COLORBOT- THE COLOR TRACKING ROBOT

MAIN PROJECT REPORT

Submitted by

ABIN JOY (JYALECS001)

ANN THERESA KENNY (JYALECS011)

DONIA AUGUSTINE (JYALECS019)

NISHAL GOVIND K R (JYALECS034)

SINI JOSE (JYALECS042)

in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY (B.TECH)

in

COMPUTER SCIENCE & ENGINEERING

of

UNIVERSITY OF CALICUT

Under the guidance of

Ms. NAMITHA T N

April 2015

Department of Computer Science & Engineering

JYOTHI ENGINEERING COLLEGE, CHERUTHURUTHY

THRISSUR 679 531

Department of Computer Science & Engineering

JYOTHI ENGINEERING COLLEGE, CHERUTHURUTHY

THRISSUR 679 531

APRIL 2015

BONAFIDE CERTIFICATE

This is to certify that the project report entitled **COLORBOT- THE COLOR TRACKING ROBOT** submitted by **ABIN JOY, ANN THERESA KENNY, DONIA AUGUSTINE, NISHAL GOVIND K R, SINI JOSE** in partial fulfilment of the requirements for the award of **Bachelor of Technology** degree in **Computer Science & Engineering** of **University of Calicut** is the bonafide work carried out by them under our supervision and guidance.

Ms.Namitha T N	**Ms.Jyothis T S**	**Prof.Muralee Krishnan C**
Project Guide	Project Coordinator	HOD
Dept. of CSE	Dept. of CSE	Dept. of CSE

CONTENTS

ACKNOWLEDGEMENT

We take this opportunity to express our heart felt gratitude to all respected personalities who had guided, inspired and helped us in the successful completion of this project. First and foremost, we express our thanks to **The Lord Almighty** for guiding us in this endeavour and making it a success.

We take immense pleasure in thanking our principal **Dr.K K Babu** and the **Management** for having permitted us to carry out this project. Our sincere thanks to the Head of the Department of Computer Science Engineering and our mentor **Prof.Muralee Krishnan C** for permitting us to make use of the facilities available in the department and for his guidance and encouragement to carry out the project successfully.

We express our sincere gratitude to Internal Project Coordinator **Ms.Jyothis T S** for her valuable supervision and timely suggestions.We are happy to express our deepest gratitude to **Ms.Namitha T N** , our mentor for her able guidance and continuous encouragement.

Last but not least we extend our gratefulness to all teaching and non teaching staff who directly or indirectly involved in the successful completion of this project work.Finally thanks to all our friends who have patiently extended all sorts of help for accomplishing this undertaking.

ABSTRACT

Our project facilitates a mobile robot vision system for object color tracking. The color tracking system uses the combined concept of robotics and app development in order to recognize the color of an object. We aim at implementing a vision system of the robot that gives the robot ability to recognize the color and thus resulting in its corresponding movement. The implementation is carried out in two stages: color recognition, and routing the robot. The image is captured using the mobile camera and the captured image is converted from analog to digital for further processing. Frequency of the selected color is generated by the android application.This frequency of the color is fed as input to the microprocessor. Motion of the robot comprises of robotic head rotation and alignment of the robot. Routing the robot facilitates the robotic movement. Thereby we are designing an efficient system for the movement of the robot using the color.

List of Figures

Chapter 1

INTRODUCTION

In recent years we have seen a drastic change in industries approaches to use color tracking embedded systems to fulfill their needs for a higher production and precise quality.However, new technologies have been made available for the integration of color tracking with a vision system which are cost-effective solutions. The applications of color tracking vision systems are many and still increasing.

1.1 OVERVIEW

Color vision is the ability of an organism or machine to distinguish objects based on the wavelengths (or frequencies) of the light they reflect, emit, or transmit.Although several color tracking systems are available today, they do not provide the required efficiency and accuracy. This thesis is brought up with an alternative solution to color tracking.The characteristic colors are, from long to short wavelengths (and, correspondingly, from low to high frequency), red, orange, yellow, green, cyan, blue, and violet. Sufficient differences in wavelength cause a difference in the perceived hue; the just-noticeable difference in wavelength varies from about 1 nm in the blue-green and yellow wavelengths, to 10 nm and more in the longer red and shorter blue wavelengths. Although the human eye can distinguish up to a few hundred hues, when those pure spectral colors are mixed together or diluted with white light, the number of distinguishable chromaticities can be quite high. Recent research in color tracking focused on programming within the chip, while we propose an android app for this purpose there by reducing the cost and complexity [5].

1.2 MOTIVATION AND SCOPE OF THE PROJECT

Many color recognition systems match the color contained within images to a predefined color. Determining the extent of color control needed in the system depends upon the application and how the tools are applied. If a part simply needs to be verified as red and the system need not analyze the exact color composition, strict controls may not be needed. However, if a

machine-vision system needs to as certain that the red exactly matches a defined pantone color, the system may benefit from white balancing and color calibration. Special attention should also be given to control even minor color shifts that are the result of aging lights, temperature, and ambient lighting conditions. When designing a color vision system, the wavelengths contained in the light source, those reflected and absorbed by the surface of the part, and the frequency-response curve of the camera must be considered. Like the eye, a camera gathers light reflected (or transmitted) from objects. If light strikes a shiny surface and reflects into the camera, surface color information is lost. For this reason, color vision systems will often benefit from using diffuse or disparate light sources such as ring lights. Fortunately, several companies provide products that improve the robustness of color systems. Most often, color machine-vision systems should use cameras that provide automatic white balancing, and lights that provide a uniform distribution of intensity over the color spectrum.

Like monochrome systems, color machine-vision systems operate effectively if they can easily detect the differences between good and bad parts [1].Choosing the correct color space is an important aspect of the system design, as it will enhance the separation distance applied to the colors. In order to track a color from the image, the input image frame is searched against a reference model describing the appearance of the object. This reference can be based on image patches, thus describing how the tracked region should look like pixel-wise, on contours, thus describing the overall shape, and/or on global descriptors such as color models.

Today, with advancements in machine vision technology, imaging processing software and inspection tools, processing color images is becoming easier, faster and less expensive. Tracking objects based on color is one of the quickest and easiest methods for tracking an object from one image frame to the next. The speed of this technique makes it very attractive for near-realtime applications.

1.3 APPLICATIONS

- Military: to detect the enemies using face detection procedures.
- Car Parking: used to properly align cars in their parking slots based on the parking lines.
- Library: to correctly route the orders to the customers based on the book image.
- Bomb Diffusion: widely used to diffuse touch sensitive explosives.

Chapter 2

LITERATURE SURVEY

2.1 INTRODUCTION

In the literature survey we researched on papers related to our project in detail.From this search we came to know that there are several methods to implement the color tracking robot and we selected certain papers for further studies.The selected papers are:-

- Real-Time Embedded Color Tracking For Motion Robot System
- Mobile Robot Navigation And Target Tracking System
- FPGA Based Color Tracking For Mobile Robot System
- Mobile Robot Vision System For Object Color Tracking
- Towards Robust Skin Color Detection And Tracking
- FPGA Based Real-Time Object Tracking For Mobile Robot
- Multi Object Color Tracking For Multi Robot System Enhancement
- Interactive Offline Tracking For Color Objects
- Using Color for Object Recognition
- Color-Based Tracking of Heads And Other Mobile Objects at Video Frame Rates

2.1.1 REAL-TIME EMBEDDED COLOR TRACKING FOR MOTION ROBOT SYSTEMS

This paper describes about the real-time embedded color tracking for motion robot system. FPGA technology is applied.FPGA architecture presented here can track a solid object.we capture the video from the camera,process it on FPGA and display.Images received by the camera are in the form of bayer pattern which are then transformed in to RGB color space.RGB is then transformed in to HSV so as to improve the result of color

segmentation in different light intensities.

To control the direction of the robot,a discrete colored object on the solid colored could be used.Moving the colored object in the right direction should instruct the robot to move in right direction.Images received by the digital camera,the colors are segmented and the binary image for each object is generated inside FPGA.Objects are being tracked based on their color.

2.1.2 MOBILE ROBOT NAVIGATION AND TARGET TRACKING SYSTEMS

This paper presents the framework for the navigation and target tracking system for a mobile robot. Navigation and target tracking are to be performed using a Microsoft Xbox Kinect sensor which provides RGB color and 3D depth imaging data to an x86 based computer onboard the robot running Ubuntu Linux [2]. A fuzzy logic controller to be implemented on the computer is considered for control of the robot in obstacle avoidance and target following.An eventual goal of this work is to create a multi-agent robot system that is able to work autonomously in an outdoor environment.Navigation of a mobile robot consists of path planning and heading changes towards the target destination.The purpose of object recognition in this paper is two-fold, firstly to identify objects to track in the environment and secondly to identify free space in the space in front of the robot for navigation purposes.

RGB-D Sensor: The Kinect is an RGB camera along with a 3D depth ranging sensor that works through infrared light. This sensor provides access to two images, one RGB image and also a 3D depth image.

Color Image Recognition: Use of the rg-chromaticity color space, adaptive color processing edge detection and color segmentation are proposed as being part of the color image recognition routine. This routine will be used for target tracking and navigation for the mobile robot.

Color Segmentation: The combination of using the rg-chromaticity color space and light intensity correction provide two components necessary for color image recognition. An image can be further segmented into components containing colors similar to primary, secondary, and tertiary colors of varying intensities. With the color segmented image, candidates of objects known to be a certain color can be isolated in an image frame and examined further.

2.1.3 FPGA BASED COLOR TRACKING FOR MOBILE ROBOT SYSTEMS

Describes about the FPGA based color tracking for mobile robot system. Designed the proposed system using VHDL and implemented the design in a Diligent Nexys2 FPGA. FPGA circuit can track a solid color object in a scene when it receives as input a digitized PAL/NTSC stream.Then capture the video from a camera, process it on FPGA and then display it on a 3.2 display.The attached image sensors collect data from the surrounding and send it to an FPGA board. The FPGA embedded processor will control the sensors and processes the measured values.A convenient system to use is hue, saturation, luminance and value (HSL/HSV). To calculate the HSV value for each pixel in the image, first the data coming from the digital camera is transformed into RGB and then from RGB values for each pixel, transform them in to HSV values. The output of the digital camera is in the form of Bayer pattern.It is converted in to RGB and the RGB is converted to HSV [3].To implement the methodology in the FPGA, it needs to define the maximum and minimum limits for HSV components.To control the direction of the robot, a discrete colored object on the solid color object could be used.Moving the colored object in the right direction should instruct the robot to move in the right direction.

2.1.4 MOBILE ROBOT VISION SYSTEM FOR OBJECT COLOR TRACKING

In the paper a vision system of the eMIR mobile robot is presented that gives the robot ability to recognize objects by their color. For the object recognition HSV color system is used where a picture divides into cells with dimensions of 9x9 pixels. After determining the cell similarity with the reference sample, identified cells are grouped in clusters that are a basis for the object recognition. For each object, size and center of gravity are determined. The whole process of recognition shown in figure 2.1 is performed in a real time, with own developed algorithms for image processing. Recognition process is not carried out in the whole image, but only around the area where the object is detected in the previous recognition step. This allows fast search and recognition of not only static objects but also moving objects. The motion of the robot towards recognized object takes place in three stages: 1. Robots rotation, to allow bringing of the object being searched for in the robots visual field, 2.Aligning of the robot, by its rotation, until they found object comes into the center of the image that is in front of the robot, and 3.Movement of the robot (simultaneous translation and rotation) to a distance of 10 cm.

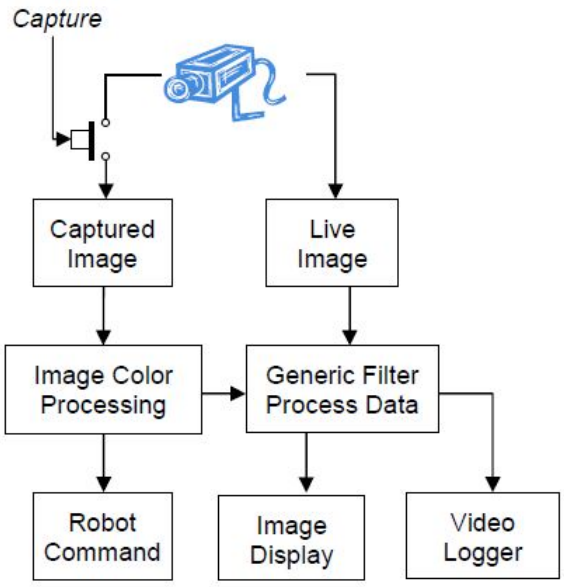

Figure 2.1: Signal flow and image processing

By defining the reference sample the robot task is to locate and move closer to designate object up to the given distance. For achieving this goal, there are three modes of the robot behavior:

- Search is activated if the selected object is not in the robot visual field. Strategy for object searching is by robot rotation, so long as the object is not found in the visual field, or until the specified time has elapsed. Rotation speed should not be too high because it can happen that although requested object is observed, it can get out of the camera field of view because the robot has not reached stop.
- Align is triggered when the selected object is in the visual field of the camera, and previously the search behavior was active. At this stage, the robot rotates until it finds the selected object close to the horizontal center of the visual field.
- Approach is triggered when the selected object is found around the middle of the visual field. Approaching to the selected object begins with translation speed setting and correcting the direction of rotation. Front rangefinder provides information about the distance to the object, and when the distance is equal to or less than the default value, the robot stops.

During each phase of the robot task execution it is possible to record robot kinematics; therefore the robot behavior can be properly quantified, not only visually monitored. In addition, video from the robot camera can be recorded with elements of recognition.

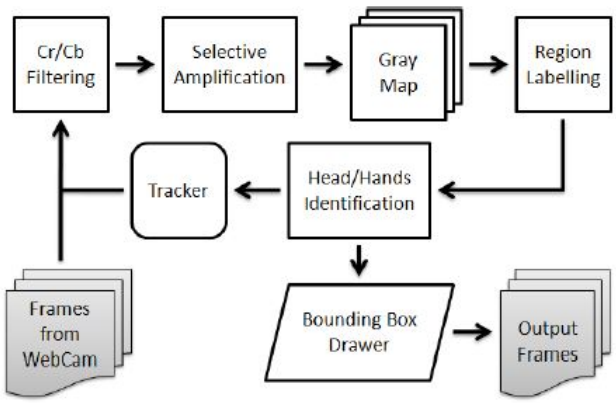

Figure 2.2: Skin detection and tracking algorithm

2.1.5 TOWARDS ROBUST SKIN COLOR DETECTION AND TRACKING

This paper presents our effort on designing such an algorithm with a number of simple cascaded filters. This paper presents the outcome of an attempt to develop a human skin detection algorithm for the use of data acquisition in an Automatic Sign Language Translator (ASLT). Proposed skin detection algorithm shown in figure 2.2 takes the form of a cascaded filter with number of stages treating the different properties of the input image.

Input frames must be in RGB format with 8-bit pixel resolution. Input frames at first goes through two pixel level processing stages, Cr/Cb Filtering and Selective Amplification. Then they it will be processed n-by-n pixel size blocks in Gray Map stage. Region labelling simply labels the skin regions identified by the previous stages. Head/Hands identification stage runs a logic routine to recognize head and hands based on few assumptions. Tracker keeps data of each individual body articles position over the time, which is again fed back to the algorithm to predict search areas for subsequent input frames. Bounding box for each skin region is drawn at the Bounding Box Drawer stage. As far as the background complexity and illumination is tolerable, this algorithm can perform near-real time with adequate accuracy.

2.1.6 FPGA BASED REAL-TIME OBJECT TRACKING FOR MOBILE ROBOT

Proposes an embedded vision system for real-time moving object tracking using modified mean shift algorithm for mobile robot application. This design of modified mean shift algorithm fully utilizing the advanced parallelism of FPGA is capable of processing

real time PAL video.This hardware implementation realizes time consumed color space transformation using pipeline operation. Thereafter Mass center calculation method used to balance the tracking complication and precision. Finally TDM technology obviously saves the FPGA hardware resource.In FPGA based algorithm architecture System captures video with a CCD camera. It is then Convert it in to digital RGB video. When operator chooses a red rectangle on real time video as target location by keyboard, tracking system begins to enable color space transformation module, modified mean shift module and tracking result display module. Tracking video data need to convert to LVDS signal for LCD display from LCD driver module and LVDS transformation module. In the color space transformation module extend the RGB cube to HSV cone space transformation matrix in FPGA by multiplying a constant 16 to H operation.Kernel based mass center calculation is performed. Pipeline based parallel mass center hardware architecture is used. A modified mean-shift real-time object tracking algorithm and its embedded FPGA based implementation have been done.

2.1.7 MULTI OBJECT COLOR TRACKING FOR MULTI ROBOT SYSTEM ENHANCEMENT

This paper presents the integration of a proposed enhanced multi-object color tracking and navigation algorithms for multi-robot navigation performance enhancement. The proposed integrated system architecture consists of a server with an attached camera vision sensor and mobile clients (robots), each client is equipped with a PDA, Bluetooth communication module, and MCU-based control system for robots motion control. Proposed system discusses and extends embedded vision system and 2D visual tracking algorithm in real-time. The hardware system comprises camera module, and real-time vision processing, interface, and control boards. The software system consists of four different modules:detection, tracking, control and motion planning. There are two methods that are useful for motion detection. The first technique is based on adaptive threshold, while the second is the color detection using hue segmentation. After the detection the problem is to track and recognize these moving objects preserving their locations even if any object stops its motion. In this section a multi-object color tracking model will be introduced, the model integrates the motion and color tracking taking the advantages of the two methods to optimize the processing time.

The color of an object as shown in the image can differ quite significantly from what is seen using naked eye. Many environmental factors can affect the color of an object in the video exposure of the camera, such as white balance setting, ambient lighting, and reflection of the spot lights around. Theoretically, an object of red color should have the RGB value of (255, 0, 0), in practice, one can never find such value although the object is really red in color when seeing it with naked eyes. Testing the RGB values produced by the camera attached to server showed that they do not give enough information to identify all objects in the region of interest. The HSV color space provides us with a component containing the color wavelengths called Hue (H) which defines the set of the color, Saturation (S) which defines how pure the color will be, and Value (V) which defines the brightness of the color

In this model, segmentation by hue is achieved by setting all pixels that have a hue in a specific range around the hue of the background to belong to the background. Pixels with a different hue are set to be objects. The range of hues which will be classified as background is selected by examination of the hue components histogram, where the background will have a distinct peak. The main advantage of this method is that effects of lighting and shadows can be reduced since the intensity and saturation information is separated from the hue component in the HSV color space. A disadvantage of this model is that it is usable only when there are enough colors in the image [4].

2.1.8 INTERACTIVE OFFLINE TRACKING FOR COLOR OBJECTS

In this paper, we present an interactive offline tracking system for generic color objects. To fully exploit user input and reduce user interaction, the tracking problem is addressed in a global optimization frame-work. The optimization is efficiently performed through three steps. First, from user input we train a fast object detector that locates candidate objects in the video based on proposed features called boosted color bin. Second, we exploit the temporal coherence to generate multiple object trajectories based on a global best-first strategy. Last,an optimal object path is found by dynamic programming.The major contribution of this paper is two fold. First, we propose a three step optimization framework that addresses offline interactive tracking problem as well as an efficient implementation:

- Fast detection in sparse video frames;
- Multiple object trajectory tracking based on detection

- Global path optimization based on multiple hypotheses Second, we propose a novel color feature, called boosted color bin that is flexible and effective for generic color objects.

  Color bin: For generic color objects, the color histogram is rotation and scale invariant, and has been shown very robust for object tracking. However, computation of 3D color histograms is too slow. Instead, we propose simple and fast color features - color bins of multiple 1D color histograms.

For a given object window, we project all pixel colors on a set of one dimensional lines in RGB color space. These lines have different directions and pass through the point. In our implementation, we evenly sample the direction by 13 lines. Then, a 1D (normalized) histogram of the projected values is calculated on each line. We use 8 bins for each histogram through an empirical comparison and treat all 13 * 8 = 104 color bins as our features. They are extracted in a short constant time for any image rectangle using integral histogram data structure. In the offline tracking framework, since all integral histograms can be pre-computed, the feature extraction is extremely fast.

2.1.9 USING COLOR FOR OBJECT RECOGNITION

In this paper the ROI selection experiments show that when there is segmentation available, we can get better results. Depending on the amount information such as complete segmentation, localized parts, or just random point on the object, the classifier performs differently. In this paper, different color spaces and their combinations were tested to see which one results in the best classification accuracy.More thorough experiments should be done using the part annotation data available for CUB200a dataset to see how much we can get from part locations data. Some experiments like automated segmentation or cropped bounding box should also be tested to able to compare the result of localized parts information. Also, a combination of segmentation and the brush strokes annotation could be a good option too. After building a color based classifier for using color feature, the next problem to study is the way to combine it with the output of other classification methods. Multiple Kernel Learning (MKL) and simple voting are possible options for this matter.

The representation of color is a factor that should be investigated too. The way that experiments are done is first find the distance between each color channels output. Then combine these distance measure using a weighted sum. For example for RGB, one dis-

tance matrix which is of size of the number of test image by the number of training images is created for c2 distance between marginal R histograms. Another one is created for G and so one for B. In the case of CUB200 they could be as big as 6000 which needs significant amount of time and storage to deal with. All weights are set to 1 in this experiment. According to this result, there is a quite large difference between different color spaces. For example, some color spaces like YIQ, YCbCr and HSV got better results possibly because of their orthogonality characteristics.

To optimize the accuracy of the classification, one may want to search for the best set of weights to calculate a better between two set of histograms of different color channels. In this project, this is done by using a simulated annealing approach. Simulated annealing is one of hill climbing type methods that decreases the learning factor (step size) the classification accuracy. It always saves the best set of weights which have been seen so far. Therefore the graph is always ascending.

2.1.10 COLOR BASED TRACKING OF HEADS AND OTHER MOBILE OBJECTS AT VIDEO FRAME RATES

We develop a simple and very fast method for object trackIng based exclusively on color information in digitizes video frames.Running on a silicon graphics R4600 Indy system with an IndyCam, our algorithm is capable of simultaneously tracking objects at full frame size (640*480 pixels) and video frame rate (30fps).Robustness with respect to occlusion is achieved via an explicit hypothesis-tree model of the occlusion process.We demonstrate the efficiency of our technique in the challenging task of tracing people,especially tracking human heads and hands.

This color tracking system is based on tracking regions of similar normalized color from frame to frame.Specifically, N regions R1,.....RN are defined with the extend of the object to be tracked; the size and relative position of these regions is assumed to be fixed. Of course there are some limitations to this scheme.The simplicity of the prior model in this particular head tracking application leads to ambiguity as to whether the head is being occluded from left to right or from the right. However, to the extend that our goal is robust tracking, and not so much an understanding of the occlusion behaviour, this is not a significant drawback.

Chapter 3

PROJECT MANAGEMENT

3.1 INTRODUCTION

Project management is the discipline of planning, organizing, securing, managing, leading, and controlling resources to achieve specific goals. A project is a temporary endeavor with a defined beginning and end (usually time-constrained, and often constrained by funding or deliverables),undertaken to meet unique goals and objectives, typically to bring about beneficial change or added value. The temporary nature of projects stands in contrast with business as usual (or operations), which are repetitive, permanent, or semi-permanent functional activities to produce products or services. In practice, the management of these two systems is often quite different, and as such requires the development of distinct technical skills and management strategies.

In our project we followed typical development phases of an engineering project

1. Initiation
2. Planning and Design
3. Execution and Construction
4. Monitoring and controlling sytems
5. Completion

3.1.1 INITIATION

The initiating processes determine the nature and scope of the project. The initiating stage should include a plan that encompasses the following areas :

1. analyzing the requirements in processing the image.
2. reviewing the pros and cons of different color spaces
3. takes into account the speed of rotation

4. analysing the cost and benefits
5. calculates the efficiency in adopting frequency as a recognition technique

3.1.2 PLANNING AND DESIGN

After the initiation stage, the project is planned to an appropriate level of detail. The main purpose is to plan time, cost and resources adequately to estimate the work needed and to effectively manage risk during project execution. As with the Initiation process group, a failure to adequately plan greatly reduces the project's chances of successfully accomplishing its goals.

- Determining how to plan the entire experimental setup
- Developing the scope statement
- Selecting the planning team
- Identifying the power and speed requirements
- Identifying the connectives needed to interconnect the required controllers and their connecting boards
- Developing the application
- Risk planning

3.1.3 EXECUTION

Executing consists of the processes used to complete the work defined in the project plan to accomplish the project's requirements. Execution process involves coordinating people and resources, as well as integrating and performing the activities of the project in accordance with the project management plan. The deliverables are produced as outputs from the processes performed as defined in the project management plan and other frameworks that might be applicable to the type of project at hand.

3.1.4 MONITORING AND CONTROLLING

Monitoring and controlling consists of those processes performed to observe project execution so that potential problems can be identified in a timely manner and corrective action can be taken, when necessary,to control the execution of the project. The key benefit is that project performance is observed and measured regularly to identify variances from the project management plan.

3.2 SYSTEM DEVELOPMENT LIFE CYCLE

The Systems development life cycle (SDLC), or Software development process in systems engineering, information systems and software engineering, is a process of creating or altering information systems, and the models and methodologies that people use to develop these systems [6]. In software engineering, the SDLC concept underpins many kinds of software development methodologies. These methodologies form the framework for planning and controlling the creation of an information system.

The SDLC phases serve as a programmatic guide to project activity and provide a flexible but consistent way to conduct projects to a depth matching the scope of the project. Each of the SDLC phase objectives are described in this section with key deliverables, a description of recommended tasks, and a summary of related control objectives for effective management. It is critical for the project manager to establish and monitor control objectives during each SDLC phase while executing projects. Control objectives help to provide a clear statement of the desired result or purpose and should be used throughout the entire SDLC process.

3.2.1 RAPID APPLICATION DEVELOPMENT MODEL

As the name suggests Rapid Application Development (RAD) model is an incremental software process model that focuses on short development cycle time. This model is a high-speed model which adapts many steps from waterfall model in which rapid development is achieved by using component based construction approach.

In case if project requirements are well understood and project scope is well known then RAD process enables a development team to create a fully functional system i.e. software product within a very short time period may be in days. RAD model is like other process models maps into the common and major framework activities.

PHASES OF RAD MODEL

Figure 3.1 shows the different stages of RAD model. It includes:

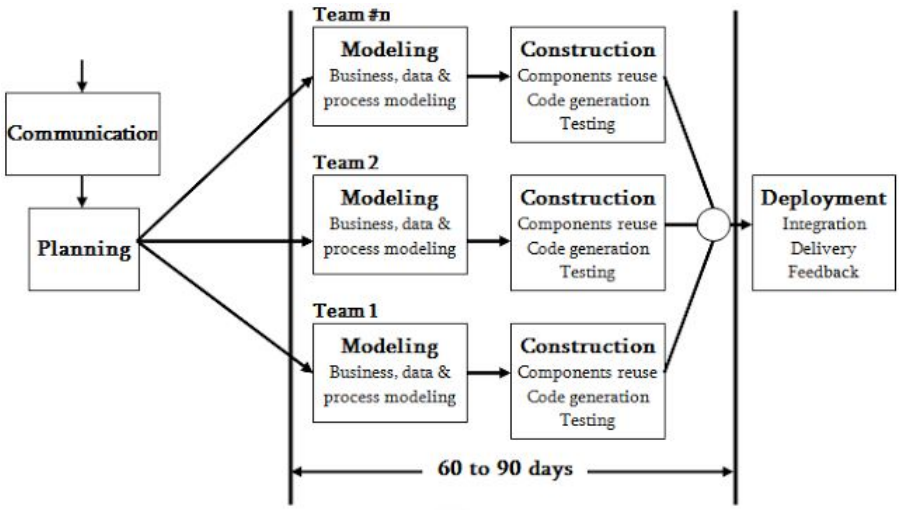

Figure 3.1: Rapid Application Development Model

COMMUNICATION: Communication is an activity which works to understand the business problem and the information characteristics that should be accommodate by the software.

PLANNING: Planning is required because numerous software teams works in parallel on different system functions.

MODELLING: Modelling includes 3 major phases-

- Business modeling
- Data modeling
- Process modeling

CONSTRUCTION: Construction focuses mainly on the use of existing software components and the application of automatic code generation.

DEPLOYMENT: Deployment establishes a basis for subsequent iterations if necessary. An application which can be modularize in a way that allows each major function to be completed in less than three months is useful for RAD. Each major function can be addressed individually by a separate RAD and then integrated to form a whole application.

ADVANTAGES OF RAD MODEL

- Flexible and adaptable to changes
- RAD generally incorporates short development cycles
- RAD involves user participation thereby increasing chances of early user community acceptance
- RAD realizes an overall reduction in project risk

Chapter 4
PROPOSED SYSTEM

4.1 INTRODUCTION

COLORBOT is an embedded mobile system which works based on a android application. The embedded system consist of a robot which identifies various colors and performs the corresponding movement for that particular color. The robot is controlled using a micro controller chip. The chip is embedded with a program that identifies different frequencies produced by the android application and perform the corresponding action specified.

The android application makes use of the mobile camera which captures the image. The user can choose the desired color from the image and based on that color a particular frequency is created and sent to the chip. Motion of the robot comprises of robotic head rotation and alignment of the robot. Routing the robot facilitates the robotic movement by rotating the head. The robotic head is aligned to the desired color and thereafter the motion of the robot follows the aligned path. Thereby we are designing an efficient system for the movement of the robot using the color.

Key features include:

- Easy tracking of specified color
- Frequency method provides more efficiency
- Chip set is not so complex
- Use of HSV algorithm provides flexibility
- Efficient routing of the robot

4.2 SOFTWARE AND HARDWARE REQUIREMENTS

4.2.1 SOFTWARE REQUIREMENTS

- Micro C
- Eclipse(android sdk)
- Java
- Windows 8
- Open CV

4.2.2 HARDWARE REQUIREMENTS

- HEXBUG Spider Toy
- Micro Controller
- Servo Motor

Chapter 5

SYSTEM DESIGN

Our project facilitates a mobile robot vision system for object color tracking. We aim at implementing a vision system of the robot that gives the robot ability to recognize the color and thus resulting in its corresponding movement. The implementation is carried out in two stages: color recognition, and routing the robot. Frequency of the particular color is fed as input to the microprocessor. Motion of the robot comprises of robotic head rotation and alignment of the robot. Routing the robot facilitates the robotic movement. Thereby we are design an efficient system for the movement of the robot using the color.

5.1 OBJECT COLOR TRACKING

The process of detecting objects is to analyze the information in the image which enable to track color. Object recognition is based on the color. After the recognition the problem is to track these objects preserving their locations. In this section a object color tracking model will be introduced, the model integrates the detection and color tracking taking the advantages of the methods to optimize the processing time. The modified algorithm is implemented and tested under different conditions to verify its robustness and its validity in detecting objects [7].

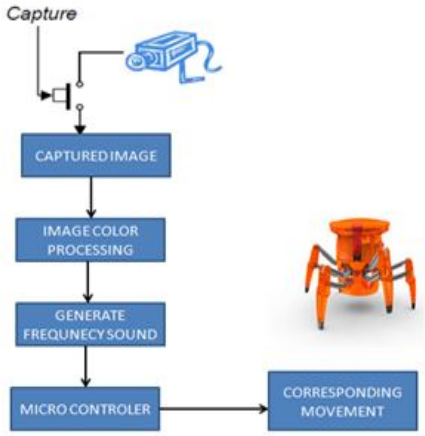

Figure 5.1: Flow diagram

The data flow diagram is shown in figure 5.1. Here the image is captured by the camera. The captured image undergo several image processing techniques and output is generated as the frequency sound. This frequency is fed as input to the micro controller. According to the signal generated by the android application the corresponding motion takes place.

HEXBUG SPIDER is a autonomous robot.It consists of two motors. First one control six legged crawling motion. Second one is to rotate its head on 360 degree. 3AA-batteries set internal mechanisms in motion. Frequency counter is used to measure frequency. Frequency counters are basically simple counter system with a limited time period for counting.

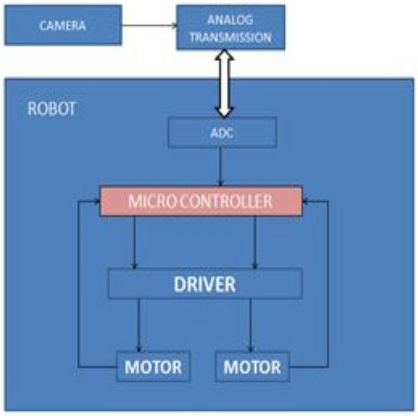

Figure 5.2: Internal structure

Figure 5.2 shows the overall view of the color tracking system. It uses a camera, basically an android phone camera to capture the image. An object color is identified by the android application and a corresponding analog signal is generated. This analog signal is fed to the ADC and a digital signal is obtained as output at the micro controller. The micro controller facilitates the control of the two motors using a driver circuit and helps in the navigation of the robot to the target.

5.2 SOFTWARES USED

5.2.1 MICRO C

MICROC is the powerful C compiler integrated in SUPER-FLASH which produces programs which can be run not only by SUPER-FLASH applications, but others as well. MICROC supports a simplified sub-set of the ANSI C standard. It implements the syntax and the typical operators of C, interprets the control structures, but does not manage pointers. By means of the virtual type, it is easy to access the mixed database of the SUPER-FLASH variables.

MICROC is a powerful compiler independent of the hardware, is simple and reliable, rich in functions and performances. With MICROC, the user can extend the functionality of the Development system enormously: in fact, more than 500 functions are available for numerous sought-after applications. The programs compiled with MICROC operate under the strict control of the runtime engine. This means only correct instructions are carried out, keeping the high level of reliability peculiar to SUPER-FLASH. Since it does not produce a machine code, the programs do not have to be recompiled to be used on other platforms.

Below are some of the possibilities offered by MICROC:

- Execution of calculations in floating point
- Execution of calculations which involve trigonometric functions
- Development of communication protocols
- Runtime modification of the characteristics of the Variables
- Deferred recording of Trends
- Deferred recording of Alarms
- Importation, processing, exportation of the data generated by the applications (Trend, Alarms, Recipes, etc.)
- Carrying out management of fully customised files
- Carrying out completely free control functions
- Implementation of control functions for the input data

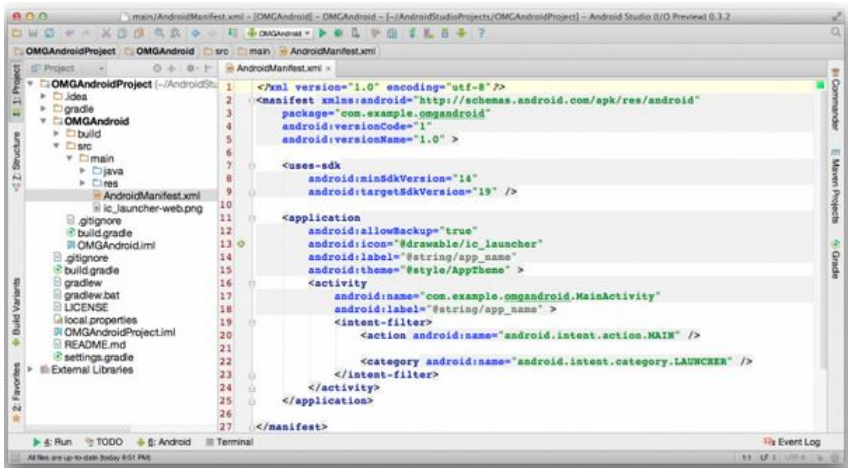

Figure 5.3: Android Studio

- Implementation of general control functions of data coherence
- Reduction in the SUPER-FLASH Variables needed for an application
- Possibility of protecting your know-how
- Making processing drivers seen by the system as a normal peripheral
- Interaction with the Event Management
- Interaction with SMARTDB

5.2.2 ANDROID STUDIO

Android Studio shown in figure 5.3 provides graphical tools for creating and managing Android projects, which contain everything that define your Android apps, from app source code to build configurations and test code. Each project contains one or more different types of modules, such as application modules, library modules, and test modules.Android Studio is an integrated development environment (IDE) for developing on the Android platform.

Main features :

- Developer Console: optimization tips, assistance for translation, referral tracking, campaigning and promotions - Usage Metrics.
- Android-specific refactoring and quick fixes.
- Template-based wizards to create common Android designs and components.

- A rich layout editor that allows users to drag-and-drop UI components, option to preview layouts on multiple screen configurations.
- Support for building Android Wear apps
- Built-in support for Google Cloud Platform, enabling integration with Google Cloud Messaging and App Engine.

5.2.3 JAVA

Java is a general-purpose computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA),meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of computer architecture. As of 2015, Java is one of the most popular programming languages in use, particularly for client-server web applications, with a reported 9 million developers. Java was originally developed by James Gosling at Sun Microsystems (which has since been acquired by Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them.

Java is a set of several computer software and specifications that provides a system for developing application software and deploying it in a cross-platform computing environment. Java is used in a wide variety of computing platforms from embedded devices and mobile phones to enterprise servers and supercomputers. Writing in the Java programming language is the primary way to produce code that will be deployed as byte code in a Java Virtual Machine (JVM); byte code compilers are also available for other languages. Writing in the Java programming language is the primary way to produce code that will be deployed as byte code in a Java Virtual Machine (JVM); byte code compilers are also available for other languages.

5.2.4 WINDOWS 8

Windows 8 is a personal computer operating system developed by Microsoft as part of the Windows NT family of operating systems. Windows 8 introduced major changes to

the operating system's platform and user interface, where Windows was now competing with mobile operating systems, including Android. Windows 8 added support for USB 3.0,Advanced Format hard drives, near field communications, and cloud computing. Additional security features were introduced, such as built-in antivirus software, integration with Microsoft SmartScreen phishing filtering service and support for UEFI Secure Boot on supported devices with UEFI firmware, to prevent malware from infecting the boot process.

Key features include:

- Speedy Boot Time
- Innovative Dynamic Desktop
- Improved Search Function
- Windows Live Syncing

Windows 8 was released to a mixed reception. Although reaction towards its performance improvements, security enhancements, and improved support for touchscreen devices was positive, the new user interface of the operating system was widely criticized for being potentially confusing and difficult to learn.

5.2.5 OPEN CV

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. The library is cross-platform. It focuses mainly on real-time image processing. If the library finds Intels Integrated Performance Primitives on the system, it will use these proprietary optimized routines to accelerate itself. OpenCV is the leading open source library for computer vision, image processing andmachine learning, and now features GPU acceleration for real-time operation.

OpenCV is released under a BSD license and hence its free for both academic and commercial use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications.

OpenCV is the most popular and advanced code library for Computer Vision related applications today, spanning from many very basic tasks (capture and pre-processing

of image data) to high-level algorithms (feature extraction, motion tracking, machine learning). It is free software and provides a rich API in C, C++, Java and Python. Other wrappers are available. The library itself is platform-independent and often used for real-time image processing and computer vision. Currently the library supports:

- real-time capture
- video file import
- basic image treatment (brightness, contrast, threshold)
- object detection (face, body)
- blob detection

5.3 HARDWARES USED

5.3.1 MICROCONTROLLER

A micro controller is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals. Program memory in the form of Ferroelectric RAM, NOR flash or OTP ROM is also often included on chip, as well as a typically small amount of RAM. Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general purpose applications.

Figure 5.4: ATmega328 Micro Chip

The ATmega328 shown in figure 5.4 is a single chip micro-controller created by Atmel and belongs to the megaAVR series [8]. The Atmel 8-bit AVR RISC-based microcontroller combines 32 KB ISP flash memory with read-while-write capabilities, 1 KB EEPROM, 2 KB SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible timer/counters with compare modes, internal and external interrupts, serial programmable USART, a byte-oriented 2-wire serial interface, SPI serial port, 6-channel 10-bit A/D converter (8-channels in TQFP and QFN/MLF packages), programmable watchdog timer with internal oscillator, and five software selectable power saving modes.

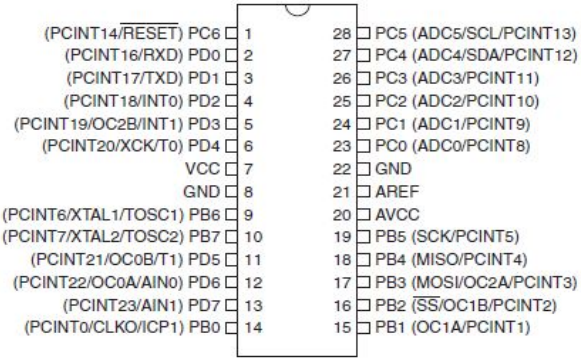

Figure 5.5: ATmega328 microcontroller pins

The device operates between 1.8-5.5 volts. The device achieves throughputs approaching 1 MIPS per MHz. The pin specifications are shown in figure 5.5.

5.3.2 COMPARATOR

A comparator is a device that compares two voltages or currents and outputs a digital signal indicating which is larger. It has two analog input terminals V+, and V-, and one binary digital output Vo.

$$V_o = \begin{cases} 1, & \text{if } V_+ > V_- \\ 0, & \text{if } V_+ < V_- \end{cases}$$

A comparator consists of a specialized high-gain differential amplifier that are used in devices like analog-to-digital converters (ADCs), as well as relaxation oscillators.

Figure 5.6: LM324 chip

LM324 a 14pin IC is shown in figure 5.6. It consists of four independent operational amplifiers (op-amps) compensated in a single package. Op-amps are high gain electronic voltage amplifier with differential input and, usually, a single-ended output. The output voltage is many times higher than the voltage difference between input terminals of an op-amp.

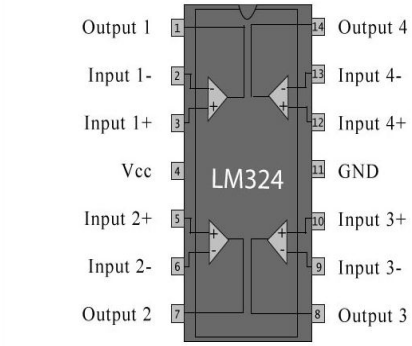

Figure 5.7: LM324 pin-out diagram

These op-amps are operated by a single power supply LM324 and need for a dual supply is eliminated. The complete pin-out of LM324 is shown in figure 5.7. They can be used as amplifiers, comparators, oscillators, rectifiers etc. The conventional op-amp applications can be more easily implemented with LM324.

5.3.3 HEXBUG SPIDER

This is a small walking robot that uses an Arduino nano an ultrasonic sensor for range finding. The HEXBUG Spiders shown in figure 5.8 have on/off switches on the

top of the main bug body which also doubles as A/B band select.

Figure 5.8: Hexbug Spider

Features include:

- The robotic creature that you control
- Full-range remote control with 2 bands:
- Dual band control means 2 separate Spiders can be controlled at the same time
- Assorted colors

5.3.4 SERVO MOTOR

Light weight 1/8 th shaft DC motors:

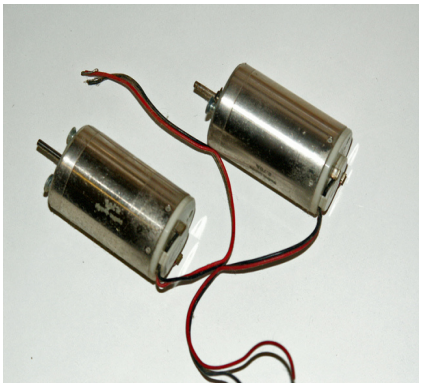

Figure 5.9: Light weight 1/8 th shaft DC motors

Figure 5.9 shows a Sagem 12-24v double ended BB motors with a stall of 10amps. They have an 8mm shaft with nice shaft encoders.

Key features includes:

- Operating supply voltage up to 6V
- The two red output shafts can be seen 6-9v

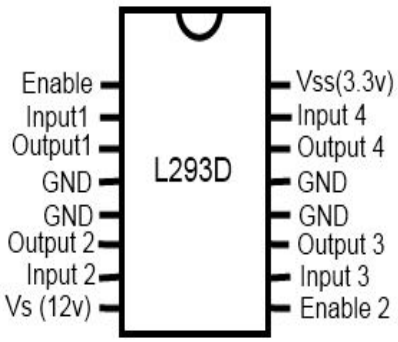

Figure 5.10: Dual H-bridge Motor Driver IC

Motor driver: L293D shown in figure 5.10 is a typical Motor driver or Motor Driver IC which allows DC motor to drive on either direction. L293D is a 16-pin IC which can control a set of two DC motors simultaneously in any direction. It means that you can control two DC motor with a single L293D IC.

Chapter 6
IMPLEMENTATION

6.1 INTRODUCTION

Computer vision is a field that includes methods for acquiring, processing, analyzing and understanding images. In the embedded world, computer vision applications have to fight with limited processing power and limited resources to achieve optimized algorithms and high performance. This thesis presents work on implementing a color tracking system on both the android platform and embedded systems to optimize the algorithms for high performance. The algorithms are benchmarked on the microprocessor chip. Functions and libraries in OpenCV was utilized for building the color tracking algorithms.

6.2 MODULES

6.2.1 IMAGE CAPTURE

This is an important module in the Android application. It does the job of accessing the android mobile camera and capturing images, so that it can provide a real time video, of the front view of the robot. This image stream is fed to the core application. Core application is responsible for handling this input for the next steps of processing.

6.2.2 COLOR-SPACE TRANSITION

Images received by the camera are transformed into RGB color space. To get a binary image out of the RGB image, color space is transformed into HSV so as to improve the result of color segmentation in different light intensities.

These three values can be thought of as coordinates of a point in three-dimensional space, giving rise to the concept of color space. Hue, saturation, and value (HSV) shown

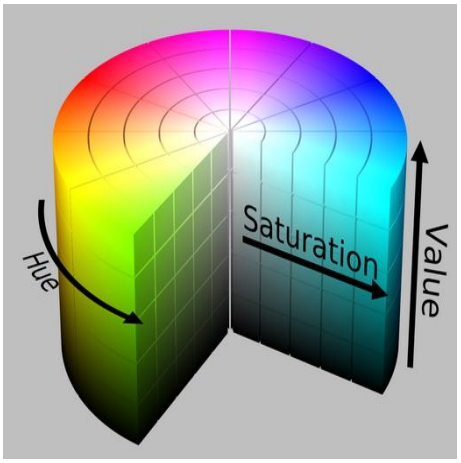

Figure 6.1: HSV Color Space

in figure 6.1 is one such color coordinate system, or color space, and is convenient for use in this application for a pair of reasons. Finally, the software implementation was performed.

6.2.3 FREQUENCY GENERATION

The captured image is further processed to concentrate on to the color of a particular image frame. The hue, saturation values are examined by the underlying android application platform to generate a unique frequency for each color. The application further processes the image and generates a frequency sound via the android application.

6.2.4 ANALOG TO DIGITAL CONVERSION

An analog-to-digital converter is a device that converts a continuous physical quantity to a digital number that represents the quantity's amplitude .The conversion involves quantization of the input. Instead of doing a single conversion, an ADC often performs the conversions periodically. The result is a sequence of digital values that have been converted from a continuous-time and continuous-amplitude analog signal to a discrete-time and discrete-amplitude digital signal.

An ADC may also provide an isolated measurement such as an electronic device that converts an input analog voltage or current to a digital number proportional to the magnitude of the voltage or current. The digital output may use different coding schemes.

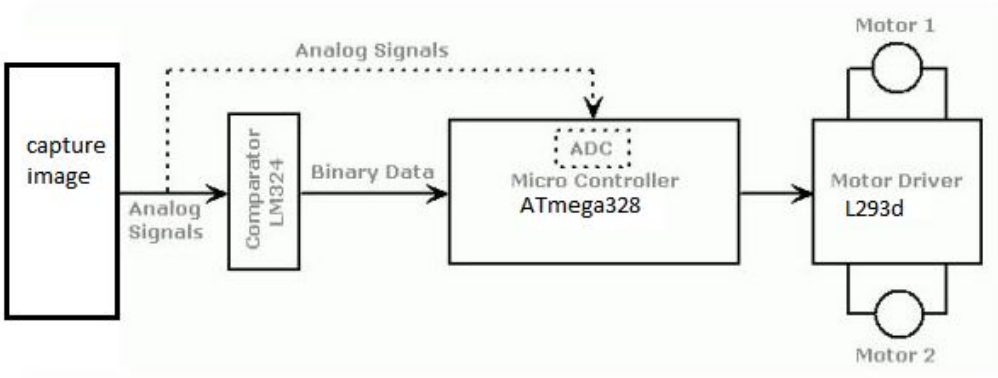

Figure 6.2: Implementation layout

Typically the digital output will be a two's complement binary number that is proportional to the input, but there are other possibilities.

It is clear that the output of the potential divider is an analog voltage. But Microcontroller does not accept the analog voltage. So we need to convert the analog voltage to digital before we feed it to the microcontroller. This conversion is carried out using ADCs as shown in figure 6.2. For the efficient functioning of the system there is necessity to convert the analog signal to digital signal. It make the system more fast and easy.

6.2.5 ROBOTIC NAVIGATION

Robot locomotion is the collective name for the various methods that robots use to transport themselves from place to place. Although wheeled robots are typically quite energy efficient and simple to control, other forms of locomotion may be more appropriate for a number of reasons such as traversing rough terrain, moving and interacting in human environments.

A major goal in this field is in developing capabilities for robots to autonomously decide how, when, and where to move. However, coordinating a large number of robot joints for even simple matters, like negotiating stairs, is difficult [9]. Autonomous robot locomotion is a major technological obstacle for many areas of robotics, such as humanoids. The generated frequency is used as an input to the microprocessor. Based on the frequency, microprocessor give instructions to the robot . Robot make corresponding movement to reach the target color.

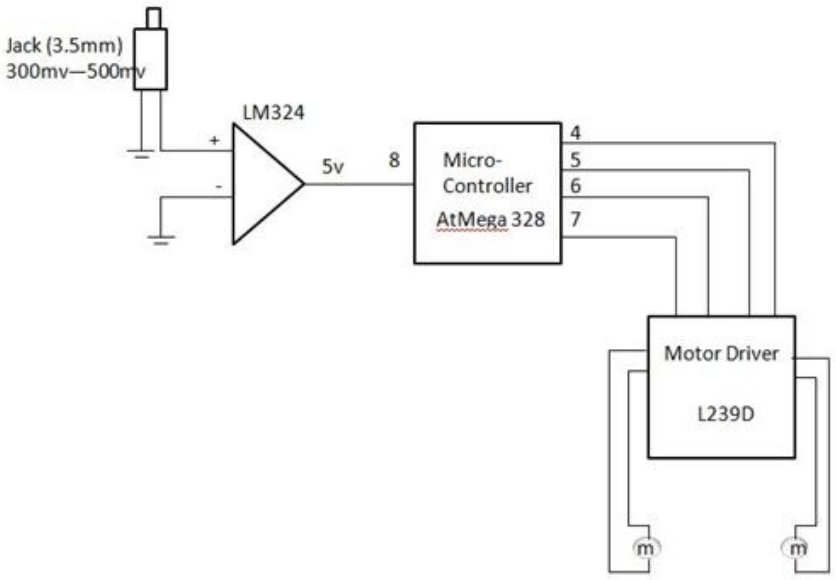

Figure 6.3: Block diagram

The implementation uses an android mobile camera with the tracking application. The android application is connected to the microcontroller ATmega328 via a LM324 comparator circuit.

Figure 6.3 represents the complete block diagram of the circuitry. The frequency generated by the application is fed to the LM324 comparator circuit via a jack of 3.5mm. It uses a voltage ranging from 300mv to 500mv. One connection is grounded and the other is connected to the positive terminal V+ of the LM324 comparator. The negative terminal of the comparator is again grounded. The output of the comparator circuit carries a 5v via the pin 8 of ATmega328 microcontroller. The outputs from the microcontroller are taken from the pins 4,5,6 and 7. These four outputs corresponds to the forward, backward , left and right movements of the hexbug spider. The microcontroller outputs are given to the motor driver chip L293D which controls the rotation of the two motors.

Chapter 7
TESTING AND MAINTENANCE

7.1 TESTING

Testing is the process of executing a program with the intent of detecting an error. Various tests were conducted in this project.It is an opportunity to show the users that this system works as expected.System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic.Software Testing is the design and implementation of a special kind of software system: one that exercises another software system with the intent of finding bugs.

7.1.1 UNIT TEST

In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use. The primary goal of unit testing is to take the smallest piece of testable software in the application, isolate it from the remainder of the code, and determine whether it behaves exactly as you expect. Each unit is tested separately before integrating them into modules to test the interfaces between modules. Unit testing has proven its value in that a large percentage of defects are identified during its use.

The most common approach to unit testing requires drivers and stubs to be written. The driver simulates a calling unit and the stub simulates a called unit [**?**]. The investment of developer time in this activity sometimes results in demoting unit testing to a lower level of priority and that is almost always a mistake. Even though the drivers and stubs cost time and money, unit testing provides some undeniable advantages. It allows for automation of the testing process, reduces difficulties of discovering errors contained in more complex pieces of the application, and test coverage is often enhanced because

attention is given to each unit.

In this test, different modules are tested against the specification developed during the design of modules.The hardware module were tested so as to ensure that the user get access to the software. The image capturing module was tested to ensure that the user can properly capture the image. The image converter module was tested in order to ensure that the image is successfully converted from analog to digital. The frequency generator module was tested to ensure that the accurate frequency of the color was generated. And finally the navigation module was tested to confirm that the robot was routed efficiently.

7.1.2 INTEGRATION TEST

Integration testing (sometimes called integration and testing, abbreviated IT) is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing.Integration testing tests integration or interfaces between components, interactions to different parts of the system such as an operating system, file system and hardware or interfaces between systems.

When the unit test was successfully conducted, the modules were linked as per the sequence of the forms. Integration testing is a systematic testing for constructing the program structure. All the modules were tested as whole to see whether the software and hardware interface (ie, the android application to the micro controller interface) worked efficiently.

7.1.3 BLACK BOX TEST

Black-box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. This method of test can be applied to virtually every level of software testing: unit, integration, system and acceptance. It typically comprises most if not all higher level testing, but can also dominate unit testing as well. This type of testing is based entirely on the software requirements and specifications.

Most likely this testing method is what most of tester actual perform and used the majority in the practical life. All testing is done as users point of view and tester is only aware of what is software is suppose to do but how these requests are processing by software is not aware. While testing, tester knows about the input and expected outputs of the software and they do not aware of how the software or application actually processing the input requests giving the outputs. Tester only passes valid as well as invalid inputs determines the correct expected outputs. All the test cases to test using such method are calculated based on requirements specifications document.The main purpose of the Black Box is to check whether the software is working as per expected in requirement document whether it is meeting the user expectations or not.

The softwares external working was tested in this case. The software module was interfaced with the line follower to test whether the robot traced the actual routing path. The feedback was collected from those experiments and the testing phase was ended up successfully.

7.2 MAINTENANCE

Maintenance can be defined as a set of activities that are performed to modify the system once the system is delivered to the customer.The system is already environment independent,so there is no need of adaptive maintenance.It is certainly preferable and cheaper to remove requirements error during analysis than after the system is deployed,because the error to be repaired in the copies installed at customer computers. Most of the errors have been detected and removed as a part of analysis.But even then if any error occur,then corrective maintenance is supported.Finally,improving,changing,or adding features and qualities to the application as required by the user is also possible in this product.

Chapter 8
CONCLUSION AND FUTURE WORK

8.1 CONCLUSION

With the increasing demands of tracking systems, color tracking systems has recently gained more interest. In this paper an algorithm is proposed to track the color of real-time objects on the basis on properties of different color spaces.

The development of mobile robot vision systems has also assured the great achievement in this field. We have implemented a color tracking system using the combined concepts of robotics and app development in order to recognize the color of an object. In order to improve the accuracy of identification in different illumination conditions an efficient color space is taken into account.

8.2 FUTURE WORK

We are planning to set up an effective system with more subjects and variations in conditions. In the proposed method, any object could be recognized based on the color patterns. This can be extended to face detection and gesture recognition by designing more sophisticated classifiers, segmentation techniques, and evaluation of different scenarios deserve more attention in future work.

REFERENCES

[1] M. Crnekovic, "Mobile robot vision system for object color tracking," *International Conference On Innovative Technologies*, vol. 4, December 2005.

[2] G. I.-G. T.Said, S.Ghoniemy, "Multi-object color tracking for multi-robot systems enhancement," *13 th International Conference On Aerospace Sciences and Aviation Technology*, vol. 2, May 2011.

[3] P. Fieguth, "Colour based tracking of heads and other mobile objects at video frame rates," *IEEE Trans.Auto.Control(24)*, vol. 5, November 1999.

[4] D. S. A.Agarwala, A.Hertzmann, "Keyframe-based tracking for rotoscoping and animation," *International Workshop On Intelligent Robots and Systems*, vol. 7, April 2010.

[5] S. R. Lee, "Fpga based color tracking for mobile robot system," *17th International Conference On Systems Signalling And Processing*, vol. 2, October 2010.

[6] B. Liu, "Color tracking vision system for the autonomous robot," *Information Science*, vol. 145, June 2002.

[7] R. A. Nuwan Gamage, "Towards robust skin colour detection and tracking," *in Graphicon Moscow State University*, vol. 1, April 2003.

[8] B. Sanjaa, "Real time embedded color tracking for motion robot system," *International Conference On Systems,Signals And Image Processing*, vol. 3, July 2010.

[9] D. Terzopoulos, "Color based probabilistic tracking," *IEEE Trans Auto Control24(6)*, vol. 3, May 2004.

[10] H.-Y. S. Yichen Wei Jian, Sun Xiaoou Tang, "Interactive offine tracking for color objects," *In computational learning theory*, vol. 5, January 2005.

Appendix A

Hardware Components Used

Figure 8.1: Experimental Setup

Figure 8.2: Hexbug Spider

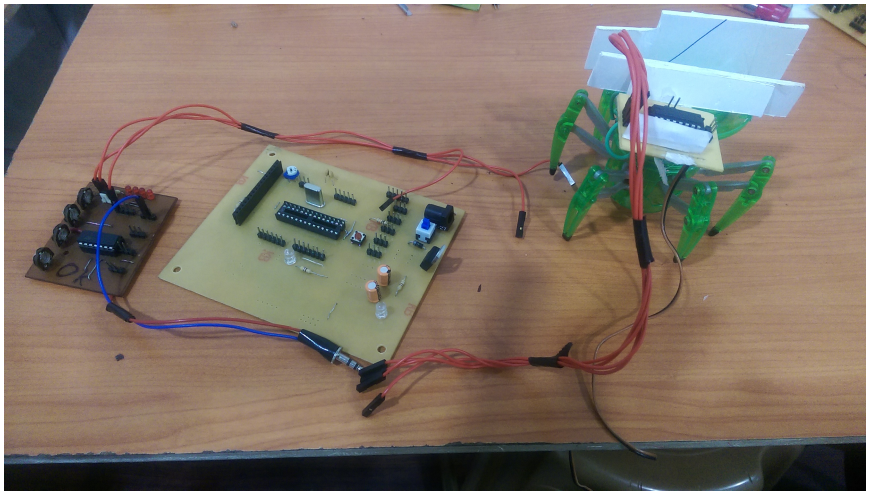

Figure 8.3: Circuit Components

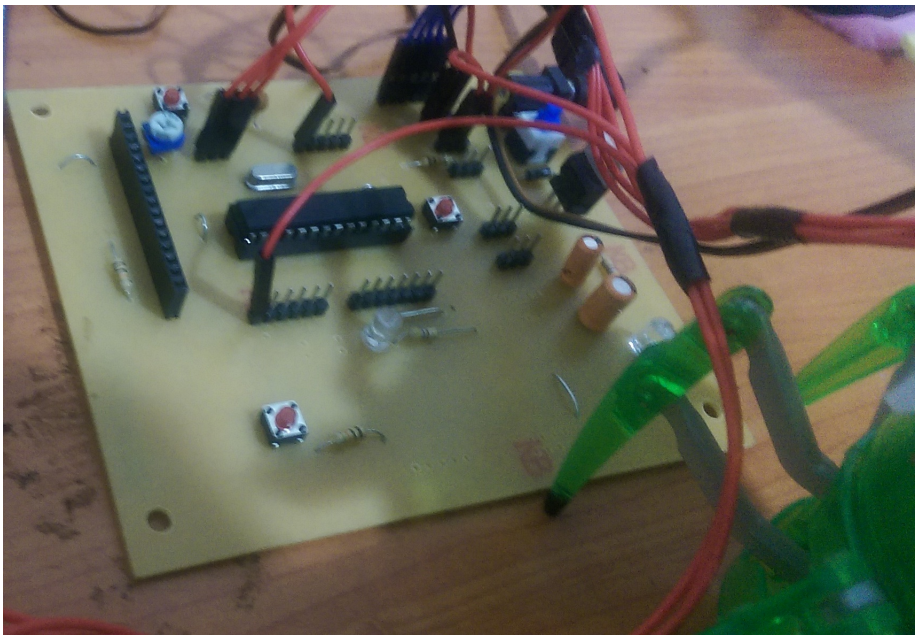

Figure 8.4: ATmega328 Circuit

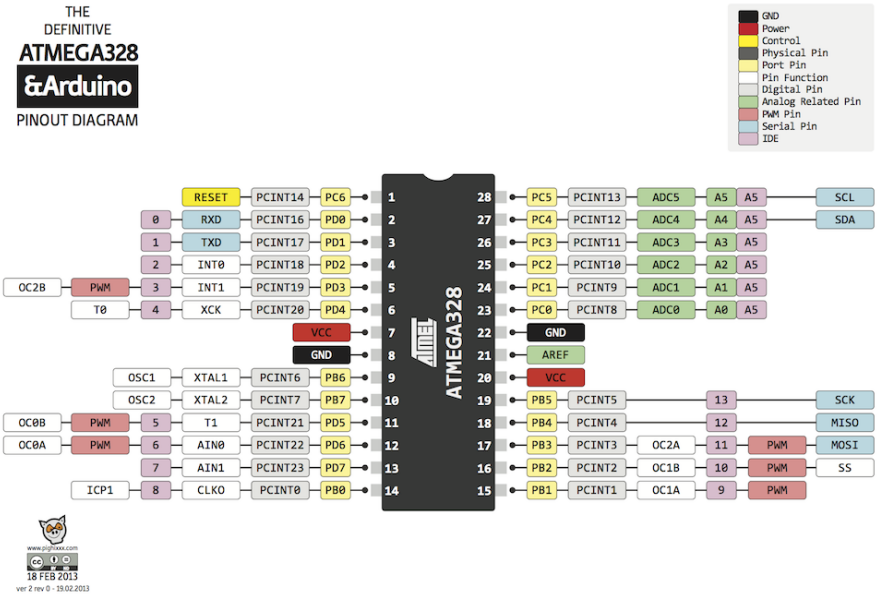

Figure 8.5: ATmega328 Pinout

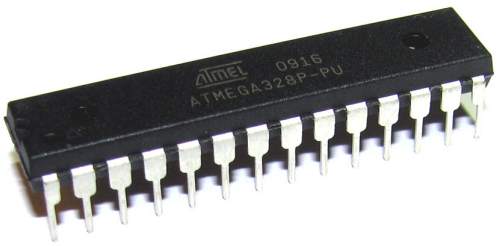

Figure 8.6: ATmega328

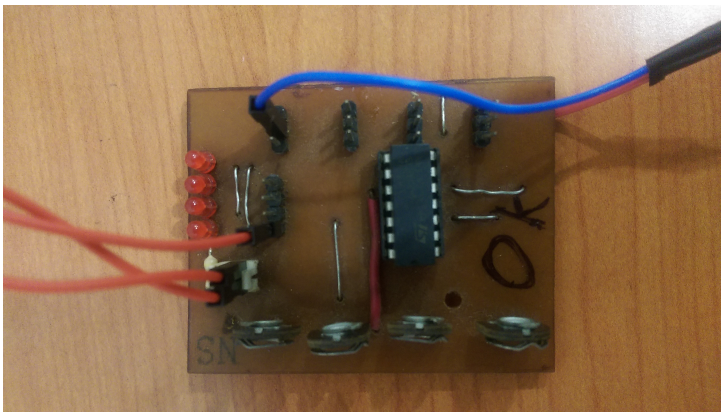

Figure 8.7: Comparator Circuit

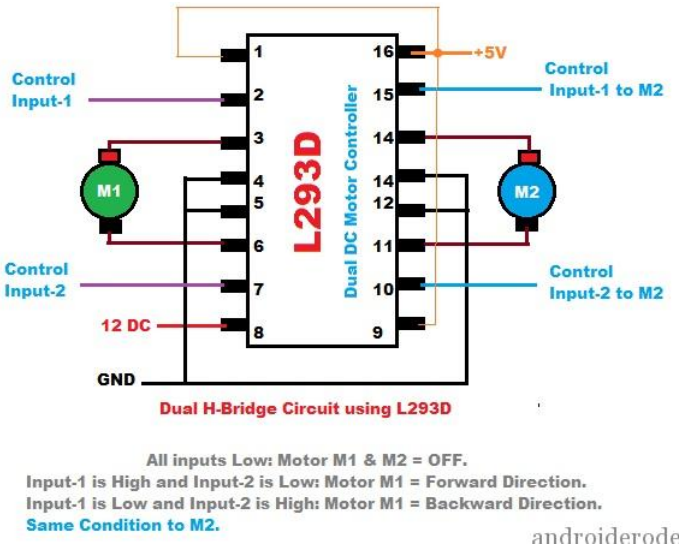

Figure 8.8: Dual H Bridge Circuit using L293D

Figure 8.9: L293D Dual H Bridge Motor Driver

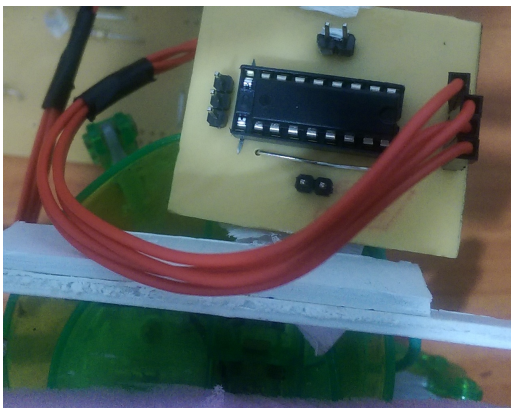

Figure 8.10: L293D Motor Driver Circuit

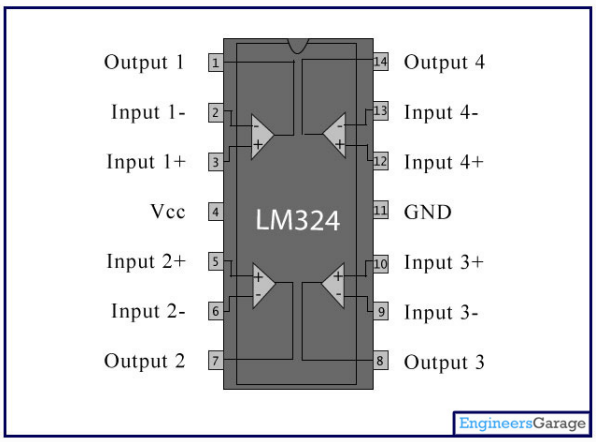

Figure 8.11: LM324 Pinout

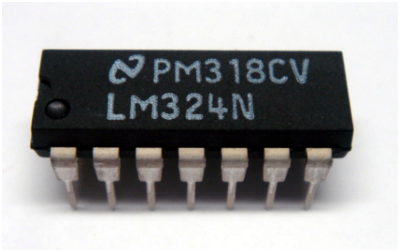

Figure 8.12: LM324N

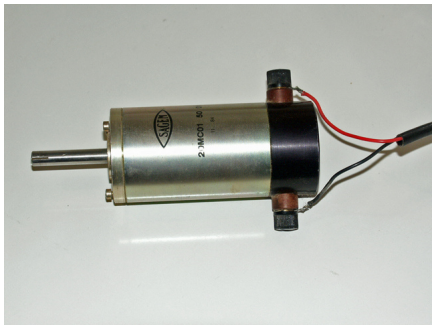

Figure 8.13: Sagem 12-24v Double Ended BB Motors

Appendix B

Main Codes

ARDUINO FREQUENCY CALCULATION

```
/* FreqMeasure - Example with serial output
 * http://www.pjrc.com/teensy/td_libs_FreqMeasure.html
 *
 * This example code is in the public domain.
 */
#include <FreqMeasure.h>

int motor11=4;
int motor12=5;
int motor21=6;
int motor22=7;

int led=13;

void setup() {
  Serial.begin(57600);
  FreqMeasure.begin();
  pinMode(motor11,OUTPUT);
  pinMode(motor12,OUTPUT);
  pinMode(motor21,OUTPUT);
  pinMode(motor22,OUTPUT);

  pinMode(led,OUTPUT);


}
double sum=0;
int count=0;

void loop() {

  if (!FreqMeasure.available())
  {
    Serial.println(count);
    Serial.println("STOP");
    digitalWrite(led,LOW);
    digitalWrite(motor11,HIGH); // STOP
    digitalWrite(motor12,HIGH);

    digitalWrite(motor21,HIGH);
    digitalWrite(motor22,HIGH);
  }
```

```
if (FreqMeasure.available())
{

  sum =  FreqMeasure.read();
  count=FreqMeasure.countToFrequency(sum);
  Serial.println(count);

  if(count<5900)
  {
    Serial.println(count);
    Serial.println("STOP");

    digitalWrite(motor11,HIGH); // STOP
    digitalWrite(motor12,HIGH);

    digitalWrite(motor21,HIGH);
    digitalWrite(motor22,HIGH);
  }

  if(count>9600)
  {
    Serial.println(count);
    Serial.println("stop");

    digitalWrite(motor11,HIGH); // STOP
    digitalWrite(motor12,HIGH);

    digitalWrite(motor21,HIGH);
    digitalWrite(motor22,HIGH);
  }

  if(count>7600 && count<8500)
  {
    //      if(count>12000 && count<12500)
    //  {
    Serial.println(count);
    Serial.println("forward");

    digitalWrite(led,HIGH);

    digitalWrite(motor11,LOW); // forward
    digitalWrite(motor12,HIGH);

    digitalWrite(motor21,HIGH);
    digitalWrite(motor22,HIGH);
  }
```

```
    if(count>6500 && count<7500)
    {
      //      if(count>11000 && count<11500)
      //{
      Serial.println(count);
      Serial.println("RIGHT");
      digitalWrite(led,LOW);
      digitalWrite(motor11,HIGH);
      digitalWrite(motor12,HIGH);

      digitalWrite(motor21,LOW);  //RIGHT
      digitalWrite(motor22,HIGH);
      delay(50);
      digitalWrite(motor21,HIGH);
      digitalWrite(motor22,HIGH);
      delay(50);
    }

    if(count>8600 && count<9500)
    {
      //            if(count>13000 && count<15500)
      //   {
      Serial.println(count);
      Serial.println("LEFT");

      digitalWrite(led,LOW);
      digitalWrite(motor11,HIGH);
      digitalWrite(motor12,HIGH);

      digitalWrite(motor21,HIGH);  //LEFT
      digitalWrite(motor22,LOW);
       delay(50);
      digitalWrite(motor21,HIGH);
      digitalWrite(motor22,HIGH);
      delay(50);
    }

    //        sum = sum + FreqMeasure.read();
    //     count = count + 1;
    //     if (count > 30) {
    //       float frequency = FreqMeasure.countToFrequency(sum /
    //       Serial.println(frequency);
    //       sum = 0;
    //       count = 0;
    //     }
  }
```

```
}
```

MAIN CODE

```
// Decompiled by Jad v1.5.8e. Copyright 2001 Pavel Kouznetsov.
// Jad home page: http://www.geocities.com/kpdus/jad.html
// Decompiler options: braces fieldsfirst space lnc

package colorbot.emgrobotics.com;

import android.app.Activity;
import android.app.AlertDialog;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

import colorbot.emgrobotics.DrawOnTop;
import colorbot.emgrobotics.ParmData;

// Referenced classes of package colorbot.emgrobotics.com:
//             ParmData, Preview, DrawOnTop

public class colorbot extends Activity
{

    public colorbot()
    {
    }

    public void onCreate(Bundle bundle)
    {
        super.onCreate(bundle);
        requestWindowFeature(1);
        ParmData.initParms();
        Preview preview = new Preview(this);
        DrawOnTop drawontop = new DrawOnTop(this);
        setContentView(preview);
        addContentView(drawontop, new android.view.ViewGroup.Lay
    }

    public boolean onCreateOptionsMenu(Menu menu)
    {
        menu.add(0, 0, 0, "Histogram");
        menu.add(0, 1, 0, "About");
        return super.onCreateOptionsMenu(menu);
    }
```

```
    public boolean onOptionsItemSelected(MenuItem menuitem)
    {
        super.onOptionsItemSelected(menuitem);
        switch (menuitem.getItemId())
        {
        default:
            return false;

        case 1: // '\001'
            AlertDialog alertdialog = (new android.app.AlertDialo
            alertdialog.setTitle("ColorBot");
            alertdialog.setMessage("Written by Eric Gregori\n (ww

            alertdialog.show();
            return true;

        case 0: // '\0'
            ParmData.setShowHistogram();
            return true;
        }
    }
}
```

HSV COLOR SYSTEM

```
/*
 * Decompiled with CFR 0_92.
 *
 * Could not load the following classes:
 *  java.lang.Object
 *  java.lang.String
 *  java.util.ArrayList
 *  java.util.Iterator
 *  java.util.List
 */
package colorbot.emgrobotics.com;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import org.opencv.core.Core;
import org.opencv.core.CvType;
import org.opencv.core.Mat;
import org.opencv.core.MatOfPoint;
import org.opencv.core.MatOfPoint2f;
import org.opencv.core.Point;
import org.opencv.core.Rect;
```

```
import org.opencv.core.RotatedRect;
import org.opencv.core.Scalar;
import org.opencv.core.Size;
import org.opencv.imgproc.Imgproc;

public class ColorBlobDetector {
    public static final int BLOB = 0;
    public static final int CIRCLE = 2;
    public static final int SQUARE = 1;
    private static final String TAG = "EMGRobotics::ColorBlobDet
    private Scalar CONTOUR_COLOR = new Scalar(255.0, 0.0, 0.0, 2
    private double ObjectArea;
    private Size SPECTRUM_SIZE = new Size(200.0, 64.0);
    private Scalar mBlobColorHsv = new Scalar(255.0);
    private Scalar mBlobColorRgba = new Scalar(255.0);
    private Scalar mColorRadius = new Scalar(10.0, 25.0, 25.0, 0
    private List<MatOfPoint> mContours = new ArrayList();
    Mat mDilatedMask = new Mat();
    private int mHeight = 0;
    Mat mHierarchy = new Mat();
    Mat mHsvMat = new Mat();
    private double mLastX = 0.0;
    private double mLastY = 0.0;
    private Scalar mLowerBound = new Scalar(0.0);
    Mat mMask = new Mat();
    private double mMinContourArea = 200.0;
    private int mMode;
    Mat mPyrDownMat = new Mat();
    private Mat mSpectrum = new Mat();
    private Scalar mUpperBound = new Scalar(0.0);
    private int mWidth = 0;

    public ColorBlobDetector(int n) {
        this.mMode = n;
        if (n == 2) {
            this.mColorRadius = new Scalar(25.0, 50.0, 50.0, 0.0)
        }
    }

    private Scalar converScalarHsv2Rgba(Scalar scalar) {
        Mat mat = new Mat();
        Imgproc.cvtColor(new Mat(1, 1, CvType.CV_8UC3, scalar), r
        return new Scalar(mat.get(0, 0));
    }

    public void SetParameters(int n, int n2) {
```

```
        this.mWidth = n2;
        this.mHeight = n;
    }

    /*
     * Enabled aggressive block sorting
     */
    public boolean Touch(Mat mat, int n, int n2, int n3, int n4)
        int n5 = mat.cols();
        int n6 = mat.rows();
        int n7 = (n3 - n5) / 2;
        int n8 = (n4 - n6) / 2;
        int n9 = n - n7;
        int n10 = n2 - n8;
        if (n9 < 0 || n10 < 0 || n9 > n5 || n10 > n6) {
            return false;
        }
        Rect rect = new Rect();
        int n11 = n9 > 4 ? n9 - 4 : 0;
        rect.x = n11;
        int n12 = n10 > 4 ? n10 - 4 : 0;
        rect.y = n12;
        int n13 = n9 + 4 < n5 ? n9 + 4 - rect.x : n5 - rect.x;
        rect.width = n13;
        int n14 = n10 + 4 < n6 ? n10 + 4 - rect.y : n6 - rect.y;
        rect.height = n14;
        Mat mat2 = mat.submat(rect);
        Mat mat3 = new Mat();
        Imgproc.cvtColor(mat2, mat3, 67);
        this.mBlobColorHsv = Core.sumElems(mat3);
        int n15 = rect.width * rect.height;
        int n16 = 0;
        do {
            if (n16 >= this.mBlobColorHsv.val.length) {
                this.mBlobColorRgba = this.converScalarHsv2Rgba(t
                this.setHsvColor(this.mBlobColorHsv);
                Imgproc.resize(this.getSpectrum(), this.mSpectrum
                mat2.release();
                mat3.release();
                return true;
            }
            double[] arrd = this.mBlobColorHsv.val;
            arrd[n16] = arrd[n16] / (double)n15;
            ++n16;
        } while (true);
    }
```

```
public List<MatOfPoint> getContours() {
    return this.mContours;
}

public double getObjectArea() {
    return this.ObjectArea;
}

public Mat getSpectrum() {
    return this.mSpectrum;
}

/*
 * Enabled aggressive block sorting
 */
public Scalar onFrame(Mat mat) {
    this.process(mat);
    List<MatOfPoint> list = this.getContours();
    double d = 0.0;
    double d2 = 0.0;
    if (list.size() > 0) {
        MatOfPoint matOfPoint = new MatOfPoint((Mat)list.get(
        int n = (int)matOfPoint.total();
        Point[] arrpoint = new Point[n];
        if (n > 0) {
            int[] arrn = new int[n * 2];
            matOfPoint.get(0, 0, arrn);
            int n2 = 0;
            do {
                if (n2 >= n) {
                    d = arrpoint[0].x;
                    d2 = arrpoint[0].y;
                    this.mLastX = d;
                    this.mLastY = d2;
                    break;
                }
                arrpoint[n2] = new Point(arrn[n2 * 2], arrn[1
                ++n2;
            } while (true);
        }
    }
    Imgproc.drawContours(mat, list, -1, this.CONTOUR_COLOR);
    mat.submat(4, 68, 4, 68).setTo(this.mBlobColorRgba);
    Mat mat2 = mat.submat(4, 4 + this.mSpectrum.rows(), 70, 7
    this.mSpectrum.copyTo(mat2);
```

```
            double d3 = this.getObjectArea();
            String string = "X=" + d + ", Y=" + d2 + " ,A=" + d3;
            Point point = new Point(80 + this.mSpectrum.cols(), 65.0
            Core.putText(mat, string, point, 1, 3.0, new Scalar(0.0,
            double d4 = this.mLastX;
            double d5 = this.mLastY;
            return new Scalar(d, d2, d4, d5);
        }

        /*
         * Unable to fully structure code
         * Enabled aggressive block sorting
         * Lifted jumps to return sites
         */
        public void process(Mat var1_1) {
            Imgproc.pyrDown(var1_1, this.mPyrDownMat);
            Imgproc.pyrDown(this.mPyrDownMat, this.mPyrDownMat);
            Imgproc.cvtColor(this.mPyrDownMat, this.mHsvMat, 67);
            Core.inRange(this.mHsvMat, this.mLowerBound, this.mUpperB
            Imgproc.dilate(this.mMask, this.mDilatedMask, new Mat());
            var2_2 = new ArrayList();
            Imgproc.findContours(this.mDilatedMask, var2_2, this.mHi
            var3_3 = var2_2.iterator();
            ** if (this.mMode != 1) goto lbl-1000
    lbl10: // 1 sources:
            do {
                if (!var3_3.hasNext()) lbl-1000: // 2 sources:
                {
                    if (this.mMode == 2) {
                        break;
                    } else {
                        ** GOTO lbl-1000
                    }
                }
                var17_4 = (MatOfPoint)var3_3.next();
                var18_6 = Imgproc.minAreaRect(new MatOfPoint2f(var17_
                var19_5 = Imgproc.contourArea(var17_4) / var18_6.size
                if (var19_5 >= 0.8 && var19_5 <= 1.2) continue;
                var3_3.remove();
            } while (true);
            do {
                if (!var3_3.hasNext()) lbl-1000: // 3 sources:
                {
                    var4_10 = this.mMinContourArea;
                    break;
                }
```

```
            var13_7 = (MatOfPoint)var3_3.next();
            if (var13_7.total() > 5) {
                var14_8 = Imgproc.fitEllipse(new MatOfPoint2f(var
                var15_9 = Imgproc.contourArea(var13_7) / var14_8.
                if (var15_9 >= 0.7 && var15_9 <= 1.25) continue;
                var3_3.remove();
                continue;
            }
            var3_3.remove();
        } while (true);
        var6_11 = var2_2.iterator();
        do {
            if (!var6_11.hasNext()) break;
            var7_12 = Imgproc.contourArea((MatOfPoint)var6_11.nex
            if (var7_12 <= var4_10) continue;
            var4_10 = var7_12;
        } while (true);
        this.ObjectArea = var4_10;
        this.mContours.clear();
        var9_13 = var2_2.iterator();
        do {
            if (var9_13.hasNext()) continue;
            return;
        } while (Imgproc.contourArea(var10_14 = (MatOfPoint)var9_
        Imgproc.boundingRect(var10_14);
        Core.multiply((Mat)var10_14, new Scalar(4.0, 4.0), (Mat)v
        this.mContours.add((Object)var10_14);
    }

    public void setColorRadius(Scalar scalar) {
        this.mColorRadius = scalar;
    }

    /*
     * Enabled aggressive block sorting
     */
    public void setHsvColor(Scalar scalar) {
        double d = scalar.val[0] >= this.mColorRadius.val[0] ? sc
        double d2 = scalar.val[0] + this.mColorRadius.val[0] <= 2
        this.mLowerBound.val[0] = d;
        this.mUpperBound.val[0] = d2;
        this.mLowerBound.val[1] = scalar.val[1] - this.mColorRad
        this.mUpperBound.val[1] = scalar.val[1] + this.mColorRad
        this.mLowerBound.val[2] = scalar.val[2] - this.mColorRad
        this.mUpperBound.val[2] = scalar.val[2] + this.mColorRad
        this.mLowerBound.val[3] = 0.0;
```

```
            this.mUpperBound.val[3] = 255.0;
            Mat mat = new Mat(1, (int)(d2 - d), CvType.CV_8UC3);
            int n = 0;
            do {
                if ((double)n >= d2 - d) {
                    Imgproc.cvtColor(mat, this.mSpectrum, 71, 4);
                    return;
                }
                byte[] arrby = new byte[]{(byte)(d + (double)n), -1,
                mat.put(0, n, arrby);
                ++n;
            } while (true);
        }

        public void setMinContourArea(double d) {
            this.mMinContourArea = d;
        }
    }
```

CAMERA PREVIEW

```
    /*
     * Decompiled with CFR 0_92.
     *
     * Could not load the following classes:
     *  android.content.Context
     *  android.hardware.Camera
     *  android.hardware.Camera$Parameters
     *  android.hardware.Camera$PreviewCallback
     *  android.view.SurfaceHolder
     *  android.view.SurfaceHolder$Callback
     *  android.view.SurfaceView
     *  java.io.IOException
     *  java.lang.Object
     */
    package colorbot.emgrobotics.com;

    import android.content.Context;
    import android.hardware.Camera;
    import android.view.SurfaceHolder;
    import android.view.SurfaceView;
    import colorbot.emgrobotics.ParmData;
    import java.io.IOException;


    class Preview
    extends SurfaceView
```

```
implements SurfaceHolder.Callback {
    int ProcessRate;
    Camera mCamera;
    SurfaceHolder mHolder;

    Preview(Context context) {
        super(context);
        this.mHolder = this.getHolder();
        this.mHolder.addCallback((SurfaceHolder.Callback)this);
        this.mHolder.setType(3);
        this.ProcessRate = 5;
    }

    public void surfaceChanged(SurfaceHolder surfaceHolder, int n
        Camera.Parameters parameters = this.mCamera.getParameters
        parameters.setPreviewSize(320, 240);
        this.mCamera.setParameters(parameters);
        this.mCamera.startPreview();
    }

    public void surfaceCreated(SurfaceHolder surfaceHolder) {
        this.mCamera = Camera.open();
        try {
            this.mCamera.setPreviewDisplay(surfaceHolder);
            this.mCamera.setPreviewCallback((Camera.PreviewCallba


                public void onPreviewFrame(byte[] arrby, Camera c
                    int n;
                    Preview preview = Preview.this;
                    preview.ProcessRate = n = preview.ProcessRate
                    if (n != 0) return;
                    int n2 = 76800;
                    int n3 = 0;
                    do {
                        if (n3 >= 19200) {
                            Preview.this.postInvalidate();
                            Preview.this.ProcessRate = 5;
                            return;
                        }
                        ParmData.updateStats(255 & arrby[n2], 255
                        n2+=2;
                        ++n3;
                    } while (true);
                }
            });
```

```
                return;
            }
            catch (IOException var2_2) {
                this.mCamera.release();
                this.mCamera = null;
                return;
            }
        }

        public void surfaceDestroyed(SurfaceHolder surfaceHolder) {
            this.mCamera.stopPreview();
            this.mCamera = null;
        }

    }
```

MOTION DETECTION

```
/*
 * Decompiled with CFR 0_92.
 *
 * Could not load the following classes:
 *  java.lang.Object
 */
package colorbot.emgrobotics.com;

import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.imgproc.Imgproc;

public class MotionDetector {
    private Mat mBackground = new Mat();
    private boolean mFirst = true;
    private Mat mGray = new Mat();
    private Mat mResult = new Mat();

    /*
     * Enabled aggressive block sorting
     */
    public void onFrame(Mat mat) {
        Imgproc.cvtColor(mat, this.mGray, 11);
        if (this.mFirst) {
            this.mGray.copyTo(this.mBackground);
            this.mGray.copyTo(this.mResult);
            this.mFirst = false;
        } else {
            Core.subtract(this.mGray, this.mBackground, this.mRe
```

```
                this.mGray.copyTo(this.mBackground);
            }
            Imgproc.cvtColor(this.mResult, mat, 9);
        }
    }
```