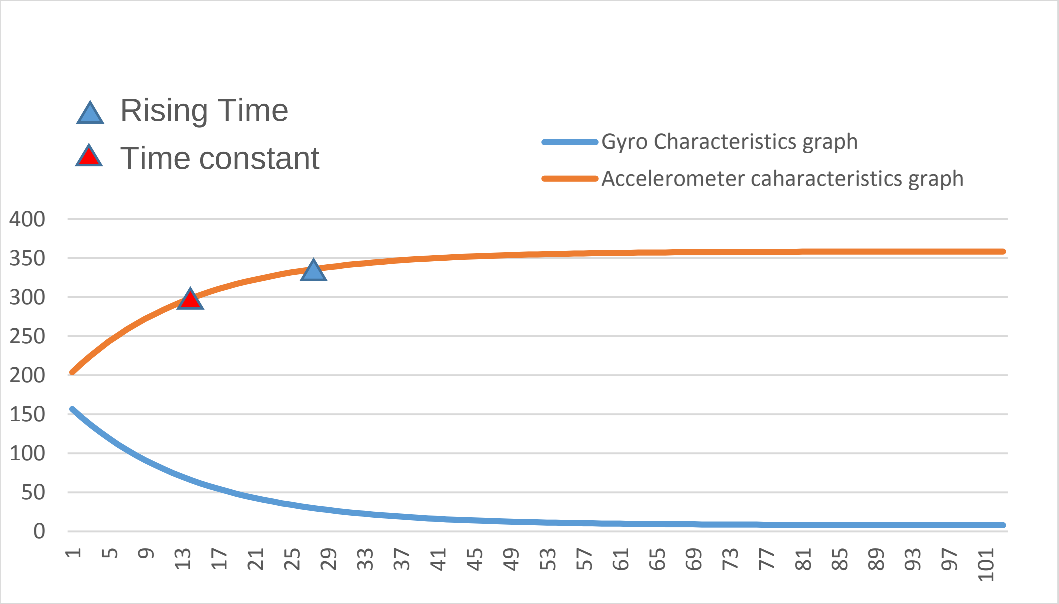


Figure 7.16 Graph in stable condition.

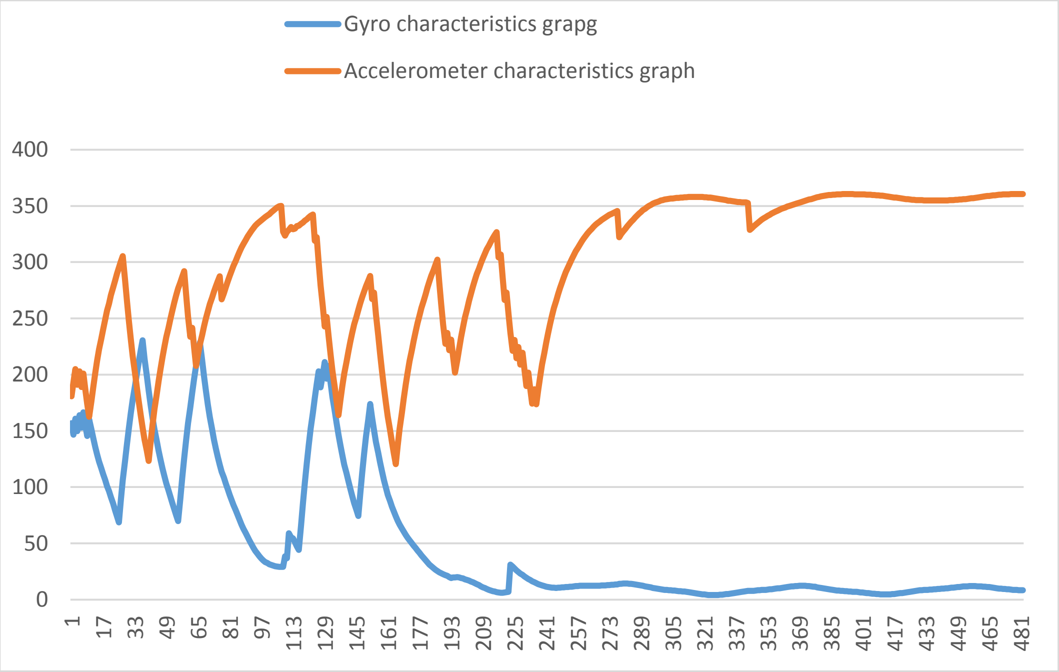


Figure 7.17 Graph in vibrating condition.

From the Figure 7.16, it is the characteristics graph of MPU 6050 when MISTBoy is in stable condition. It is seen that the sensor takes only a few seconds to give stable data. Gyro and accelerometer data become stable at same moment. When MISTBoy is in unstable or vibrating conditions, MPU 6050 give another characteristics graph which is shown in Figure 7.17. From the characteristics graph it is seen that the deviations of data from the stable condition. This graph also shows that the sensor is taking a long time to give stability.

After Filter

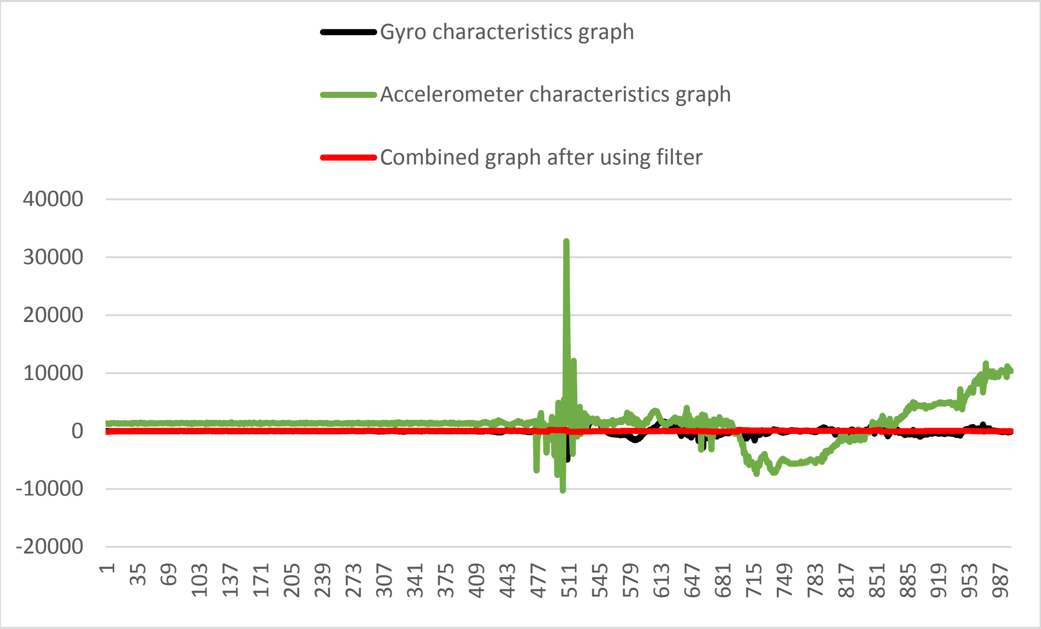


Figure 7.18 Characteristics graph after filtering.

Though the accelerometer and gyro data are deviating from the standard condition, filter gives almost a straight line like in Figure 7.18. Complementary filter reduces the noise and pass the useful information from the sensor.

7.4 Integrated Control system (ICS)

ICS increases the stability of a system.It combines the acquired data from various sources and feedback techniques. The most powerful feedback technique in a biped robot control system is the combination of Gyro-Acelerometer,IR and Sonar sensors data with the vision system.

Accelerometer and Gyro acknowledge the position of the body.These data are helpful to maintain the center of gravity (COG) between the two feet to keep the Zero Moment Point (ZMP) below the feet surface area.

Sonar and IR help to detect and measure distance among the obstacles like own team players,opponent team players.

Arduino and LabVIEW are considered as the brain of the robot.Vision data are analyzed in LabVIEW using the laptop processor. Camera data can be sent there by using the serial or WiFi communication.LabVIEW detects the Game field,Field Marekrs,D-box,Goal post, same and oppsite team players

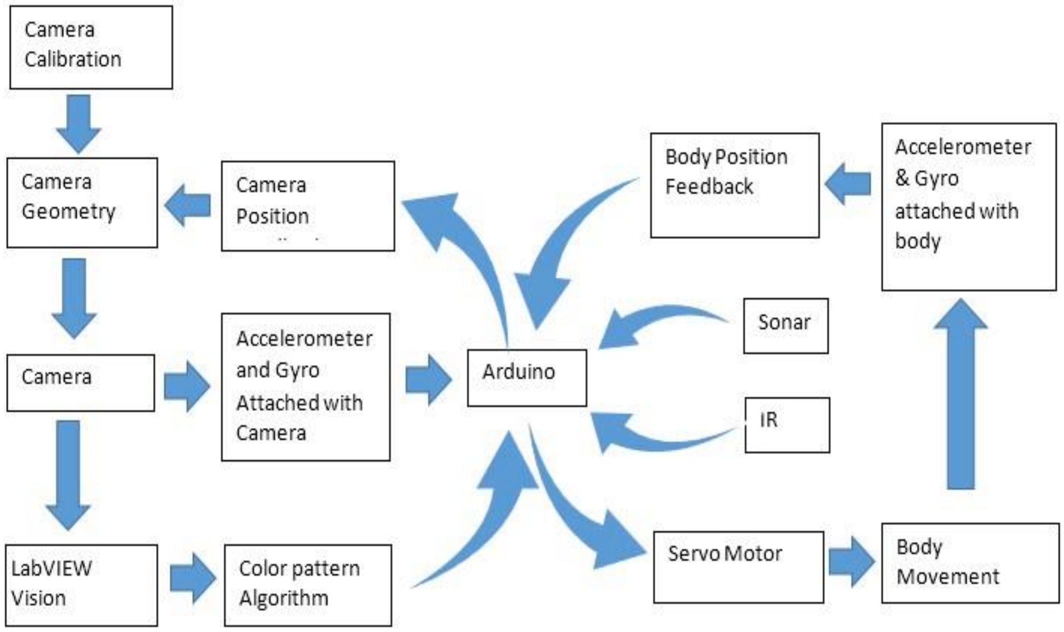
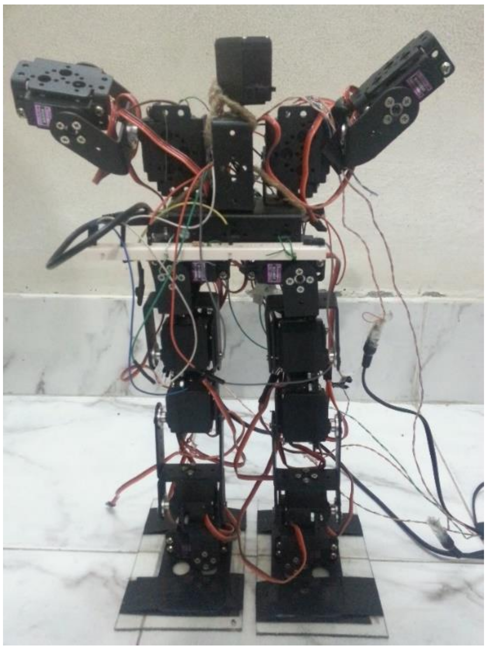


Figure 7.19 Integrated Control System of MISTBOY.

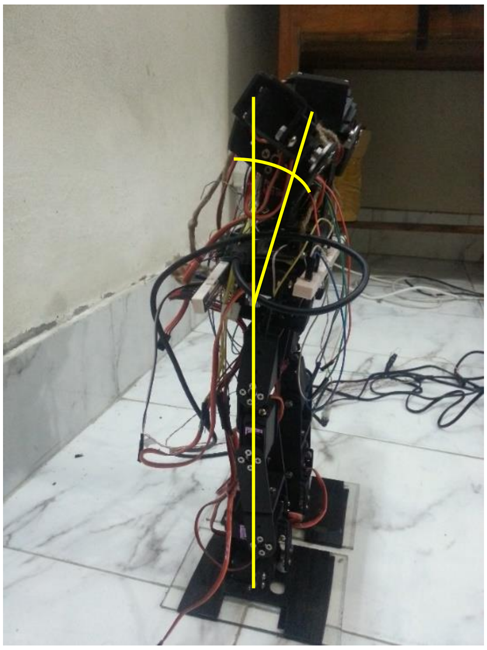
and the ball.Arduino controls all the servo motors according to the vision and other sensing system feedback.

Only a powerful vision system can eliminates the drawback or noises of other feedbacks.One of the most challenging step is to control the accuracy of the position of the motors according to the vision system.Using Vision System,MISTBOY can just detect the desired objects.But,the motors have to be mapped with the vision system, to lead the robot towards the desired position. The image resoulution is 640*480.That means the whole image is devided into 640 pixels towards the X axis and 480 pixels towards the Y axis.On the other hand, a typical servo motor can rotate from 0^0 to 180^0. So,if the matching occurs at 320 pixels the camera will rotate about 90^0 .This mapping is being processsed in Arduino. The camera is attached with a servo motor and the servo is getting the the order from Arduino.Arduino takes deision according to the matching happened result.

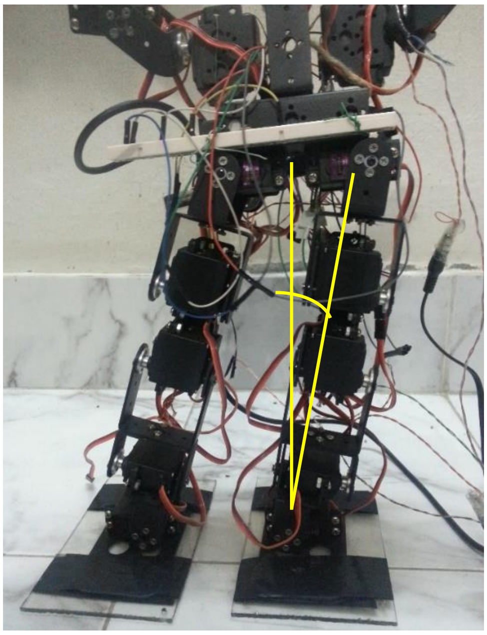
7.5 Walking mechanism



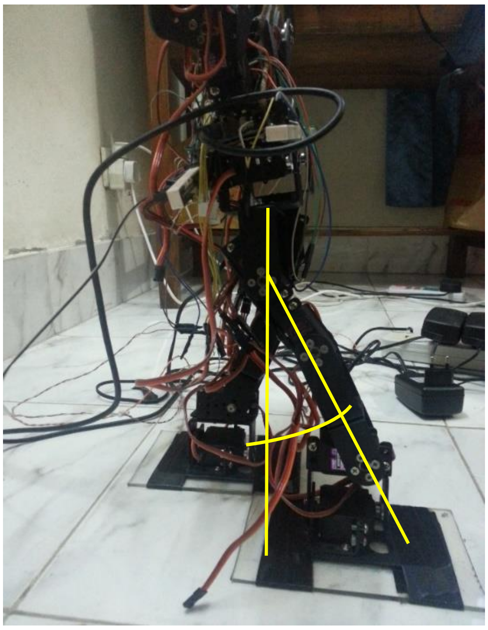
(a) Stable position.



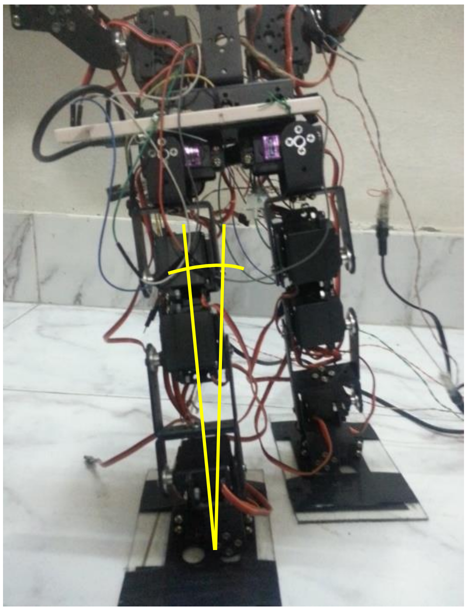
(b) Bow down 15^0



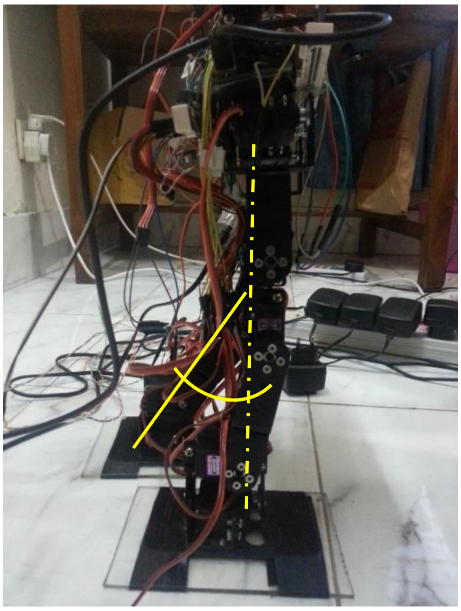
(c) Bending of left leg by 15^0



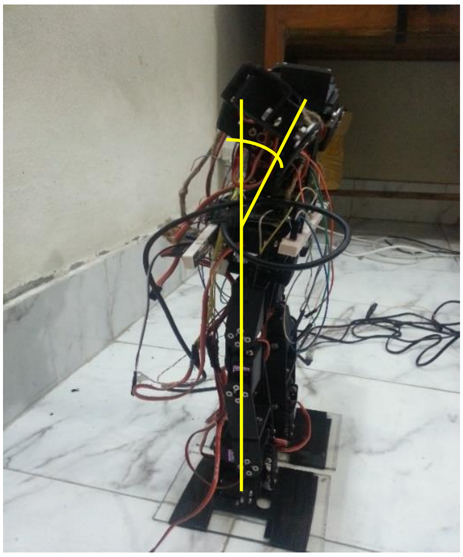
(d) Forward motion by 30^0



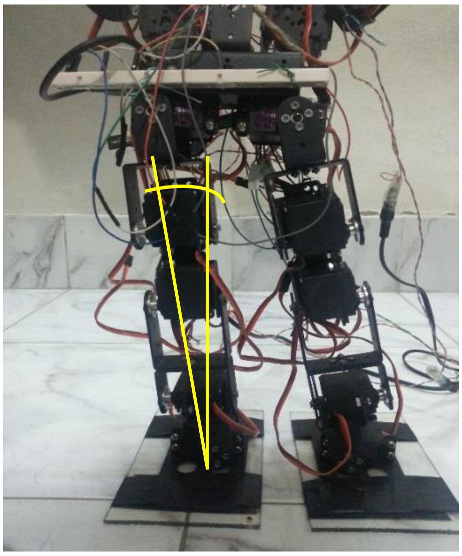
(e) Bending of right leg by 15^0



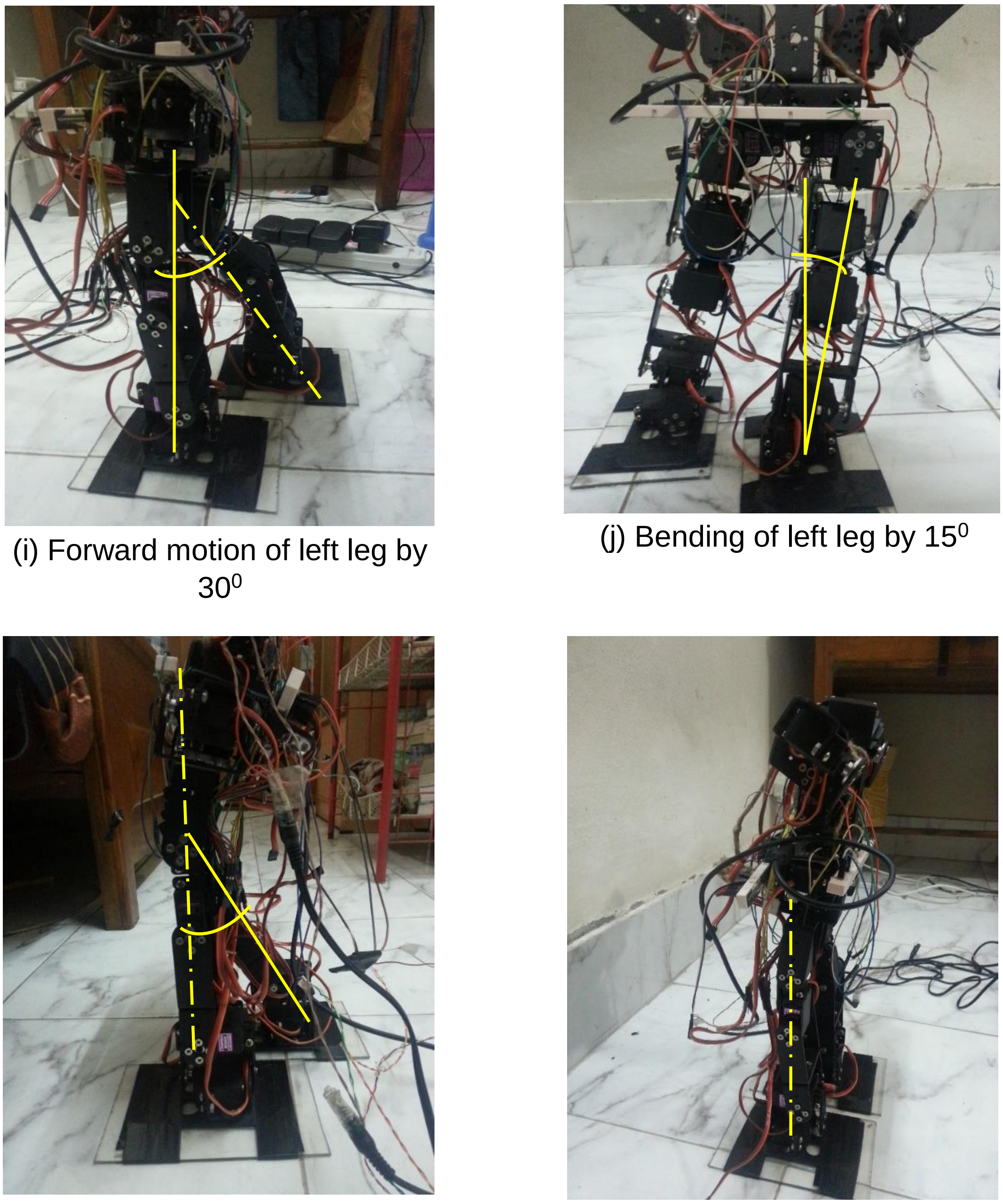
(f) Right leg stable from bending



(g) Bow down of head by 15^0



(h) Bending right leg by 15^0



(i) Forward motion of left leg by 30^0

(j) Bending of left leg by 15^0

(k) Right leg came forward

(l) Both leg became stable

Figure 7.20 Walking mechanism.

CHAPTER 8

Conclusion and Recommendations

8.1 Conclusion

8.2 Recommendations

8.3 Future work

Chapter 8: Conclusion and Recommendations

8.1 Conclusion

A systematic approach is presented in this book to derive the parameters for the integrated control system of a humanoid robot, best on its ability to sense and take decission. At first, the camera geometry and its calibration to integrate it with LabVIEW and Arduino is discussed. The matching techniques and algorithms, used in the vision system, are discussed and real time images are analyzed to find out the ball in various critical condition to find out the best matching technique for humanoid robot. Sensors characteristics graph before and after applying filters are also discussed at various stable and vibrating condition to experiment and analyze the technique to get the best suitable value from the sensor. Therefore, a relation among the electrical sensors,vision and actuators is also established to make the MISTBOY autonomous. This discussions are very useful for designing and increasing the accuracy of vision and control system of a humanoid robot.

8.2 Recommendations

To make a robot behave like human is a very complex task. To make it possible, it is compulsory to combine the theory and practical. First of all static and dynamic balance are necessary to make the robot capable of smooth walking. There are some robotics theory like D-H parameters and inverse kinematics to analyze this motion though it is tough job because the matrices have to be use in C programming. Researchers and professionals use this theories in industrial level. So, it is recommended to apply those theories in programming.

Second recommendation about the project is to use high torque motor. To hold the robot, the motors need to be very powerful. The motor torque should be more than 20 kg-cm. In this project Modified TowerPro MG996R is used whose torque is 16 kg-cm. The best and safe way is to use Dynamixel motor because it is precise and powerful though it is very costly.

Third recommendation is about the power. 1 amp 6V per motor should be used. Otherwise, motor will not give its maximum torque. Numbers of adapter may be used in parallel. It will be great if Li-Po battery can be used.

Next recommendation is about Sensor. In this project, Complimentary filter is used to remove the noise from the values of MPU 6050. There are also some other filter like Kalman Filter. It is recommended to try other filters to find out which is suitable for balancing of a humanoid robot.

Final recommendation is about Vision System. LabVIEW is a powerful tool for image processing as well as for Robotics projects. There are a lot of facilities for image processing in LabVIEW. There was not enough time to go to deep of LabVIEW image processing. A lot of filters and techniques are given in LabVIEW which are recommended. There are some problem in LabVIEW and Arduino serial communication. The problem should be found out. An open source and popular vision software is OpenCV (Open Source Computational Vision) .It is also recommended for them who wants to work with this project in future.

8.3 Future work

The design and function of MISTBoy can be simulated to analyse the dynamic motion of the system and thereby the design can be optimized. Then the artificial intelligence and the vision system can be introduced to give MISTBoy the intelligence of a soccer player.

Bibliography

1. http://www.preservearticles.com/201107158722/robots-and-robotics.html. Visited on 28-11-2013.
2. http://www.elprocus.com/robots-types-applications. 03-12-2013.
3. http://www.sciencekids.co.nz/sciencefacts/technology/historyofrobotics.html. Visited on 03-12-2013
4. http://www.galileo.org/robotics/intro.html. Visited on 06-12-2013.
5. J. Yamaguchi, A. Takanishi and I. Kato: "Development of a bipedal walking robot compensating for three-axis moment by trunk motion", Proc. IEEE RSJ Int. Conf. on Intelligent Robots and Systems, (1993), Pages 561 – 566.
6. Y. Sakagami, R. Watenabe, C. Aoyama, S. Matsunaga, N. Higaki and K. Fujimura: "The intelligent ASIMO : System overview and integration", Proc. IEEE RSJ Int. Conf. on Intelligent Robots and Systems, (2002), Pages 2478 – 2483.
7. http://global.kawada.jp/mechatronics/hrp4.html. Visited on 07-12-2013
8. E. Hashemi, M.G. Jadid, M. Lashgarian, M. Yaghobi, M Shafiei.: "Particle Filter Based Localization of the Nao Biped Robots", Proc. IEEE Southeastern symposium on System Theory, University of North Florida, Jacksonvile, FL, (2012).
9. W. Park, J-Y. Kim, J. Lee and J-H. Oh: "Mechanical Design of Humanoid Robot Platform KHR-3 (KAIST Humanoid Robot – 3 : HUBO)", Proc. IEEE – RAS Int. Conf. on Humanoid Robots, (2005), Pages 321 – 326.
10. Dr. Daniel D Lee,"Stability Control System for a Bioloid Humanoid Robot".

11. http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code =MMA745xL. Visited on 07-12-2013.

12. http://www.scribd.com/doc/42528503/Gyroscope. 08-12-2013.

13. http://www.scribd.com/doc/89663365/A-Comparison-of-Complementary-and-Kalman-Filtering. Visited on 08-12-2013.

14. Niofer Mehta, "Kalman filtering of sensor data".

15. Toshak Singhal, Akshat Harit, and D N Vishwakarma,"Kalman Filter Implementation on an Accelerometer sensor data for three state estimation ofa a dynamic system", International Journal of Research in Engineering and Technology Vol. 1, Nov 6,2012 ISSN 2277-4378, Pages 330-334.

16. Shane Colton,"A simple solution for integrating Accelerometer and gyroscope measurements for a Balancing Platform", Chief Delphi White paper,June, 2007.

17. Ms Shewta Tyagi, Mr. Hemant Amhia, Mr. Shivdutt Tyagi,"Comparative Study of Image Enhancement and Analysis of Therman Images Using Image Processing and Wavelet Techniques", International Journal of Computational Engineering Research, Vol. 03, April 2013, Pages 32-38.

18. http://dasl.mem.drexel.edu/alumni/bGreen/www.pages.drexel.edu/_weg22/edge.html. Visited on 09-12-2013.

19. http://www.tutorialspoint.com/dip/Sobel_operator.htm. 09-12-2013

20. Gaurav Sharma, Sonali Sood, Gurjot Sing Gaba and Nancy Gupta , "Image Recognition System using Geometry Matching and Contour Detection ," in Proc. Int. Journal of Computer Applications (0975-8887), vol. 51-No. 17, August 2012.

21. http://zone.ni.com/reference/en-X/help/372916L01/nivisionco ncepts /color_pattern_matching. Visited on 10-12-2013.

22. Christopher G. Relf, "Image Acquisition and Processing with LabVIEW," ISBN- 0-8493-1480-1.

23. http://www.teachengineering.org/view_activity.php?url=collection/nyu_/activities/nyu_soundwaves/nyu_soundwaves_activity1.xml. Visited on 11-12-2013.

24. G Benet, F. Blanes, J.E. Simo, P. Perez,“Using Infrared Sensors for Distance Measurement in Mobile Robots“, Robotics and Autonomous Systems 1006(2002), Pages 1-12.

Appendix – A

```
#include <Wire.h> //Include the Wire library
#include <MMA_7455.h> //Include the MMA_7455 library
#include <Servo.h>
#include "I2Cdev.h"
#include "MPU6050.h"

Servo waistServoR;
Servo thighServoR;
Servo kneeServoR;
Servo ankleServoR;
Servo feetServoR;

Servo waistServoL;
Servo thighServoL;
Servo kneeServoL;
Servo ankleServoL;
Servo feetServoL;
Servo headServo;

int RwaistPin =11;
int RthighPin = 10;
int RkneePin = 9;
int RanklePin = 8;
int RfeetPin = 7;


int LwaistPin = 6;
int LthighPin = 5;
int LkneePin = 12;
int LanklePin = 3;
int LfeetPin = 2;

int headPin =13;

int flag = 0;

MMA_7455 mySensor = MMA_7455(); //Make an instance of MMA_7455
//char xVal, yVal, zVal; //Variables for the values from the sensor
MPU6050 accelgyro;

int16_t ax, ay, az;
int16_t gx, gy, gz;
#define LED_PIN 13
bool blinkState = false;

//*****//KALMAN FILTER-START//
unsigned long timer;

double zeroValue[5] = {950, -400, 13500, -100, -500};

/* All the angles start at 180 degrees */
double gyroXangle = 180;
double gyroYangle = 180;

double compAngleX = 180;
double compAngleY = 180;
double comppAngleX, comppAngleY;

//******////KALMAN FILTER -FINISH

void setup()
                {    Wire.begin();
                     Serial.begin(38400);
```

```
      // MPU 6050 setup
      // initialize device

      Serial.println("Initializing I2C devices...");
      accelgyro.initialize();
      // verify connection
      Serial.println("Testing device connections...");
     Serial.println(accelgyro.testConnection() ? "MPU6050
connection successful" : "MPU6050 connection failed");

      // configure Arduino LED for
      pinMode(LED_PIN, OUTPUT);

     //*****////KALMAN FILTER-START

     timer = micros();

    // ****///KALMAN FILTER-FINISH

       //"Attach" the servo object
      rahidservo.attach(servoPin);
      waistServoR.attach(RwaistPin);
      thighServoR.attach(RthighPin);
      kneeServoR.attach(RkneePin);
      ankleServoR.attach(RanklePin);
      feetServoR.attach(RfeetPin);

      waistServoL.attach(LwaistPin);
      thighServoL.attach(LthighPin);
      kneeServoL.attach(LkneePin);
      ankleServoL.attach(LanklePin);
      feetServoL.attach(LfeetPin);
      headServo.attach(headPin);
```

```
                }

          //******   Variable declaration   *****
                      float s=0;
                       float xVal, yVal, zVal; //Variables for the values from the
sensor
                       float y_val[20],x_val[1000];
                       float avg_y ,avg_x;
                       float sum;

                        //Speed Control
                      float robot_speed = 20;
                      float pos_delay = 20;
                       //Waist bend
                  //Expres the angles by using  GLOBAL variables

         //******  Stable angles variables
                      float rws= 95; //95
                      float rts= 70; //70
                      float rks= 75; //75
                      float ras = 58; //55
                      float rfs= 80 ; //80

                      float lws = 120; //120
                      float lts= 90;  //90
                      float lks= 70 ;  //70
                      float las = 77;  //80
                      float lfs= 100 ;  //100

                      float wb=10;
                      float fb = 15;
                      float kb = 20;
                      float tb =40;
```

```
            //Forward angles variables
   //****** thigh angles

               float rtf = rts -tb;
               float rtb;
               float ltf;
               float ltb;

              //float ltf= ;

    //****** ankle angles
               float raf;
               float rab = ras + 20;
               float laf;
               float lab;

   //******* waist angles
               float lwl = lws - 20;
               float rwr = rws - 10;

  //******* feet angles
               float rfr= rfs + 10;
               float rfl = rfs - 20;
               float lfl= lfs - 10;
               //float lfl= ;

 //*******knee angles
               float rkf;
               float rkb= rks-kb;
               float lkf = lks - 10;
               //float lkb= ;
```

```
float stable () //1
               {
                   //STABLE CONDITION
                     lts= 90;
                     rts = 70;
                     ras = 58;
                     las = 77;
                  waistServoL.write(lws);
                  thighServoL.write(lts);   //110
                  kneeServoL.write(lks);     //95
                  ankleServoL.write(las);
                  feetServoL.write(lfs);     //90
                  waistServoR.write(rws);
                  thighServoR.write(rts);    //85
                  kneeServoR.write(rks);     //90
                  ankleServoR.write(ras);
                  feetServoR.write(rfs);     //75
                  Serial.print("STABLE");
                  delay(1000);
               }

void left_feet_bend(int lfl,float lwl)//2
               {
                    // waistServoL.write(lwl);
                     //delay(100);
                     feetServoL.write(lfl);
                     delay(100);
               }

void right_leg_forward() //3  //float rts,float rtf,float rks,float rkb,float rfs,float
rfr
               {
                         for(int l=rws; l> rwr; l--)
                  {
                   waistServoR.write(l);
                   float k = map(l,rws,rwr,rfl,rfs);
                   feetServoR.write(k);
                   delay(robot_speed);
                   Serial.println("Hello");

                  }

                  for(int i = rts; i>=rtf ;i--)
                    {
                     thighServoR.write(i);
                     float j = map(i,rts,rtf,rks,rkb);
                     kneeServoR.write(j);
                     float m = map(i,rts,rtf,ras,rab);
                     ankleServoR.write(m);
                     float n = map(i,rts,rtf,lks,lkf);
                     kneeServoL.write(n);
                    float p= map(i,rts,rtf,rfs,rfr);
                      feetServoR.write(p);
                     delay(robot_speed);
                      }
         }

void left_feet_stable(int lfs)  //4
                 {
                           for( float i =lfl; i < lfs; i++)
                          {
                            feetServoL.write(i);
                            delay(pos_delay);
                            Serial.print("LEFT side");


                        }
```

```
                          }

void right_feet_stable_after_forward_motion()  //5
                        {
                            waistServoR.write(rwr);
                            delay(100);
                            feetServoR.write(rfr);
                             for(int i = rtf; i<=rts ;i++)

                              {
                               thighServoR.write(i);
                               float j = map(i,rtf,rts,rkb,rks);
                               kneeServoR.write(j);
                               float m = map(i,rtf,rts,rab,ras);
                               ankleServoR.write(m);
                               float n = map(i,rtf,rts,lkf,lks);
                               kneeServoL.write(n);
                              float p= map(i,rtf,rts,rfr,rfs);
                                feetServoR.write(p);
                               delay(pos_delay);
                             }
                      }
void left_leg_forward()
                    {
                          for(int l=lws; l> lwl;l--)
                            {
                             waistServoL.write(l);
                             float k = map(l,lws,lwl,lfs,lfl);
                             feetServoL.write(k);
                             delay(robot_speed);
                             Serial.println("Hello");
                            }
                             thighServoL.write(10);
```

```
                         delay(3000);
                        /*
                         for(int i = lts; i>=ltf ;i--)
                           {
                            thighServoL.write(i);
                            float j = map(i,lts,ltf,lks,lkb);
                            kneeServoR.write(j);
                            delay(robot_speed);
                           }*/
                       }
void right_feet_stable()
                       {
                         feetServoR.write(rfs);
                       }

void right_leg_stable()//change the angle nmes and sign
                     {
                         for(int l=rws; l> rwr;l--)
                         {
                          waistServoR.write(l);
                          float k = map(l,rws,rwr,rfs,rfr);
                          feetServoR.write(k);
                          delay(robot_speed);
                          Serial.println("Hello");
                         }
                          thighServoR.write(10);
                          delay(3000);
                        /*
                         for(int i = rts; i>=rtf ;i--)
                           {
                            thighServoR.write(i);
                            float j = map(i,rts,rtf,rks,rkb);
                            kneeServoR.write(j);
```

```
                              delay(robot_speed);
                             }*/
                     }
void left_leg_stable()
                     {
                           for(int l=lws; l> lwl;l--)
                           {
                            waistServoL.write(l);
                            float k = map(l,lws,lwl,lfs,lfl);
                            feetServoL.write(k);
                            delay(robot_speed);
                            Serial.println("Hello");
                           }

                            thighServoL.write(10);
                            delay(3000);
                          /*
                           for(int i = lts; i>=ltf ;i--)
                             {
                              thighServoL.write(i);
                              float j = map(i,lts,ltf,lks,lkb);
                              kneeServoR.write(j);
                              delay(robot_speed);
                             }*/
                       }
void right_leg_bend();
void left_leg_forward();
void right_feet_table();
void left_leg_bend();
void right_leg_forward();
void left_leg_stable();
void left_leg_stable();
void print_acc_value()
```

```
            {
                        xVal = mySensor.readAxis('x'); //Read out the 'x' Axis
                        yVal = mySensor.readAxis('y'); //Read out the 'y' Axis
                        zVal = mySensor.readAxis('z'); //Read out the 'z' Axis

               Serial.print(xVal,DEC);
               Serial.print("    ");
               Serial.print(yVal,DEC);
               Serial.print("    ");
               Serial.print(zVal,DEC);
               Serial.println();
               delay(100);
              // return (xVal,yVal,zVal);
            }

void balance ()
            {
              while(1)
            {
               print_gyro();
               xVal =comppAngleX;
                       xVal = mySensor.readAxis('x'); //Read out the 'x' Axis
                       yVal = mySensor.readAxis('y'); //Read out the 'y' Axis
                       zVal = mySensor.readAxis('z'); //Read out the 'z' Axis

               Serial.print("Inside baance Loop, yVal: ");
               Serial.print(xVal, DEC);
               Serial.print("    ");
               Serial.print(yVal, DEC);
               Serial.print("    ");
               Serial.print(zVal, DEC);
               Serial.print("    ");*/
```

```
                float m = avg_x + 5;
                float n = avg_x - 5;
                        if (xVal > m)
                         {
                                                rtb =
map(comppAngleY,226,132,180,0);
                                                ltb = map(comppAngleY,226,132,0,180);

                                                 if(ltb > 180 || rtb < 0)
                                                   {
                                                   stable();
                                                  //Serial.print("RTB");
                                                  Serial.print(" ");
                                                  Serial.print(rtb,DEC);
                                                  Serial.print(" ");
                                                  //Serial.print("LTB:");
                                                  Serial.print(" ");
                                                  Serial.print(ltb,DEC);
                                                  Serial.print(" ");
                                                  //Serial.print("BACKWARD CROSS
180 OR 0");
                                                  Serial.print("    ");
                                                     //ltb =180;
                                                   }
                                                if(rtb<0)
                                                  {
                                                     stable();
                                                   //rtb = 0;
                                                  }
                                                  else
                                                 {
                                                   rtb = rts - 1; // previous (rts-1)
                                                   ltb = lts + 1;
```

```
                                rab = ras - .5;
                                rab = las + .5;

                              thighServoR.write(rtb);
                              thighServoL.write(ltb);
                               Serial.print(" ");
                               Serial.print(rtb,DEC);
                               Serial.print(" ");
                               Serial.print(" ");
                               Serial.print(ltb,DEC);
                               Serial.print(" ");
                               //Serial.print("FALLING
BACKWARD");

                               Serial.print("    ");
                                     rts=rtb;
                                     lts=ltb;
                              }
                        ankleServoR.write(rab);
                       ankleServoL.write(lab);


                           ras=rab;
                           las=lab;
                            }
                              else if(xVal < n)
               {
                          rtb = map(comppAngleX,226,132,180,0);
                          ltb = map(comppAngleX,226,132,0,180);
                          if(rtf > 180 || ltf < 0)
                               {
                                stable();
                                Serial.print(" ");
                                Serial.print(rtf,DEC);
                                Serial.print("LTF:");
```

```
                                                      Serial.print(" ");
                                                      Serial.print(ltf,DEC);
                                                      Serial.print(" ");
                                                      Serial.print("FORWARD CROSS 0
or 180");

                                                      Serial.print("    ");
                                                     }
                                          if(ltf<0)
                                                     {
                                                      stable();
                                                     }
                                            else
                                            {
                                               rtf = rts + 1; //previouss(rts+1)
                                               ltf = lts - 1;
                                               raf = ras + .5;
                                               laf = las - .5;

                                              thighServoR.write(rtf);
                                              thighServoL.write(ltf);
                                              ankleServoR.write(raf);
                                             // ankleServoL.write(laf);

                                            Serial.print("RTF:");
                                            Serial.print(" ");
                                            Serial.print(rtf,DEC);
                                            Serial.print("LTF:");
                                            Serial.print(" ");
                                            Serial.print(ltf,DEC);
                                            Serial.print(" ");
                                            Serial.print("FALLING FORWARD");
                                            Serial.print("    ");
                                                     rts=rtf;
```

```
                             lts=ltf;
                             }
                        ras=raf;
                        las=laf;
            }
              else
              {
                       lts = 90;
                       rts = 70;
                       ras = 58;
                       las = 77;
                    thighServoL.write(lts);
                    thighServoR.write(rts);
                    kneeServoL.write(las);
                    kneeServoR.write(ras);
                  Serial.print("RTS: ");
                  Serial.print(" ");
                  Serial.print(rts,DEC);
                  Serial.print(" ");
                  Serial.print("LTS:");
                  Serial.print(" ");
                  Serial.print(lts,DEC);
                  Serial.print(" ");
                  Serial.print("STAND STRAIGHT");
                  Serial.print("    ");
          }
          Serial.println();
          delay(50);
        }
        }
float find_average()
          {  sum=0;
           for(int i=0;i<1000;i++)
```

```
{
  print_gyro();
     yVal = mySensor.readAxis('y'); //Read out the 'y' Axis
     y_val[i] = yVal;
   xVal = comppAngleX;
   x_val[i]= xVal;
   sum += x_val[i];

 Serial.print("comppAngleX = ");
  Serial.print(comppAngleX, DEC);
  Serial.print("    ");
  Serial.print("    ");
   Serial.print("xVal[");
  Serial.print(i,DEC);
  Serial.print("] =  ");
 Serial.print(x_val[i],DEC);
  Serial.print("    ");

  Serial.print(sum,DEC);
  Serial.print(x_val[1000],DEC);
  Serial.println();
  delay(20);

}
 avg_x = sum/ 1000;
 Serial.print("Sum is:");
 Serial.print(sum,DEC);
 Serial.print("    ");
 Serial.print("The average of 1000 values is: ");
 Serial.print("    ");
 Serial.print(avg_x,DEC);
 Serial.print("    ");
 Serial.println();
```

```
                        delay(20);
                    }
void print_gyro()
                    {
                        accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
                       double gyroXrate = -((gx-zeroValue[3])/131); //Change the
sensitivity after your sensor
                        gyroXangle += gyroXrate*((double)(micros()-
timer)/1000000); // Without any filter

                        double gyroYrate = ((gy-zeroValue[4])/131);
                        gyroYangle += gyroYrate*((double)(micros()-
timer)/1000000); // Without any filter

                        ///////////////////////////
                        //The acc X and Y angle///
                        //////////////////////////

                        double accXval = ax-zeroValue[0];
                        double accZval = ay-zeroValue[2];
                        double accXangle =
(atan2(accXval,accZval)+PI)*RAD_TO_DEG;

                        double accYval = ay-zeroValue[1];
                        accZval = ay-zeroValue[2];
                        double accYangle =
(atan2(accYval,accZval)+PI)*RAD_TO_DEG;

                        compAngleX =
(0.93*(compAngleX+(gyroXrate*(double)(micros()-
timer)/1000000)))+(0.07*accXangle);
```

```
                      compAngleY =
(0.93*(compAngleY+(gyroYrate*(double)(micros()-
timer)/1000000)))+(0.07*accYangle);

                     comppAngleX = compAngleX;
                     comppAngleY = compAngleY;
                     timer = micros();

                    Serial.print("    ");
                    Serial.print(compAngleX);
                    Serial.print("\t");
                    Serial.print(compAngleY);
                    Serial.print("\t");
                    Serial.print(timer); Serial.print("\t");
                    Serial.print("\n");

                     delay(10);
                      if(compAngleX <180)
                      {
                      comppAngleX = map (compAngleX,0,180,180,0);
                      }
                      else
                      {
                       comppAngleX = map (compAngleX,181,360,360,181);
                      }
                  }
void loop()
{
                     //RECEIVE VALUE FROM MMA_7455
                    xVal = mySensor.readAxis('x'); //Read out the 'x' Axis
                    yVal = mySensor.readAxis('y'); //Read out the 'y' Axis
                    zVal = mySensor.readAxis('z'); //Read out the 'z' Axis
```

```
  Serial.print("BEFORE LOOP, Values : ");
  Serial.print(xVal, DEC);
  Serial.print("    ");
   Serial.print(yVal, DEC);
  Serial.print("    ");
   Serial.print(zVal, DEC);
  Serial.print("    ");
  Serial.println();
  delay(100);
  stable(rws,rts,rks,ras,rfs,lws,lts,lks,las,lfs);
  delay(pos_delay);

  print_acc_value();
   Serial.print("AFTER LOOP, Values : ");
  Serial.print(xVal, DEC);
  Serial.print("    ");
   Serial.print(yVal, DEC);
  Serial.print("    ");
   Serial.print(zVal, DEC);
  Serial.print("    ");
  Serial.println();
  delay(100);
  balance (yVal);
  Serial.println();
  delay(100);

//*******START TAKING VALUE *******

    // read raw accel/gyro measurements from device
//*****    accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

    // these methods (and a few others) are also available
    //accelgyro.getAcceleration(&ax, &ay, &az);
```

```
                //accelgyro.getRotation(&gx, &gy, &gz);
                // display tab-separated accel/gyro x/y/z values

                 Serial.print("a/g:\t");
                Serial.print(ax); Serial.print("\t");
                Serial.print(ay); Serial.print("\t");
                Serial.print(az); Serial.print("\t");
                Serial.print(gx); Serial.print("\t");
                Serial.print(gy); Serial.print("\t");
                Serial.println(gz);

               //***** // KALMAN FILTER-START // ****

                  // **** //GYRO ***
if(flag == 0)
{
stable();
delay(3000);
print_gyro();
Serial.println("next");
find_average();

flag =1;
}
else
{
  stable();
  delay(pos_delay);
  left_feet_bend(lfl,lwl);
  delay(pos_delay);
  right_leg_forward();//rts,rtf,rks,rkb
  delay(pos_delay);
  left_feet_stable(lfs);
```

```
         delay(500);
         right_feet_stable_after_forward_motion();
         delay(pos_delay);
        right_feet_bend();
         delay(pos_delay);
        delay(3000);

        left_leg_forward();
         delay(pos_delay);
         Serial.print("Else");
        left_feet_stable(int lfs)

         delay(pos_delay);
         delay(pos_delay);
         balance();
         Serial.println("previous");
        }
}
```

A list of publications produced by candidate as a result of the project

Journal paper:

1. Alamgir Hossain, Rahid Zaman, Miftahur Rahman, Raihan Masud , Niloy Arafat and Fardan Abdullah, "**Design and kick analysis of a Soccer Robot**," in *"**Applied Mechanics and Materials Journal** "(ISSN: 1660-9336).)*, Sydney Australia, 2013.

Conference Paper:

1. Alamgir Hossain, Rahid Zaman, Miftahur Rahman, Raihan Masud , Niloy Arafat and Fardan Abdullah, Mizanur Rahman, Aziz Rahman "**Development of an Integrated Vision system to Control a Soccer Playing Humanoid Robot** "(accepted) in Proc **4th Global Engineering, Science and Technology Conference** (ISSN 2201-6848)**,** Dhaka, Bangladesh, 2013.