Bachelor Thesis

Controlling a Mindstorms Robot using Image Processing on Android Devices

Shasindran Poonudurai

Martikel-Nr: 700461

Studiengang Mechatronik

Fakültät Technik

Hochschule Reutlingen

Betreuer: Prof.-Dr. Stefan Mack

Prof.-Dr. Jens Weiland

Contents

List of Figures

List of Table

Acknowledgements

I would like to express my sincere thanks to my supervisor Prof.-Dr. Stefan Mack for his support and helpful guidance throughout all stages of the thesis work, as well as Prof.-Dr. Jens Weiland who was willing to supervise my thesis work. I take this opportunity to record my sincere thanks to all the faculty members of the Department of Mechatronics for their help and encouragements.

I also thank my family and friends for their unceasing encouragement and support. I also place on record, my sense of gratitude to one and all who, directly or indirectly, have lent their helping hand to successfully complete the thesis work.

Chapter 1:

Introduction

The goal of this thesis is to build an autonomous Robot which detects an object with the help of its sensors and Image Processing. Autonomous robots are intelligent machines capable of performing tasks by themselves without explicit human control. For this thesis an autonomous Robot can be built with fewer expanses in the form of LEGO Mindstorms.

The LEGO Mindstorms includes a programmable brick computer, a 32 bit microcontroller that controls the system. So far, it seems impossible to control the Lego Mindstorms robot via a camera with digital Image Processing because of the limited computational power of the microcontroller to capture and process the camera pictures. With the help of a PC inclusive a camera the controlling of a Mindstorms robot is seems possible where the PC captures and processes the Images and send the corresponding controlling command to the microcontroller. However, the size and the weight of the PC make this form of solution not suitable for a small autonomous robot.

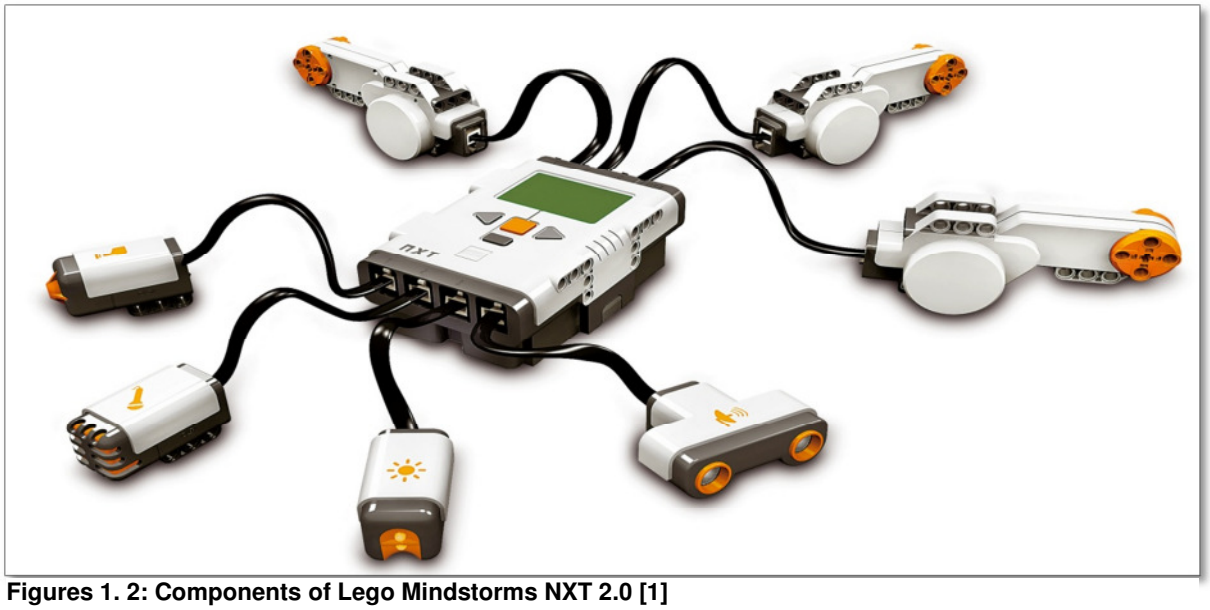
Nowadays the Smartphone have similar computational power and internal memory that of a PC. Besides the ever growing computational capacity of Smartphone, they come with Camera which is getting bigger and better. With these capabilities it seems very possible to control the Mindstorms Robot using Image Processing via Smartphone. Furthermore, both Smartphone and Mindstorms come with Bluetooth which makes easier to send the corresponding command from Smartphone to the microcontroller without wires and cables.

Lastly the Android Smartphone has an open Operating System which allows the developers to create or build hundred of applications. For this thesis an Android application is built with the help of OpenCV computer vision library to process the images. This way the Mindstorms NXT robot packed with the mobile phone becomes autonomous as it is controlled by the camera images.

Figures 1. 1: Components of LEGO Mindstorms NXT autonomous Robot

1.1 Lego Mindstorms NXT

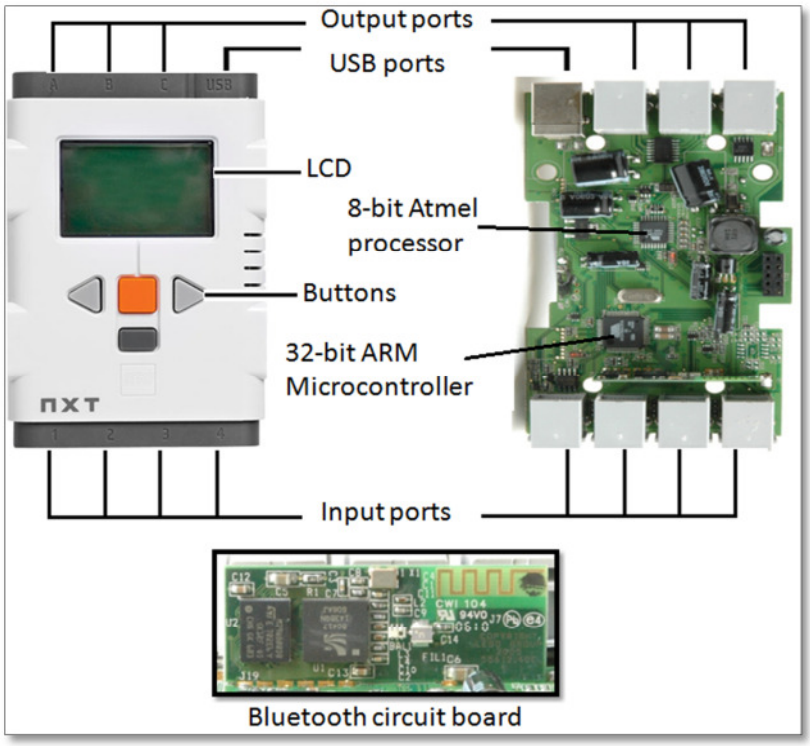
The Lego Mindstorms series of kits contains software and hardware to create small, programmable and customizable robot. In 2006 Lego released a new generation of programmable Mindstorms NXT robotic kits, replacing the first generation Lego Mindstorms kit. In this thesis Lego Mindstorms NXT 2.0 kit is used to build an autonomous robot. The details of the components and parts of the Mindstorms NXT are described in below:



Figures 1. 2: Components of Lego Mindstorms NXT 2.0 [1]

1.1.1 NXT Intelligent Brick

NXT Intelligent Brick is the main important part of the Lego Mindstorms robot. It is the brain of the robot which has a 32 bit ARM processor running at 48MHZ, with 64KB of RAM and 256KB of flash memory. It also contains a separate 8 bit AVR processor running at 8MHZ to control the servo motors and rotation sensors to guarantee the accuracy of the motor operations.



The brick comes with three motor ports labeled A, B, and C and four sensor ports labeled S1, S2, S3 and S4. It has a small monochrome LCD screen together with four buttons to allow users to make various choices. It also contains a Bluetooth circuit board to support the Bluetooth functionality.

Figures 1. 3: NXT Intelligent Brick and Components

1.1.2 The Motors

For the Lego Mindstorms new servo motors are developed. The Servo Motors have a built-in rotation sensor that measures speed and distance and reports back to NXT Intelligent Brick. This makes the brick possible to control the motor in a closed loop manner within one degree of accuracy.

Figures 1. 4: Structure of NXT Servomotor [1]

1.1.3 The Sensors

Besides the NXT Intelligent Brick and the motor they are several sensors provided by Lego. [1]

Touch Sensor

Touch sensor is the most basic sensor with a simple switch which acts similar to a button.

Light Sensor

This sensor measures the intensity of light, visible or not, that enters its tiny “eye”. Also comes with an LED which lights up in front of the sensor so it can operate in active (LED on) or passive (LED off) mode. If the LED is used it will act as a light switch. Without LED it works as a light level sensor. In both cases the sensor uses an analog output.

Ultrasonic sensor

Ultrasonic sensor produces a sonar cone to measure the distance between it and an object. It can measure up to 255 cm, and is accurate between 6 to 180 cm. It uses an I^2C Sensor-Interface to the NXT Brick.

Sound sensor

Sound sensor measures the loudness of sound in decibels. It can be used for example to detect clapping. This sensor also uses an anolog interface to the NXT brick.

Color Sensor

The color sensor enables the robot to detect not only between black and white, and also range of 6 colors. It also can detect both reflected and ambient light.

Besides these sensors, there are several other sensors that users can purchase such as compass sensor, color sensor, acceleration sensor and etc which are manufactured by third party companies [2].In addition there are several hundred parts that users can use to build concrete robots.

Lego is preparing for the release of third generation of Lego Mindstorms in summer 2013. The new generation kit is called Mindstorms EV3; EV stands for “Evolution”. Just like NXT 2.0 the EV3 version will have the same programmable brick that have features and possibilities that were not available in NXT 2.0. There will be an infrared sensor attached to the intelligent brick and a SD slot that will allow the user to store more information. The EV3 also have the ability to work with Android and Apple Smartphone devices to permit the user to control the robot. All the building instructions can be accessed through tablet or mobile devices. Robotic building with EV3 is now simpler and more convenient than before. [3]

1.2 Image Processing for Mobile Robots

In the past decade, video tracking, surveillance systems and robotic fields that have been well studied. However, in the most of the surveillance and video tracking systems the sensors are stationary, which requires the desired object to stay within the surveillance range. If the object goes beyond this range, it no longer becomes tractable. The solution for this problem is to design the system as a mobile system that uses sensors and a camera, to track the object and avoid obstacles. In this thesis work, the autonomous robot should detect an object and avoid obstacles on its own. Therefore it is a good idea to place or mount a camera on the robot to detect the desired object with image processing.

Image processing applications have become extremely essential in robotics over the last 15 years. Image processing is any form of signal processing for which the input is an image, such as a photograph or video frame; the output of image processing may be either an image or a set of characteristics or parameters related to the image. The output for such image is depends on the tasks such as navigation, orientation, object and surrounding identification.[5]

The navigation system plays very important role in controlling the mobile robots. There is a variety of theories and technologies such odometry technique, ultrasonic mapping and vision system for navigating a mobile robot. In this thesis we concentrate particularly on vision system.[4] Controlling using vision approach has many tasks, including target matching, target identification and others, which can be solve using image processing methods like canny edge detectors, circle detection, color detection and so on.

There are many image processing software available for mobile robots. However, some heavy software such as MATLAB image processing toolbox is not suitable for small mobile robot. The idea is to make it possible to do image processing on android based Smartphone, which has computational power equivalent of a PC. Moreover, the Smartphone come with cameras, which fed the Smartphone with images to process them in real-time. We will go through the possibility of this idea in next few subchapters.

1.3 Android and Smartphone

A Smartphone is a cellular telephone that able to perform many basic functions of a personal computer. Today Smartphone are able to perform many tasks that only could have done in a personal computer 10 years ago such as Internet access, creating a document and many others. Smartphone usually incorporate advanced PDA (Personal Digital Assistant) functions, large screens and data entry and text typing methods. Smartphone has been evolved year by year to much better device not only as a cellular phone but also as a device to take images, a digital video recorder, a Bluetooth and WLAN(Wireless Local Area Network) capable device ,GPS(Global Positioning System) and consists of many internal sensors. As an example at the moment you may find

- Motion sensor
- Position sensor
- Magnetic field sensor
- Light sensor
- Proximity sensor
- Touch sensor
- GPS-Receiver

in a single Smartphone device. Therefore it is easier for developer to create their applications abb. Apps (Apps are user programs for mobile devices) using these available sensors, so that the user able to interact with the Smartphone comfortably. Almost all Smartphone-Operating systems (OS) has access to every single sensors and functions; make it possible to do Image Processing using integrated camera, control the display brightness using light sensors, type the input data on the touch screen using touch sensor and many more.

In chapter 1.3.1 we will discuss about every available Smartphone OS and the decision to choose an Android platform for this thesis.

1.3.1 Mobile Operating System (OS)

Much like the Linux or Windows operating systems control our desktop or laptop computers, a mobile operating system is the software platform on top of which other programs can run on mobile devices. OS for Smartphone such as Symbian (Nokia), Palm OS (Palm), and Windows Mobile (Windows) und BlackBerry OS (Research in Motion) are losing their Market mainly as a result of new OS such as

- Android (Google)
- iOS (Apple)
- Windows (Microsoft)

The first two operating systems concur the market with Android (Q4 2012:70.1%) leading the iOS (Q4 2012: 21%) [6]. Finally, the Android OS is chosen because the Android is an Open Source platform und its distribution relatively high at the moment. As a result of Open Source, Android has a big user community in Internet and it is easier to learn the steps for creating an Android Apps as they are several tutorials and guides available in Internet. Furthermore, Android comes with open development platform, which permits to create application for free under Windows, Mac OS und Linux. The advantage of Android is the disadvantage of iOS. To develop an iOS Apps one need a Developer Account, for which the developers have to pay yearly, and a Mac OS computer. In addition, if a developer wants to create an iOS App under Windows, he/she needs to buy software, through which Apps can be created. With these arguments in mind, an existing Android Smartphone is chosen for this thesis to avoid extra costs.

1.3.2 Smartphone Hardware

For this thesis work an existing Android Smartphone, Samsung Galaxy S Wi-Fi 4.0 is used. The Galaxy S Wi-Fi 4.0 Smartphone has no telecommunication unity. The Smartphone is unable to call or SMS because there is no SIM-Card slot available. These functions are not required in this thesis, so we do not have to worry about it.

Samsung galaxy S Wi-Fi has following connectivity capabilities:

- Wi.Fi(802.11 b/g/n)
- Bluetooth v3.0 (A2DP/AVRCP)
- USB 2.0

It has a 1 GHz processor and an internal memory of 8GB, extensible to 32 GB via MicroSD. There is a 3.2 megapixel camera at the rear and a VGA camera on the front and a 4'' Display with WVGA Resolution (800x480). The 1200mAh battery can last up to 36 hours of music playback or 5 hours of continuous video playback. Important for this thesis is the camera, which captures images and feed the Smartphone to do Image Processing. The Apps is transferred via USB Interface from computer to Smartphone. The communication between Smartphone and NXT Brick works with Bluetooth connection. This topic is discussed in chapter 2.0.

1.3.3 Development Environment for Apps

In this chapter we will discuss about the IDEs for the development of the Apps. IDE stands for Integrated Development Environment or Interactive Development Environment. An IDE, a software application that provides facilities to computer programmers for software development, normally contains a source code editor, a Debugger, a Linker and a Compiler. Several modern IDEs integrate with Intelli-sense (intelligent sense) coding features, which attempts to speed up the process of coding applications by reducing misunderstandings, typos and other mistakes that programmers often make.

In this thesis the IDE Eclipse is used to create Android applications. Eclipse is a multi-language IDE compromising a base workspace and an extensible plug-in System for customizing the environment. It is mostly written in Java programming language. Eclipse software development kit is free and open source software, which is the ideal option for software development. For the application development under Android 2.3 is the Tegra Android Development Pack from the company Nvidia installed. This development pack consists of:

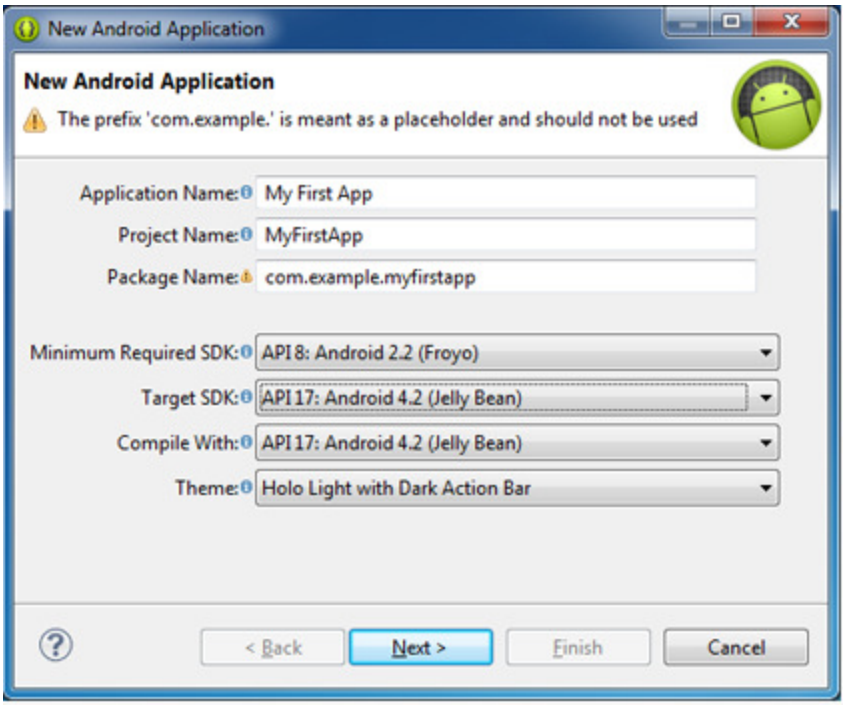
- Eclipse 4.2.1
- Android API (Application Programming Interface)
- Android SDK (Software Development Kit) r21.0.1
- Android ADT (Android Development Tools)
- Android support library
- Android Virtual Device (AVD)
- OpenCV for Tegra 2.4.3
- Google USB Driver

and can be downloaded and installed under the Website *https://developer.nvidia.com/tegra-android-development-pack*. With Eclipse one can test his/her Java-Program in an Android Emulator on a virtual Android Device with the help of Android Virtual Device (AVD). In this thesis however the AVD feature was not used, because the cameras as well as the sensors of the robot are evidently not available in a virtual device. In addition to that developers can test their application directly with an Android Device, where the application is transferred from the computer to the Android device through USB connection. For this purpose the Android device must first set in USB-debugging mode, which can be activated under "Settings"→ "Application" → "Development".

1.3.4 Creating an Android Application Project

This chapter shows how to create a new project using Eclipse for the Android application development. If a developer decides to program an Android-App with Java using Eclipse IDE, he/she must first create a new Project. A new project can be created under Menu 'File' →'New'→'Project'. In this case we choose the project type as 'Android Project' and fill in the window that appears next as in figure 1.5:

- ***Application Name*** is the app name that appears to users.
- ***Project Name*** is the name of the project directory and the name visible in Eclipse.
- ***Package Name*** is the package namespace of the App (following the same rules as packages in the Java programming language).



Figures 1. 5: The Android App Project wizard in Eclipse.

- ***Minimum Required SDK*** is the lowest version of Android that the Apps supports. To support many devices as possible, one should set this to the lowest version available that allows your Apps to provide its core feature set. The Samsung Galaxy S supports the API version until API 10.
- ***Target SDK*** indicates the highest version of Android, with which one test the application.
- ***Compile With*** is the platform version against which one will compile the Apps. By default this is set to the latest version of Android available in the SDK.
- ***Theme*** specifies the Android UI style to apply the Apps. You can leave this alone.

The next step is to configure the project, which can be set to default selections. The following steps are to create a launcher icon and select activity templates, for this project a Blank Activity can be selected. Now the project is set up with some default files and we are ready to begin building the app [7]. Note that through these steps the User Interface is defined via xml-data. However the User Interface in this thesis is created using Java Code, so that the Project has less complexity. Before we can write an App, we must first and foremost understand what an Activity is and it's Lifecycle. The following chapter helps you to understand the activity lifecycle.

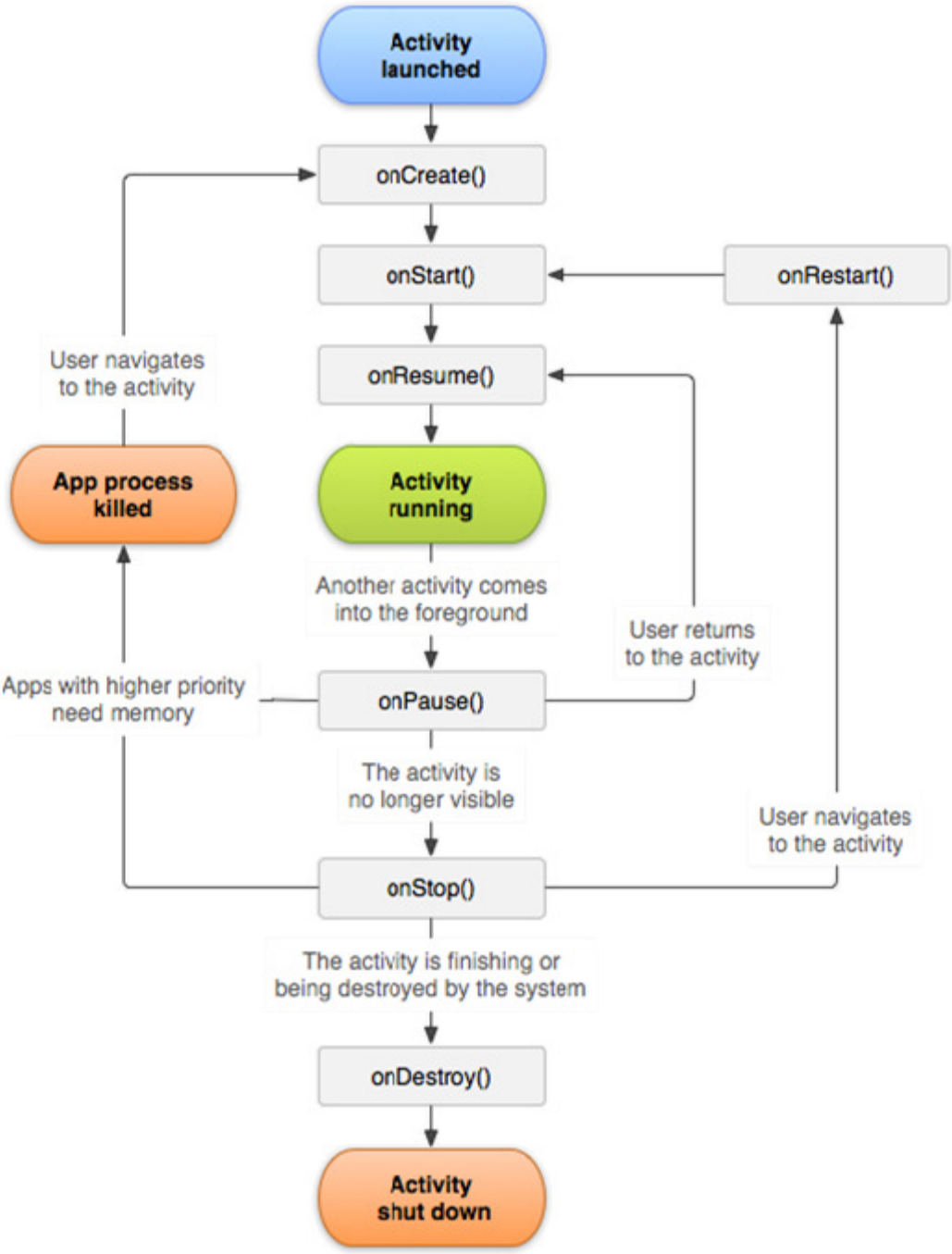
1.3.5 Android Activity Lifecycle

Any Android application must adapt to the Android OS's concepts of application lifecycle. Applications that is written without coding for application lifecycle can be less responsive, less stable and less compatible than an Apps that handles simple lifecycle events. An activity is a single, focused thing that a user can do. Because it implements all User Interface and views in an Android application, the Activity class is considered as the most visible to the users. Activity is often presented to the user as full-screen window. Compare to a Windows OS, where many windows can run simultaneously, in Android only one Window or Activity runs or visible to the user. Activities are managed as an activity stack. As soon as a new Activity is started, it is located on the top of the stack and becomes the current running activity, where the previous activity remains below it in the stack and will not come to the foreground until the new activity exits. [8]

An Activity has four states:

- Active or running, if an activity in the foreground of the screen
- Paused, if an activity lost focus but still visible (a new not full sized activity has focus on top of the activity). The activity is alive, which means it maintains all state and member information and remains attached to the window manager. However the activity can be killed by the system in low memory scenario.
- Stopped, if an activity is completely hidden by another activity. It still holds all state and member information, but it is not visible so its window is hidden and it will often killed by the system when memory is needed.
- If an activity is paused or stopped, the system can drop the activity from memory by killing its process. To display it again to the user, it must be completely restarted and restored to its previous state.

Figure 1.6 shows the important state paths of an Activity. The rectangles represent the callback methods one can implement to perform operations when the Activity moves between states. The colored ovals are major states the Activity can be in.



Figures 1. 6: The Activity Lifecycle of an Android application [8]

The whole lifecycle of an Activity is described by the following Activity methods. All these methods are override-able members of Activity that a developer can use to do appropriate work when the activity changes state. [8]

onCreate(): called when the activity is first created to set up the Java class or the instance of the Apps. Always followed by onStart ().

onStart(): called when the activity is becoming visible to the user. Followed by onResume () if the activity comes to foreground or onStop ().it becomes hidden.

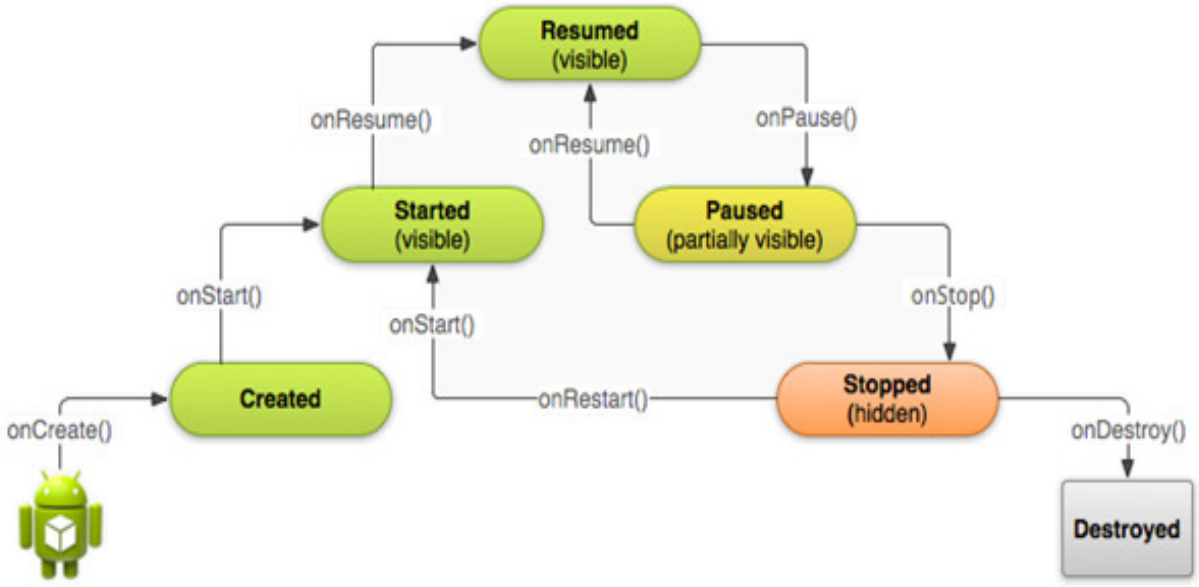
onResume(): called when the activity will start interacting with the user.

onPause(): the App is losing its foreground state and the system is about to start resuming a previous activity. Followed by onResume () or Onstop ()

onStop(): the end of the current visible lifespan of the Apps. This may happen either because a new activity is being started, an existing one is being brought in front of this one, or this one is being destroyed.

onRestart(): called after the activity has been stopped. Followed by onStart ().

onDestroy(): called when the Java class is about to be destroyed. Once this function is called, there is only one option for transition: onCreate ().



Figures 1. 7: An illustration of the Activity lifecycle, expressed as a step pyramid. [9]

During the life of an activity, the system calls a set of lifecycle methods in a sequence similar to a step pyramid which means each stage of the activity lifecycle is a separate step on the pyramid. As soon as the system creates a new activity instance, each callback method moves the activity state one step toward the top. The top of the pyramid is the point at which the activity is running in the foreground and the user can interact with it. [9]

As the user starts to exit the activity, the system calls other methods that move the activity state back down the pyramid in order to disassemble or destroy the activity. In some scenarios, the activity will move only part way down the pyramid to allow the user resume from where he/she left. The activity calls some specific method such as onResume() or onRestart() to move back the activity to the top of the pyramid. The whole events of callback method are illustrated in figure 1.7.

1.4 OpenCV

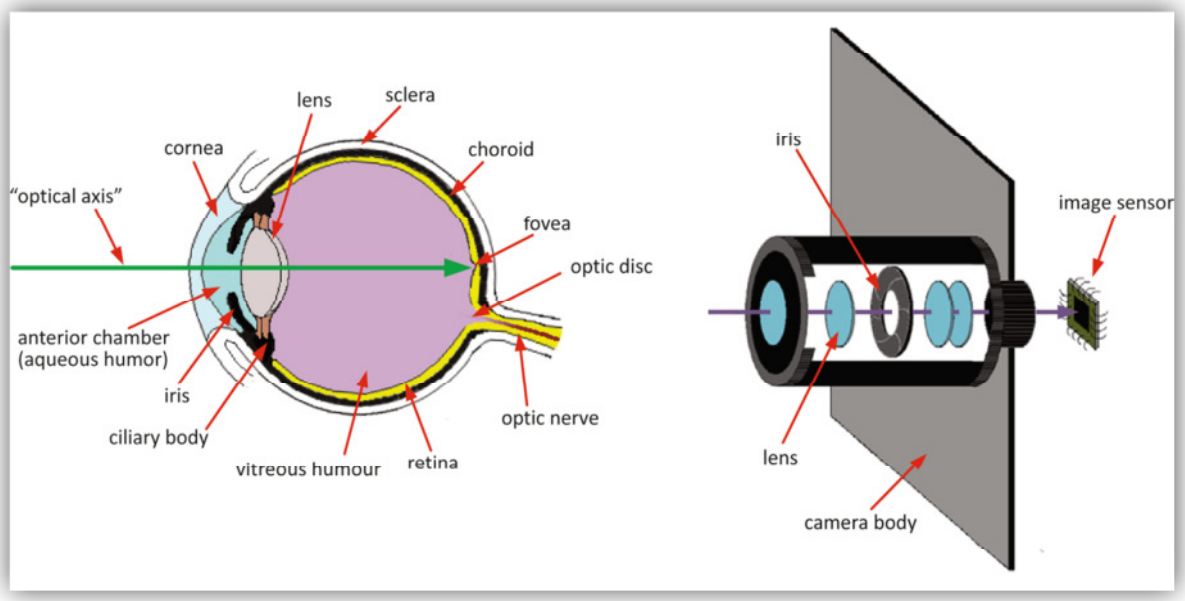
Computer vision is a fast growing field, partly as result of cheaper and more affordable camera, partly because of affordable processing power and partly because of the growth of vision algorithm. OpenCV itself has contributed significantly in the growth of computer vision by allowing thousands of developers to do more productive work in vision. OpenCV, which its focus on real time vision helps students and professionals implements projects and research by giving them with a computer vision and machine learning infrastructure. In this chapter we will go through what computer vision is all about and in what way does OpenCV contributes to computer vision field.

1.4.1 Computer Vision

Vision is our most powerful sense, which provides us with an outstanding amount of information about our surroundings and makes it possible for us to interact intelligently with environment without any direct physical contacts. Through vision we learn the identities and positions of objects and the relationships between them. It is considerably a disadvantage if we do not have this sense. Several successful attempts have been made to give machines a sense of vision with the help of cameras.

Computer vision or machine vision is the science and technology of making machines that see. It is concerned with the theory, design and implementation of algorithms that can automatically process visual data to recognize objects, track and recover their shape [10]. Because we are such a visual creatures, it is easy to be mistaken into thinking that computer vision tasks are easy. Since human biological vision systems are complex, it is not surprising that many attempts to provide machines a vision have ended in failure.

To understand the process of computer vision we have to understand first and foremost the human's biological vision. Human's vision starts when the lens of eye focuses an image of its surroundings onto retina, which is a part of brain that is isolated to serve as a transducer for the conversion of lights into neuronal signals. The human brain then split the vision signals into many channels that stream different kinds of information into our brain. Our brain identifies important parts of an image to examine according to the task. The feedback loops in the brain go back to all stages of processing including the hardware sensors, the eye, which mechanically control lighting via iris and tune the reception on the surface of the retina.

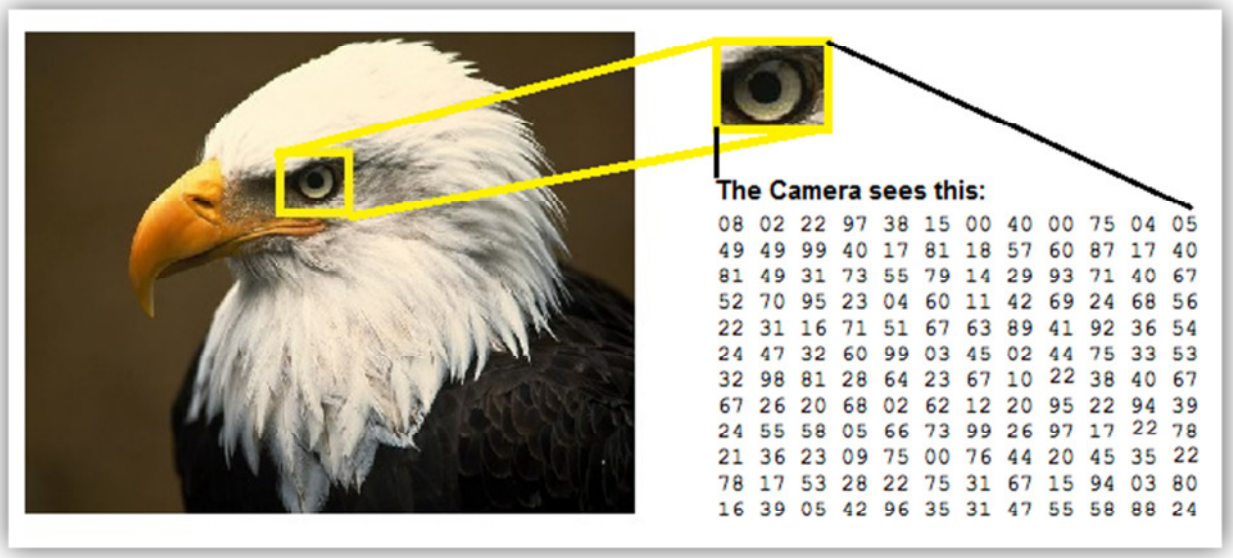


Figures 1. 8: Comparison between a human's eye and a digital camera. [12]

In computer vision function digital cameras as the eye of human's biological vision system. Although more and better digital cameras are available, there are, no digital camera comparable with a human's eye. This is mainly the result of the digital camera's inability to capture an image or view the surroundings in 3-dimensional and the camera has higher chance of noise level in dark environments. Figure 1.8 shows the comparison of a human's eye and its digital counterparts, a digital camera. As we can see the camera only resembles the important components of a human's eye, the lens, the iris and the retina (image sensor of a camera).

In a computer vision system, however, what a computer receives from the camera or from disk is a grid of numbers. For the most part, there's no automatic control of

focus and aperture and no built-in-pattern recognition. Figure 1.9 shows a picture of an eagle. In this image we see an eye of the eagle. What a computer “sees” is just a grid of numbers of the eagle’s eye. Any number within the grid has a large noise component, but this grid of numbers is what all the computer or camera “sees” [11]. The next task is to convert this noisy grid of numbers into the perception: “eye of an eagle”.



Figures 1. 9: To a computer, the eagle’s eye is just a grid of numbers.

The decisions or actions that computer vision attempts to make are depends on the camera data, which are performed according to specific task or purpose. As an example, we may want to remove noise and distortion from an image so that our security system will give an alert if someone tries to climb a fence. In this kind of situation OpenCV definitely contribute a little bit by providing the developer its functions in the library to remove the noise and damages.

1.4.2 What is OpenCV?

Open Source Computer Vision (OpenCV) is an open source [13] computer programming library available to support applications that use computer vision. The library is written originally in C and now in C++ and runs under Linux, Windows and Mac OS X. There is active development on interfaces for Matlab, Phyton, Java and other languages.

Launched officially in 1999, OpenCV was initially an Intel Research initiative to advance CPU-intensive applications, part of a series of projects including real-time ray tracing and 3D display walls. A number of optimization experts in Intel Russia, as well as Intel's Performance Library Team are the main contributors to the OpenCV project. The first alpha-version of OpenCV was released in 2000 followed by five betas between 2001 to 2005. In 2008 OpenCV got the corporative support from Willow Garage, a robotics research lab and technology incubator devoted to developing hardware and open source software for personal robotics applications.[14]. Today OpenCV is taken over by a non-profit foundation, OpenCV.org, which maintains a developer and user sites [11], [13].

OpenCV is intended to provide the basic tools needed to solve computer vision problems. In some situations, high-level functionalities in the library will be sufficient to solve more complex problems in computer vision. The basic components in the library are complete enough to create a complete solution of own to almost any computer vision problem.

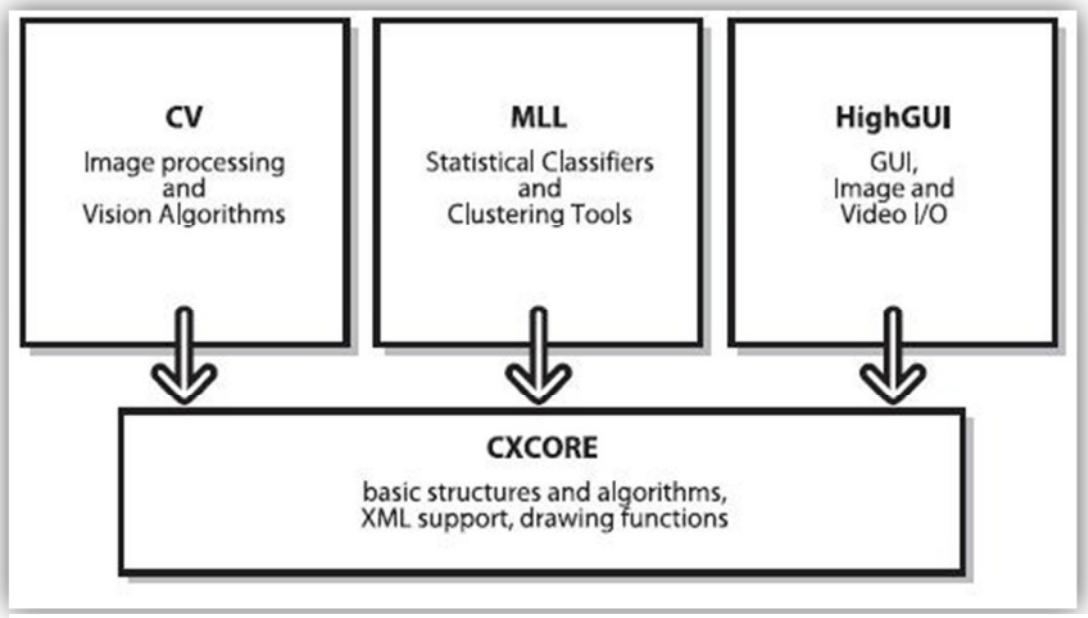
OpenCV was designed for computational efficiency and with a strong focus on real time applications. OpenCV library contains hundreds of functions that support the capture, analysis, and manipulation of visual information that given to a computer by cameras, webcams or other types of devices. The OpenCV library functions are used in many areas in vision, including factory product inspection, medical imaging, security, user interfaces, camera calibration, stereo vision and robotics.

With the help of OpenCV, a developer can avoid some of the complex and tedious work when creating an application. For example rather than creating algorithms for facial recognition, a developer can add just few lines of code to access the OpenCV

library function which contains code for face recognition. In this way a programmer does not need to master every aspect of computer vision to build an application.

1.4.3 OpenCV Structure

OpenCV is structured into five main components, four of which are shown in Figure 1.10.



Figures 1. 10: The basic structure of OpenCV [11]

CV component contains the basic image processing and higher-level computer vision algorithms. The MLL is machine learning library, which includes many statistical classifiers and clustering tools. HighGUI contains I/O routines and functions for storing and loading video and images. CXCore contains the basic data structures and content. Another component is CvAux, which contains both defunct areas (embedded HMM face recognition) and experimental algorithms (background/foreground segmentation) [11].

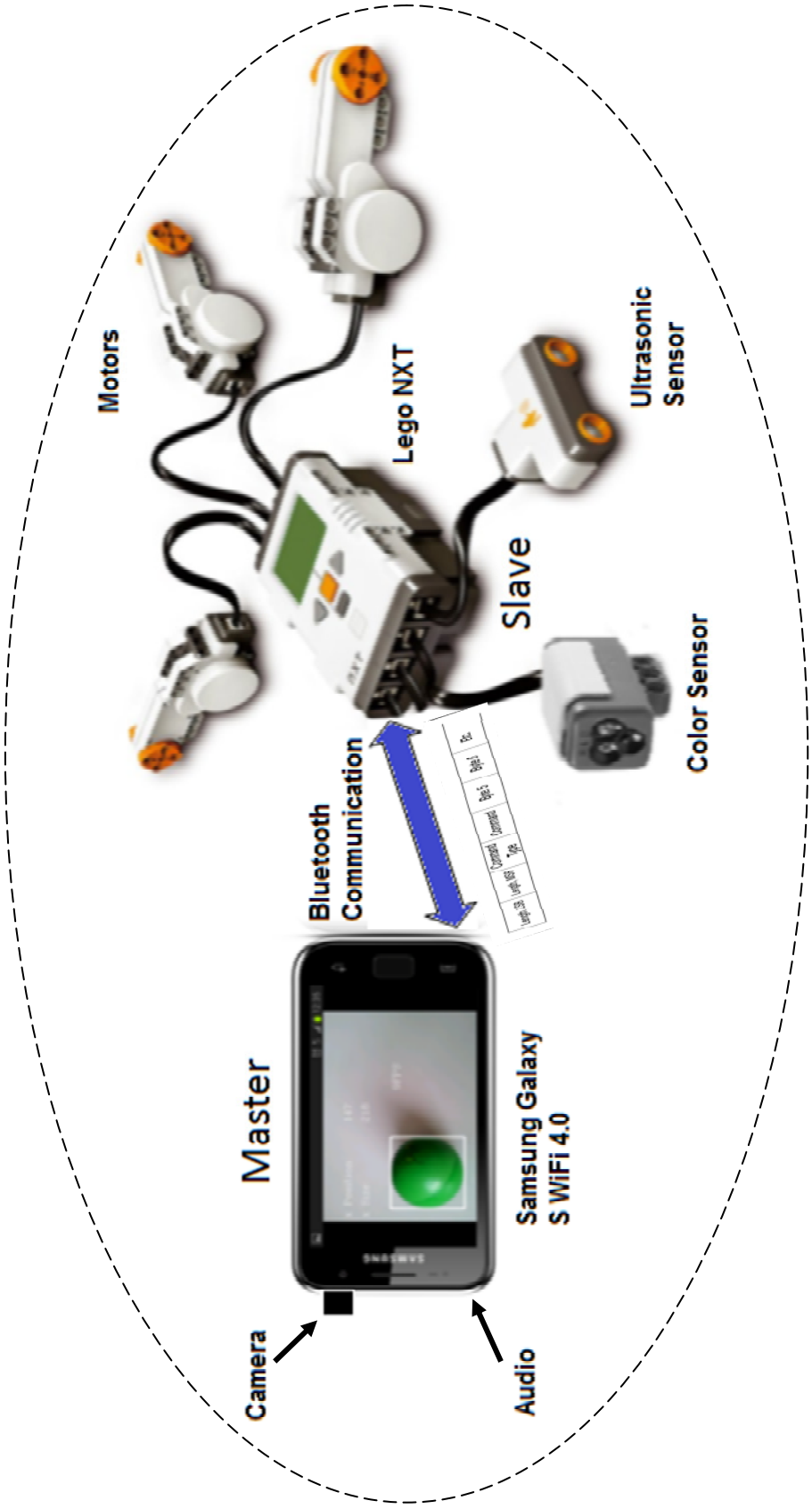
1.5 System Design

Once all three components namely Lego Mindstorms, Android and OpenCV are well familiarized an autonomous Robot is built combining all three components. The Android based Smartphone is attached or placed on the robot. Figure 1.11 shows the system design of an autonomous Lego Mindstorms robot.

The Smartphone captures images through its integrated camera and communicates with the user by playing audio tone via its speaker. With the help of OpenCV, an Android Apps performs image processing on the captured images and extracts information.

Both the Smartphone and Lego Mindstorms have Bluetooth functionalities. So to avoid excessive wires, the Bluetooth communication is used to exchange the messages between Lego Mindstorms and the Smartphone. All the controlling methods are performed on the Smartphone platform and the Lego NXT receives massages via Bluetooth from the Smartphone and reacts according to the messages. In this case the Smartphone behaves as the Master, whereas the Lego NXT behaves as the slave. The Bluetooth communication is explained in chapter 2.0.

Out of 3 Servomotors, 2 servomotors are used to drive the robot and the remaining motor is used to grab the detected object. An ultrasonic sensor is used to measure distance and NXT sent the values to the Smartphone to control the robot via Collision Avoidance method. An additional color sensor is used to indicate the object is there for grabbing. The controlling methods are explained well in chapter 4.0.



Figures 1. 11: Sytem Design

1.6 Example Applications outside NXT-Projects.

The ultimate goal of this thesis work is to demonstrate the importance of image processing not only in robotic field but also in other filed such as security and surveillance and etc. Furthermore, there are several projects and applications were created to use image processing on the Smartphone platform in our daily routines.

One of the example applications of image processing on the Smartphone platform is the barcode recognition. There are several iPhone, Android Apps such as the 1D and 2D Barcode Scanners which use image processing to extract the information. Furthermore, there is a thesis work done to create an application which reads a 2D color Barcode.[29]. In that application the camera captures the barcode image in JPEG format and the Smartphone analyses the image to extract the information.

Another example is the image recognition as a visual museum guide application for the open source Android based Smartphone. When a user takes a picture of a painting, the application searches related information in a database using image recognition [30].

Another interesting example is a life saving application for visually impaired persons during an emergency which is done by a student. In that application a Symbian OS Smartphone (also applicable to Android and iPhone Smartphone) detects an emergency exit signs using the phone's camera. In case of detection, the Smartphone indicates this through an acoustic signal and if an arrow is present on the sign, the application specifies the direction through text output [31].

There are many applications such as above examples demonstrate the important and relevant of image processing on Smartphone platform.

Chapter 2:

2.0 Lego NXT System

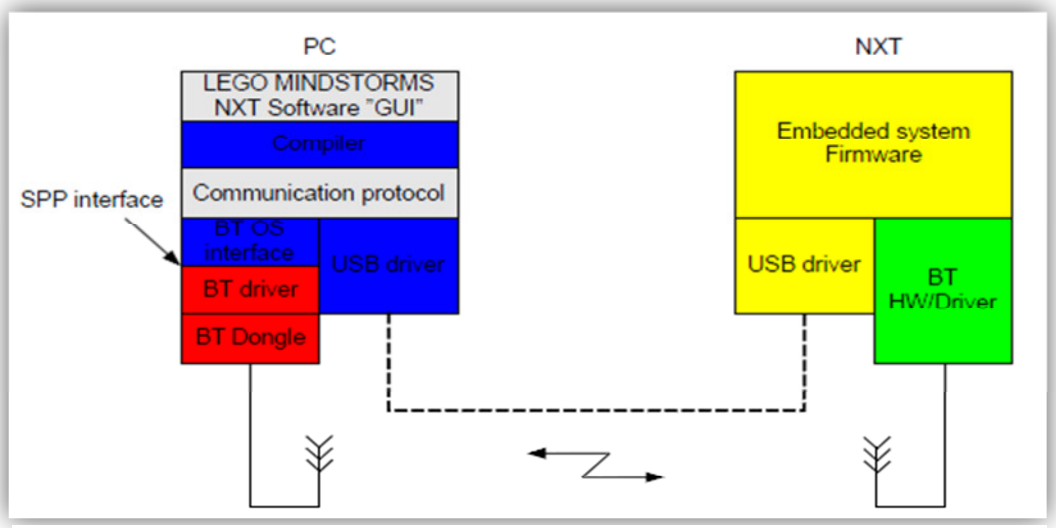
In this chapter we'll look closely at connecting a Lego Mindstorms NXT robot to a remote device, Samsung Galaxy S (Android Smartphone), via a Bluetooth wireless communication. The NXT robot supports a communications protocol known as Direct Commands, allowing it to be controlled like a slave by a remote device. In this usage mode the NXT Intelligent Brick acts only as an interface between the Smartphone and the NXT Sensors/Actuators. In the following two subchapters Bluetooth communication and direct commands are explained in details.

2.1 Communication via LCP commands

There are two possibilities in Lego Communication Protocol (LCP):

- Bluetooth communication, V2.0 with EDR
    - Supporting the Serial Port Profile(SPP)
- USB communication, V2.0

The figure below shows the main layers in the communication stack between the



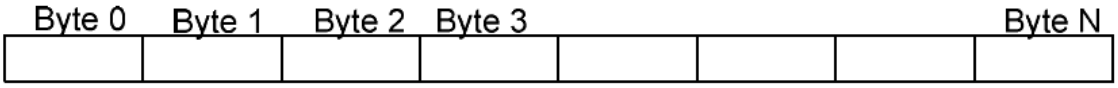
Figures 2. 1: Communication block diagram.[15]

embedded device and the software on the PC or Smartphone. As shown in the figure 2.1, it is possible to communicate between the two devices in two different ways, using wireless or using a wire.

Since we want to connect the NXT robot with android device via Bluetooth communication, it is better if we study a little bit about direct commands. Direct commands are a sub-protocol of the main LCP which make it possible to control the NXT robot from outside devices. These outside devices may be other NXT bricks, a PC or any other Bluetooth devices, in our case Samsung Galaxy S Smartphone. The objective of this sub-protocol is to enable a simple interface for these outside devices to utilize the basic brick functionality such as motor control, sensor readings and power management, without the necessity to write or run specialized remote control programs on the brick.[15]

2.1.1 General protocol architecture

It is possible to control the NXT brick either through the USB communication or through the Bluetooth communication channel, which both uses the LCP.



Figures 2. 2: General protocol architecture[15]

The figure 2.2 shows the general telegram architecture:

Byte 0: Telegram type, as dictated by main LCP specification.

- 0x00: Direct command telegram, response required
- 0x01: System command telegram, response required
- 0x02: Reply telegram
- 0x80: Direct command telegram. no response
- 0x81: System command telegram, no response

Byte 1-N: the command itself or a reply, depending on telegram type

Total direct command telegram size is limited to 64 bytes, including the telegram type byte as listed above. : An example of direct command is explained as below:

SETOUTPUTSTATE

Byte 0: 0x00 or 0x80
Byte 1: 0x04
Byte 2: Output port
Byte 3: Power set point
Byte 4: Mode byte
Byte 5: Regulation mode
Byte 6: Turn Ratio
Byte 7: Run state
Byte 8-12: TachoLimit

Return package:
Byte 0: 0x02
Byte 1: 0x04
Byte 2: status Byte

This direct command is for driving the motors of the NXT robot. As mentioned above Byte 0 can be set either with response or without response. Meanwhile Byte 1 indicates the order of the command in direct command list. As for response package a status byte is included, where 0x00 means “success” and any non zero value indicates a specific error condition.

2.2 Bluetooth Connection with Lego NXT

The topic of Bluetooth communications is much broader and deeper than we can hope to cover in single sub-chapter. The aim of this sub-chapter is to explore Bluetooth connection between an Android platform and Lego NXT brick.

What is Bluetooth?

Bluetooth is a wireless technology similar to Wi-Fi but constrained to exchange data over short distances approximately 100 meters. In addition Bluetooth enables peer to

peer network access, object exchange, cable replacement and advanced audio/media capabilities. The Bluetooth capability of interest to us is the cable replacement functionality of the Serial Port Profile (SPP), which is referred to as RFCOMM. The RF stands for radio frequency and the COMM stands for communications port.

The Bluetooth works in a similar fashion to other communications environment where there is a primary or master device that initiates communications with one or more secondary or slave devices. In this theses work the Android device is the master that initiate communications with the slave NXT brick, where an Android application exchanges data through a socket interface.

Although, the Bluetooth allows for both encrypted and unencrypted communications between peer devices, the Android platform permits only encrypted connections. This means the two Bluetooth devices must first be paired or bonded. It is recommended to look for documents or (book) for more details how to bond two devices through Bluetooth.

To access the Bluetooth on an Android device, one need to use the *android.bluetooth* package, which is first appeared in Android version 2.0. Table 2.1

Class	Comment
BluetoothAdapter	This class represents the local Android device's Bluetooth hardware and interface constructs. Everything begins with the BluetoothAdapter.
BluetoothClass	This class enables a convenient means of accessing constant values related to Bluetooth communications and operations.
BluetoothDevice	Any remote device is represented as a BluetoothDevice.
BluetoothSocket	This class is used for exchanging data. A primary device to connect initiates a socket connection with a secondary device by first creating a BlurtoothSocket.
BluetoothServerSocket	A Bluetooth secondary device listens for a primary device to connect through a BluetoothserverSocket. In much the same ways that a web server awaits a TCP socket connection from a browser.

Table 2.1: Android Bluetooth Java Classes [16]

shows the major Java classes used by Bluetooth-enabled Android applications. [16]

Working with a paired device peer isn't the only place where permissions come into play. An Android application must have the BLUETOOTH permission defined in the AndroidManifest.xml file:

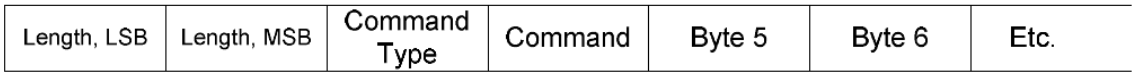
<uses-permission android:name="*android.permission.BLUETOOTH*"/>

The Bluetooth Message

The Bluetooth Serial Port communication has some disadvantages when it comes to streaming data. It is the result of bigger data packages, where the Bluecore chip waits to determine the length of the message and there is a time penalty of around 30mS within the Bluecore chip when switching from receive mode to transit mode. [15]. Bluecore is a self-contained implementation of BT that manages the BT hardware and protocols. [17]

To help reduce the problem some additional data and functionality is added to the NXT communication protocol during Bluetooth communication. This additional functionality only used when using Bluetooth communication.

To handle the problem of determining the size of the data that is being received, length bytes should be added in front of the NXT communication protocol. These two bytes indicates the total length of the data packages. The length of the packages is counted without the two length bytes. By inserting these two bytes the chip does not wait to determine the length. The figure below shows a Bluetooth message:



Figures 2. 3: Bluetooth protocol package[15]

To minimize the problem of time penalty, it is recommended to send data using Bluetooth without requesting a reply package. This means that the chip would not have to switch direction for every received package.
Below is a method written to set the output state of the motor via Bluetooth:

```
public void MoveMotor(int motor, int speed) {
            try {
```

```
                byte[] buffer = new byte[14];

                buffer[0] = (byte) (14 - 2); // length LSB
                buffer[1] = 0; // length MSB
                buffer[2] = 0x80; // direct command (without response)
                buffer[3] = 0x04; // set output state
                buffer[4] = (byte) motor; // output 1 (motor B)
                buffer[5] = (byte) speed; // power
                buffer[6] = 1 + 2; // motor on + brake between PWM
                buffer[7] = 0; // regulation
                buffer[8] = 0; // turn ration??
                buffer[9] = 0x20; // run state
                buffer[10] = 0;
                buffer[11] = 0;
                buffer[12] = 0;
                buffer[13] = 0;

                outpStr.write(buffer);
                outpStr.flush();
                }

        } catch (Exception e) {
                Log.e(TAG, "Error in MoveForward(" + e.getMessage() + ")");
        }
}
```

This code performs the simple yet precise operation of formatting a command, which is sent to NXT robot to provide direct control over the motors. First and foremost a buffer of the appropriate size is declared according to the SetOutputState command. Then each of various data elements are carefully wrote in their respective locations. As discussed above the buffer 0 and buffer 1 are the two bytes needed in a Bluetooth communication. Once the command buffer is formatted, it's written and flushed to the socket.

Now let's have a look at the parameters to set the input state of an Ultrasonic sensor. Below is a method written using direct command to set the Ultrasonic sensor in active mode:

```
public void setUSSensor() {

        try {
                Log.i(TAG, "getUSDistance(), SETINPUTMODE");
                // US Sensor (SETINPUTMODE)
                byte[] buffer = new byte[7];

                buffer[0] = (byte) 0x05; // length lsb
                buffer[1] = (byte) 0x00; // length msb
                buffer[2] = (byte) 0x80; // direct command (without response)
                buffer[3] = (byte) 0x05; // SETINPUTMODE
```

```
                buffer[4] = (byte) 0x00; // Input port
                buffer[5] = (byte) 0x0B; // Sensor type (LOWSPEED_9V)
                buffer[6] = (byte) 0x00; // Sensor mode (RAWMODE)

                outpStr.write(buffer);
                outpStr.flush();

                Thread.sleep(100);

            } catch (Exception e) {
                Log.e(TAG, "Error in getUSDistance, SETINPUTMODE(" +
e.getMessage() + ")");
            }
      }
```

There is no much difference compare to the output state method as both are direct commands. It is important to call this method before calling other methods to measure the distance using the Ultrasonic sensor. Failing to do so will not activate the Ultrasonic sensor. Another important thing is that, there is a special communication protocol to read the values of the NXT ultrasonic sensor, because the Ultrasonic sensor differs from other sensor in terms of using I²C Communication [15]. To read the values the direct commands and I²C communication protocol must followed as shown in the method below:

```
public int getUSDistance() {
      int distance = 0;
      try {
            Log.i(TAG, "getUSDistance(), LSWRITE_ReadByte0");
            // I²C write command,
            byte[] buffer = new byte[9];

            buffer[0] = (byte) 0x07; // length lsb
            buffer[1] = (byte) 0x00; // length msb
            buffer[2] = (byte) 0x80; // direct command (without response)
            buffer[3] = (byte) 0x0F; // LSWRITE
            buffer[4] = (byte) 0x00; // Input port
            buffer[5] = (byte) 0x02; // Tx Data Length
            buffer[6] = (byte) 0x08; // RX Data Length (8 Echomesswerte)
            buffer[7] = (byte) 0x02; // Tx Data Byte 0
            buffer[8] = (byte) 0x42; // Tx Data Byte 1

            outpStr.write(buffer);
            outpStr.flush();

            Thread.sleep(100);

      } catch (Exception e) {
            Log.e(TAG, "Error in getUSDistance(), LSWRITE_ReadByte0(" +
e.getMessage() + ")");
      }

      try {
            Log.i(TAG, "getUSDistance(), LSRead");
            // I²C read command
```

1

2

```
            byte[] buffer = new byte[5];

            buffer[0] = (byte) 0x03; // length lsb
            buffer[1] = (byte) 0x00; // length msb
            buffer[2] = (byte) 0x00; // direct command (with response)
            buffer[3] = (byte) 0x10; // LSRead
            buffer[4] = (byte) 0x00; // Input port

            outpStr.write(buffer);
            outpStr.flush();

            Thread.sleep(100);

            byte response[] = ReadResponse(16);
            if (response == null) {
                  Log.e(TAG, "No LSRead response??");
            } else {

                  distance = response[4];
                  for (int i = 0; i < response.length; i++) {
                        Log.i(TAG, "Byte[" + i + "][" + response[i] + "]");
                  }
            }
      } catch (Exception e) {
            Log.e(TAG, "Error in getUSDistance(), LSRead(" + e.getMessage() +
")");
      }
      return (distance);
}
```

(3)

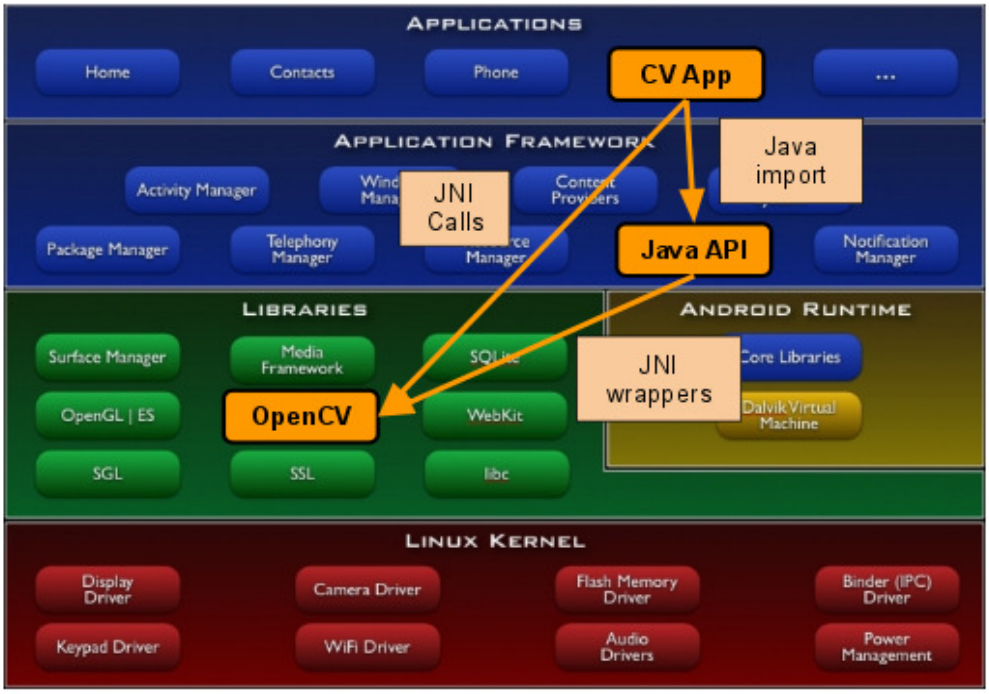
The method getUSDistance() uses two direct commands namely, LSWrite and LSRead and the I²C communication protocol to get the values. LS stand for low speed communication which indicates the I²C communication. At first LSWrite command is used to transmit the write message to get the reply telegram with values. For that purpose corresponding buffer size is declared. The first 5 buffers (1) are allocated for the direct command and the next 4 buffers (2).are allocated for the I²C communication command. Buffer [5] indicates the transmit data length ; buffer [6] is the receive data length (maximum 8 measurements) ; buffer [7] and buffer [8] are the commands to read measurements from byte 0. The next step, the LSRead method is to read the return package (3).which contains measurements [15].

Chapter 3:

3.0 OpenCV for Android

OpenCV supports several desktop and mobile operating systems including Windows, Linux, Mac OS X, Android and iOS. The cross platform of OpenCV is advantageous to developers. OpenCV for Android or OpenCV4Android is the official name of the Android port of the OpenCV library. In early 2010, OpenCV began supporting Android in a limited “alpha” fashion with OpenCV version 2.2. Then NVIDIA subsequently joined the project and accelerated the development of OpenCV for Android by releasing OpenCV version 2.3.1 with beta Android support. In April 2012, the first official non-beta of OpenCV (OpenCV 2.4) was released. Later versions of OpenCV, at time of this thesis’s writing OpenCV 2.4.5 has just been released, with even more Android support improvements and features.

OpenCV for Android supports two languages for OpenCV applications to run on Android-based devices. The first is in Java using the Android SDK and the second in C/C++ using the Android Native Development Kit (NDK). The Android NDK is a toolset that allows the developer to implement parts of the app using native code languages such as C and C++. For certain types of Apps such as Apps using

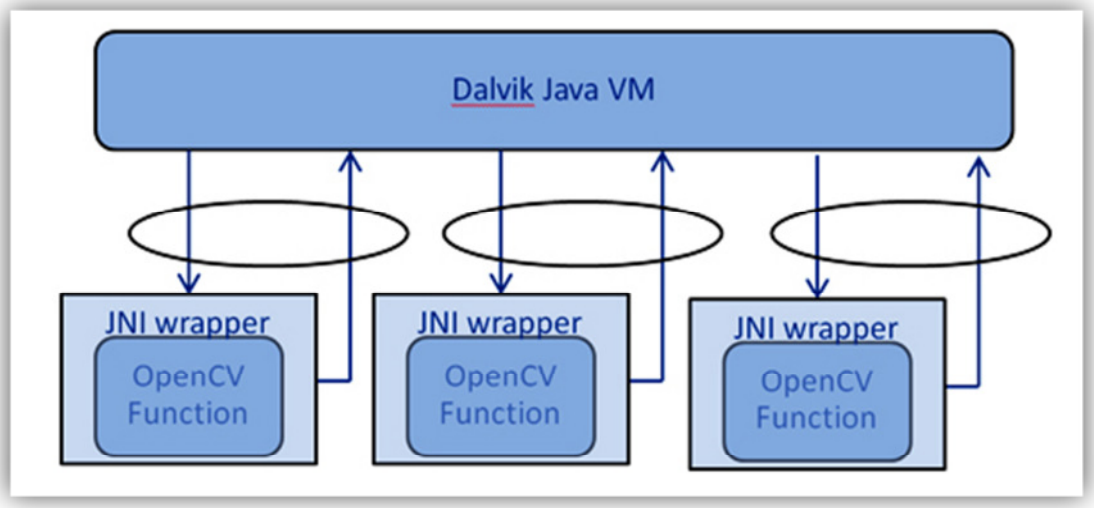


Figures 3. 1: The illustration of software stacks for Android application that use OpenCV for Android[19]

OpenCV, this can be helpful so that the user can reuse existing code libraries written in these languages[18]. The differences between the two methods, one is through its native API and the other through its Java API, is explained in sub-chapter 3.1and 3.2. The figure 3.1 shows the software stacks for Android applications that uses OpenCV for Android functions directly in native code, and in Java Code via their Java wrapper interfaces.

3.1 OpenCV Java API

This is the easiest way to develop the android application. Each ported OpenCV functions is “wrapped” with a Java Interface as shown in figure 3.1. Although the OpenCV function wrapped with Java Interface, all of the actual computations are



Figures 3. 2: Android application using the Java API[20]

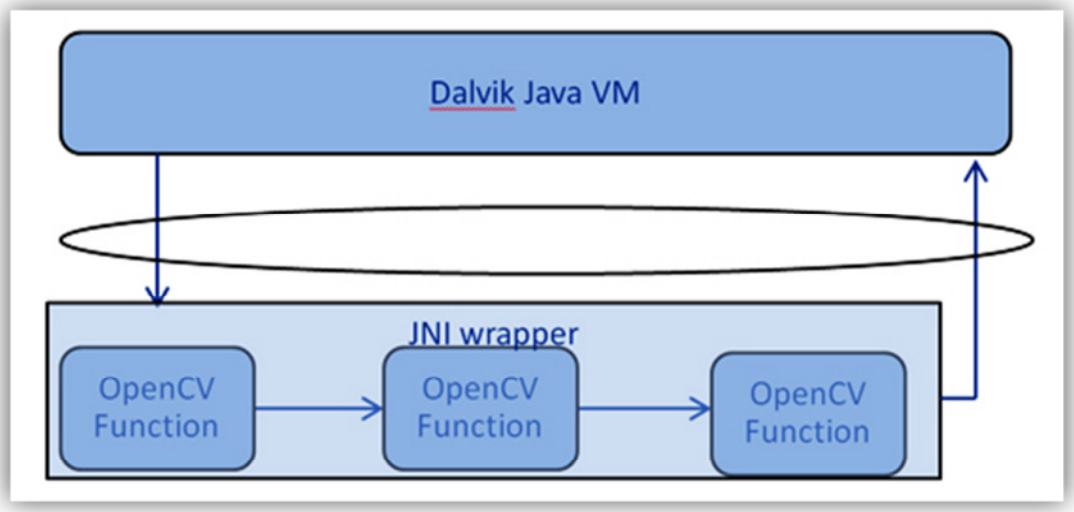
done at a native level. This is the result of the OpenCV function, which is written in C++ and compiled using the Android NDK.

Note that however, there is a performance penalty in the form of JNI (Java Native Interface) overhead, which occurs twice as a result of the Java wrapper.[20] The first occurs at the start of each call to the native OpenCV function the other occurs during the return. The performance penalty occurs for every OpenCV function. In other words, the more OpenCV functions called per frame, the bigger the total performance penalty.

The figure 3.2 illustrates an OpenCV Android application written using Java API: In this example the particular application calls three OpenCV functions per frame. The ellipses show the two JNI penalties (entry and exits) in one OpenCV function. In this application there are all together six JNI call penalties per frame which obviously decrease the performance.

3.2 OpenCV Native

This method uses the Android NDK (Native Development Kit), which slightly requires more programming efforts but optimizes the performance. In this approach, as shown in figure 3.1, the OpenCV code and hence the complete image processing algorithm



Figures 3. 3 Android application using the Android NDK[20]

is written entirely in C++, with direct calls to OpenCV. We simply wrap all of the OpenCV calls in a single C++ class, calling it once per frame. As a result only two JNI call penalties are incurred per frame, so JNI performance is significantly reduced compare to Java API.[20]

Figure 3.3 shows the same OpenCV for Android application as in Figure 3.2, but this time the Android application written using the native C++ API. The application calls three OpenCV function per video frame. The ellipse illustrates the resulting JNI penalties. The OpenCV portion of the application is written entirely in C++; therefore acquire no JNI penalties between OpenCV calls. Compare to the application using Java API, this application reduces the per video frame penalties from six to two.

OpenCV4Android contains a tutorial, which is split into 3 projects. The first project does not use OpenCV but shows only the basic framework of Image Processing App that is retained in subsequent projects .The second project uses OpenCV for image processing, but does perform neither image acquisition nor camera control tasks. The third project is then used for both tasks.

3.3 OpenCV Tutorial 0: Android Camera

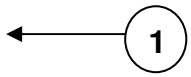
The purpose of this chapter is to explain the program flow of an OpenCV Tutorial program. The Tutorial 0 is an Android Java application, which does not implement the OpenCV library. However, this tutorial's basic framework constructs the backbone of all other OpenCV examples. Furthermore, the program flow in this thesis work is based on the tutorial 0; hence it is important to understand the program flow. For more tutorials check the tutorials from Tegra Android Toolkit Documentation [19]. To know how to build these tutorials in the Eclipse IDE, it is recommended to follow the "OpenCV for Android SDK" [22]

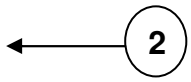
Figure 3.4 illustrates the UML Class Diagram of tutorial 0: Android Camera. In order to have better understanding of the structure of this application, some background of the Android Activity Lifecycle and Android Camera is required. We already have discussed about the Android Lifecycle. For more information about the Android Camera, it is recommended to look the Android documentation. [23]. The green colored classes in figure 3.4 are the android classes which are imported to the blue classes; the three main Java classes that are created for this application. The purple classes are the Java runnable classes. The functionality of these classes will be discussed in below.

Sample0Base

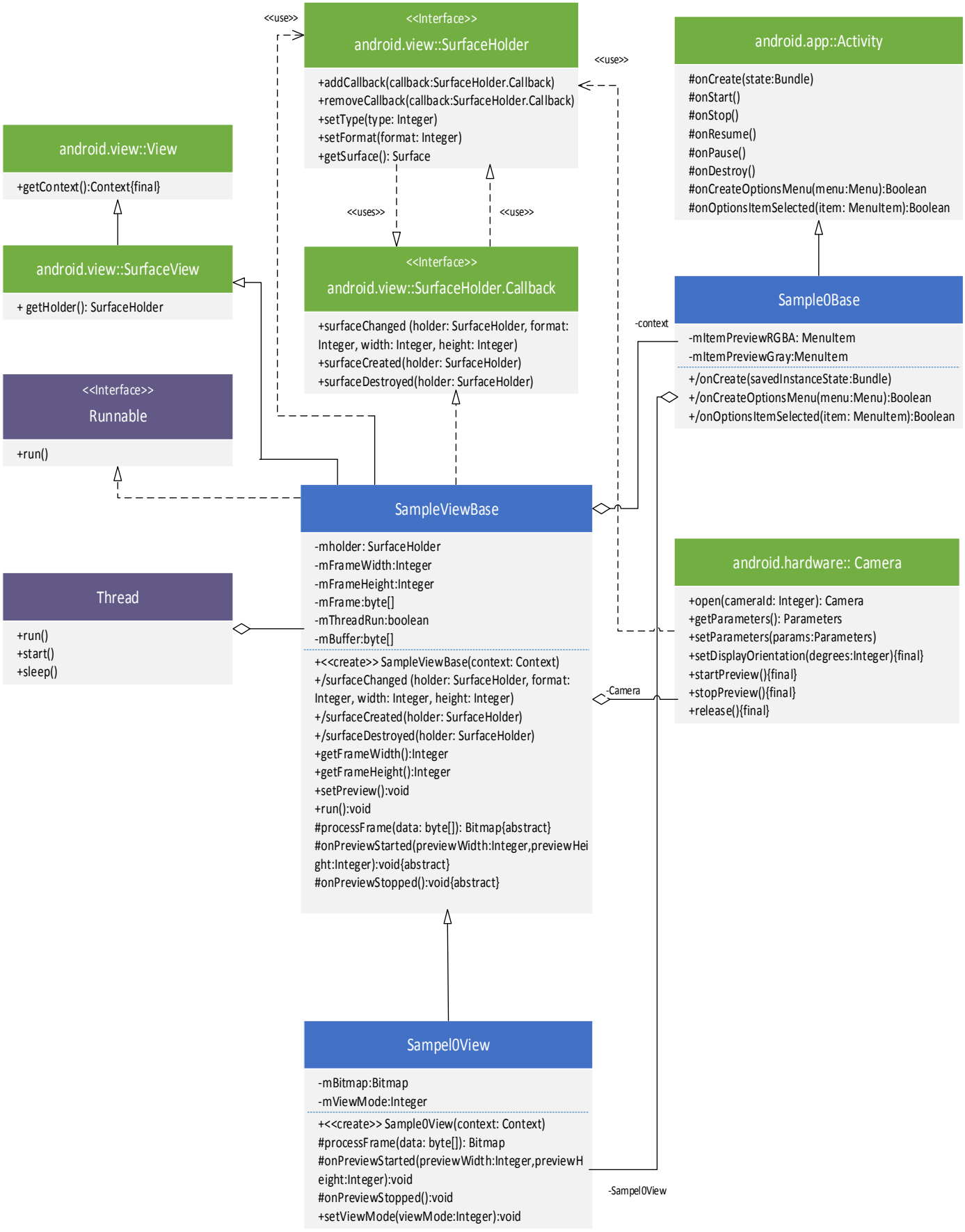
```
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.Window;

public class Sample0Base extends Activity {
    private static final String TAG            = "Sample::Activity";

    private MenuItem            mItemPreviewRGBA;
    private MenuItem            mItemPreviewGray;
```

```
    private Sample0View mView;


    public Sample0Base() {
        Log.i(TAG, "Instantiated new " + this.getClass());
    }

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {

        Log.i(TAG, "onCreate");
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        mView = new Sample0View(this);
        setContentView(mView);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        Log.i(TAG, "onCreateOptionsMenu");
        mItemPreviewRGBA = menu.add("Preview RGBA");
        mItemPreviewGray = menu.add("Preview GRAY");
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        Log.i(TAG, "Menu Item selected " + item);
        if (item == mItemPreviewRGBA)
             mView.setViewMode(Sample0View.VIEW_MODE_RGBA);
        else if (item == mItemPreviewGray)
             mView.setViewMode(Sample0View.VIEW_MODE_GRAY);
        return true;
    }
}
```





This class extends the Android Activity class and defines the operations that are performed when the application is created (1), paused and resumed. It also creates various menu options for the application (2), in our case two menus, Preview RGBA and Preview Gray. These two Menus determine the actions in which the original preview frames form the camera are processed before being displayed. The layout of this application is set via an instance of the Sample0View (1) class by using setContentView (). It is better to consider of the Sample0Base class as the entry point of this Android application.[19]



Figures 3. 4: UML (Unified Modeling Language) Class Diagram of Tutorial 0

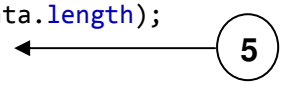
SampleViewBase

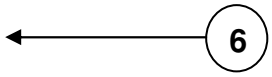
```
import java.io.IOException;
import java.util.List;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.ImageFormat;
import android.graphics.SurfaceTexture;
import android.hardware.Camera;
import android.hardware.Camera.PreviewCallback;
import android.os.Build;
import android.util.Log;
import android.view.SurfaceHolder;
import android.view.SurfaceView;

public abstract class SampleViewBase extends SurfaceView implements
SurfaceHolder.Callback, Runnable {
    private static final String TAG = "Sample::SurfaceView";

    private Camera              mCamera;
    private SurfaceHolder       mHolder;
    private int                 mFrameWidth;
    private int                 mFrameHeight;
    private byte[]              mFrame;
    private boolean             mThreadRun;
    private byte[]              mBuffer;


    public SampleViewBase(Context context) {
        super(context);
        mHolder = getHolder();
        mHolder.addCallback(this);
        Log.i(TAG, "Instantiated new " + this.getClass());
    }

    public int getFrameWidth() {
        return mFrameWidth;
    }

    public int getFrameHeight() {
        return mFrameHeight;
    }

    public void setPreview() throws IOException {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB)
            mCamera.setPreviewTexture( new SurfaceTexture(10) );
        else
           mCamera.setPreviewDisplay(null);
    }


    public void surfaceChanged(SurfaceHolder _holder, int format, int width, int
height) {
        Log.i(TAG, "surfaceCreated");
        if (mCamera != null) {
            Camera.Parameters params = mCamera.getParameters();
            List<Camera.Size> sizes = params.getSupportedPreviewSizes();
```

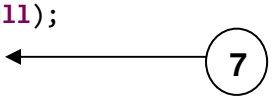
1

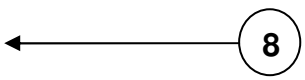
```
            mFrameWidth = width;
            mFrameHeight = height;

            // selecting optimal camera preview size
            {
                int  minDiff = Integer.MAX_VALUE;
                for (Camera.Size size : sizes) {
                    if (Math.abs(size.height - height) < minDiff) {
                        mFrameWidth = size.width;
                        mFrameHeight = size.height;
                        minDiff = Math.abs(size.height - height);
                    }
                }
            }

            params.setPreviewSize(getFrameWidth(), getFrameHeight());

            List<String> FocusModes = params.getSupportedFocusModes();
            if
(FocusModes.contains(Camera.Parameters.FOCUS_MODE_CONTINUOUS_VIDEO))
            {
             params.setFocusMode(Camera.Parameters.FOCUS_MODE_CONTINUOUS_VIDEO);
            }

            mCamera.setParameters(params);                                    2

            /* Now allocate the buffer */
            params = mCamera.getParameters();
            int size = params.getPreviewSize().width *
params.getPreviewSize().height;
            size  = size * ImageFormat.getBitsPerPixel(params.getPreviewFormat())
/ 8;
            mBuffer = new byte[size];
            /* The buffer where the current frame will be copied */
            mFrame = new byte [size];
            mCamera.addCallbackBuffer(mBuffer);

                   try {
                          setPreview();
                   } catch (IOException e) {
                          Log.e(TAG, "mCamera.setPreviewDisplay/setPreviewTexture
fails: " + e);
                   }

            /* Notify that the preview is about to be started and deliver preview
size */
            onPreviewStared(params.getPreviewSize().width,
params.getPreviewSize().height);

            /* Now we can start a preview */
            mCamera.startPreview();
        }
    }

    public void surfaceCreated(SurfaceHolder holder) {
        Log.i(TAG, "surfaceCreated");
        mCamera = Camera.open();                                              3

        mCamera.setPreviewCallbackWithBuffer(new PreviewCallback() {          4
            public void onPreviewFrame(byte[] data, Camera camera) {
```

```
                synchronized (SampleViewBase.this) {
                    System.arraycopy(data, 0, mFrame, 0, data.length);
                    SampleViewBase.this.notify();
                }
                camera.addCallbackBuffer(mBuffer);
            }
        });

        (new Thread(this)).start();
    }

    public void surfaceDestroyed(SurfaceHolder holder) {
        Log.i(TAG, "surfaceDestroyed");
        mThreadRun = false;
        if (mCamera != null) {
            synchronized (this) {
                mCamera.stopPreview();
                mCamera.setPreviewCallback(null);
                mCamera.release();
                mCamera = null;
            }
        }
        onPreviewStopped();
    }

    /* The bitmap returned by this method shall be owned by the child and released
in onPreviewStopped() */
    protected abstract Bitmap processFrame(byte[] data);

    /**
     * This method is called when the preview process is beeing started. It is
called before the first frame delivered and processFrame is called
     * It is called with the width and height parameters of the preview process.
It can be used to prepare the data needed during the frame processing.
     * @param previewWidth - the width of the preview frames that will be
delivered via processFrame
     * @param previewHeight - the height of the preview frames that will be
delivered via processFrame
     */
    protected abstract void onPreviewStared(int previewWidtd, int previewHeight);

    /**
     * This method is called when preview is stopped. When this method is called
the preview stopped and all the processing of frames already completed.
     * If the Bitmap object returned via processFrame is cached - it is a good
time to recycle it.
     * Any other resourcses used during the preview can be released.
     */
    protected abstract void onPreviewStopped();

    public void run() {
        mThreadRun = true;
        Log.i(TAG, "Starting processing thread");
        while (mThreadRun) {
            Bitmap bmp = null;

            synchronized (this) {
                try {
                    this.wait();
                    bmp = processFrame(mFrame);
```









```
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }

            if (bmp != null) {
                Canvas canvas = mHolder.lockCanvas();
                if (canvas != null) {
                    canvas.drawBitmap(bmp, (canvas.getWidth() - getFrameWidth()) /
2, (canvas.getHeight() - getFrameHeight()) / 2, null);
                    mHolder.unlockCanvasAndPost(canvas);
                }
            }
        }
    }
}
```

This class extends the Android SurfaceVew class. It also uses the Android SurfaceHolder.Callback and the Java Runnable interfaces. The main motive of this class is to determine a layout for the Android application, which includes a generalizable framework to continuously capture live preview frames from the camera, process and display them. [19]

The SurfaceHolder.Callback class has three callbacks method: surfaceCreated(),surfaceChanged(), and surfaceDestroyed(). These three callbacks form a hierarchy within themselves. The implementation of SurfaceHolder.Callback always receives a surfaceCreated callback first to indicate a surface is available for the SurfaceView. Then one or more surfaceChanged callbacks are called to indicate that the format (width and height) have changed. Lastly the surfaceDestroyed is called to indicate that the SurfaceView no longer makes a surface available.

An essential member of the SampleViewBase class is an instance of the Android Camera class. The SampleViewBase class uses methods from the android camera class to open (3), configure (2) and release (7) its member Camera object. As soon as the SurfaceCreated() method is called, it opens the camera. Once the camera is successfully opened, the live preview is started and video frames are received. Link with the Camera object is a PreviewCallback() (4),which run on every preview frame received by the camera. The PreviewCallback defined at (4), copies the preview picture data to the byte[ ] mFrame and notifies(5) an asynchronous processing work thread of the SurfaceHolder.Callback to process and display the frame. [19]

A thread, also called a lightweight process, is a single sequential flow of programming operations, with a definite beginning and an end[18]. A thread is not a program because it cannot run on its own; instead it runs within a program. [24]

A new Thread is created in this class by creating a new instance at (6) and invoke the start() method , which will call back the run() on a new thread. The run() method specifies the running behavior of the thread and gives the thread to do something. In our case the thread waits (8) to receive a preview frame from the camera in the byte[ ] mFrame, processes it, and pushes the output to an object of the Android Canvas class to be displayed. The thread resumes when the notify() method is called at (5).in the next loop.

The SampleViewBase class also includes abstract methods processFrame(byte[ ] data), on PreviewStarted(int previewWidth, int previewHeight) and onPreviewStopped(), which are implemented in its child Sample0View class. The general framework defined in the SampleViewBase class is copied in all the OpenCV examples and in this thesis work. But, its implementation modifies slightly depending on the options to use the Android Camera class or the OpenCV VideoCapture class to access the device camera. In this thesis the OpenCV VideoCapture class is used.

Sample0View

```
import android.content.Context;
import android.graphics.Bitmap;
import android.util.Log;

class Sample0View extends SampleViewBase {

       private static final String TAG = "Sample0View";
       int mSize;
       int[] mRGBA;
       private Bitmap mBitmap;
    private int mViewMode;

    public static final int     VIEW_MODE_RGBA = 0;
    public static final int     VIEW_MODE_GRAY = 1;


    public Sample0View(Context context) {
        super(context);
        mSize = 0;
        mViewMode = VIEW_MODE_RGBA;
    }

    @Override
    protected Bitmap processFrame(byte[] data) {
        int frameSize = getFrameWidth() * getFrameHeight();
```

```
        int[] rgba = mRGBA;

        final int view_mode = mViewMode;
        if (view_mode == VIEW_MODE_GRAY) {
            for (int i = 0; i < frameSize; i++) {
                int y = (0xff & ((int) data[i]));
                rgba[i] = 0xff000000 + (y << 16) + (y << 8) + y;
            }
        } else if (view_mode == VIEW_MODE_RGBA) {
            for (int i = 0; i < getFrameHeight(); i++)
                for (int j = 0; j < getFrameWidth(); j++) {
                   int index = i * getFrameWidth() + j;
                   int supply_index = frameSize + (i >> 1) * getFrameWidth() + (j
& ~1);
                    int y = (0xff & ((int) data[index]));
                    int u = (0xff & ((int) data[supply_index + 0]));
                    int v = (0xff & ((int) data[supply_index + 1]));
                    y = y < 16 ? 16 : y;

                    float y_conv = 1.164f * (y - 16);
                    int r = Math.round(y_conv + 1.596f * (v - 128));
                    int g = Math.round(y_conv - 0.813f * (v - 128) - 0.391f * (u -
128));
                    int b = Math.round(y_conv + 2.018f * (u - 128));

                    r = r < 0 ? 0 : (r > 255 ? 255 : r);
                    g = g < 0 ? 0 : (g > 255 ? 255 : g);
                    b = b < 0 ? 0 : (b > 255 ? 255 : b);

                    rgba[i * getFrameWidth() + j] = 0xff000000 + (b << 16) + (g <<
8) + r;
                }
        }

        mBitmap.setPixels(rgba, 0/* offset */, getFrameWidth() /* stride */, 0, 0,
getFrameWidth(), getFrameHeight());
        return mBitmap;
    }

      @Override
      protected void onPreviewStared(int previewWidth, int previewHeight) {
            /* Create a bitmap that will be used through to calculate the image
to */
        mBitmap = Bitmap.createBitmap(previewWidth, previewHeight,
Bitmap.Config.ARGB_8888);
      mRGBA = new int[previewWidth * previewHeight];
      }

      @Override
      protected void onPreviewStopped() {
            mBitmap.recycle();
            mBitmap = null;
            mRGBA = null;
      }

      public void setViewMode(int viewMode) {
            mViewMode = viewMode;
      }
```

This class extends the SampleViewBase class and implements its abstract methods processFrame(byte[] data),onPreviewStarted(int previewWidth, int previewHeight) and onPreviewStopped().

The onPreviewStarted() method is called before the camera preview is started. Meanwhile, the processFrame() method hold responsibility to transform each incoming camera preview frame by doing image processing and computer vision functions on it. The specific processing mode is determined by the menu option selected by the user at the time (1),(2). At last onPreviewStopped() method clean up the member arrays of the Sample0View class after its Android Camera object has been destroyed.[19]

To summarize, the OpenCV Tutorial 0-Android Camera application creates a view in which live previews are continuously received from the camera. Then the images are modified into specific tasks by the user via the application menu options and displayed.

3.4 Android Camera vs OpenCV VideoCapture (Native Camera)

As mentioned in chapter 3.3, in this thesis the OpenCV VideoCapture class is used to grab the frames instead of the Android camera in the tutorial 0. There is no solid reason why this class is used, as one of the other OpenCV sample tutorial is used to develop the application. Therefore, in this chapter we will go through the differences among them and evaluate the efficiency of both classes.

In most cases there are not many differences between these two classes. Only few differences are noticed during this thesis work. To perform the image processing methods for the OpenCV library group the OpenCV Imgproc.java must imported. For the android library, the raw data is utilized from android.hardware.Camera and android.hardware.Camera.PreviewCallback as the input frame image. The android camera class grabs the video in YUV format, whereas the VideoCapture grabs the images and convert them into specified color space usually to gray or RGB color space.

In OpenCV library the frame data are saved in the Mat (Matrix) structure, which is later passed to the OpenCV's image processing functions in order to process each pixel in the frame. Meanwhile the Android Camera saved the frames in one dimensional byte array.

For the Android Camera, Preview Callback() is nessesary as it ensures the clock timing of image processing. The camera generates thereby continuously images. This is the direct access to a camera on the Android operating system. In the OpenCV native camera, the developer must not worry about the camera. It will trigger itself with the command "capture.retrieve" the image acquisition, whereby this command is located within a loop. Presumably, this command is converted into corresponding Android commands (like the above) after compiling. More details on this are unfortunately not available. A disadvantage of using the OpenCV camera is (at time of writing) the lack of options to set the camera parameters (eg auto focus mode, white balance mode select, etc.).

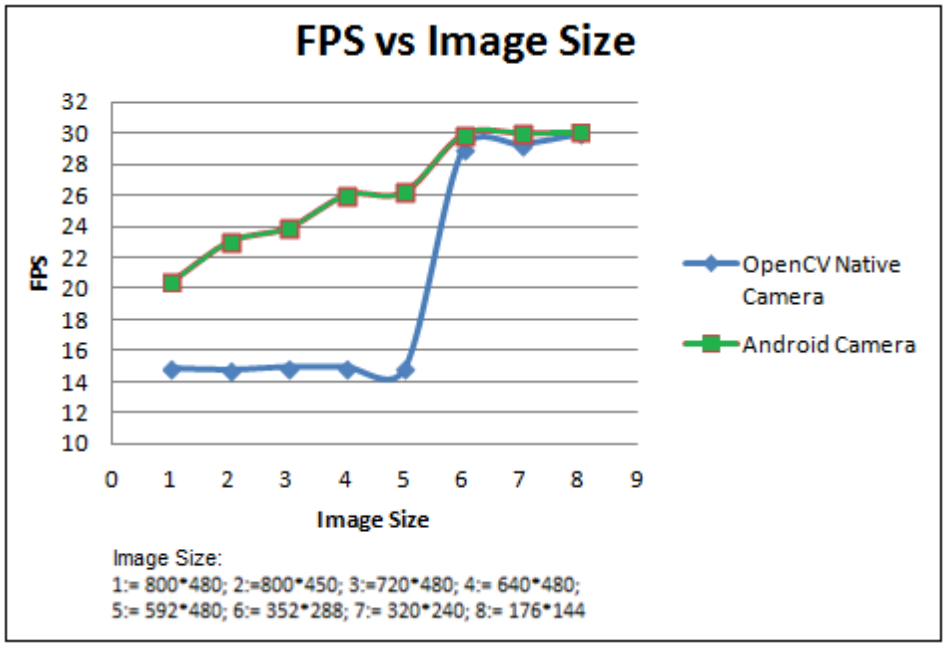
During this thesis work the efficiency of both classes were experimented. The efficiency is calculated in the form of frame rate or frame per second (FPS) according to the equation below:

$$Frame\ rate = \frac{no.of\ processed\ frame}{1s}$$

In any image processing methods, the higher FPS make the process faster, thus more efficient and vise versa. However, the FPS is influenced by some other factors, such as illuminance, image size, the complexity of image processing and so on. Therefore, the experiment is conducted to figure out not only the influence of both classes, but also the influence of the factors to optimize the efficiency.

3.4.1 FPS vs. Image Size

In this experiment, different image preview sizes are used to figure out the resulting frame rates. The FPS is measured using FPSMeter java class. Below is the graph with the usage of Android Camera and OpenCV VideoCapture (Native):



Figures 3. 5: Graph frame rate against Image Size.

Figure 3.5 shows the graph frame rate against image size for both android camera and OpenCV native camera. The numbers in the graph are the average value of 5 repeated tests. As shown in the figure there is a significant improvement in FPS as the image size is getting smaller. However, there is a notable different in FPS between OpenCV Native camera and Android Camera. Although the FPS of both classes for smallest image size is similar (the camera is only running at up to 30FPS; a higher frame rate is therefore not possible), the FPS for the largest image size at the beginning of the graph is different. This means that the FPS of Android camera are much higher for larger images than the OpenCV native camera, thus OpenCV native camera reduces the effiency for larger images.

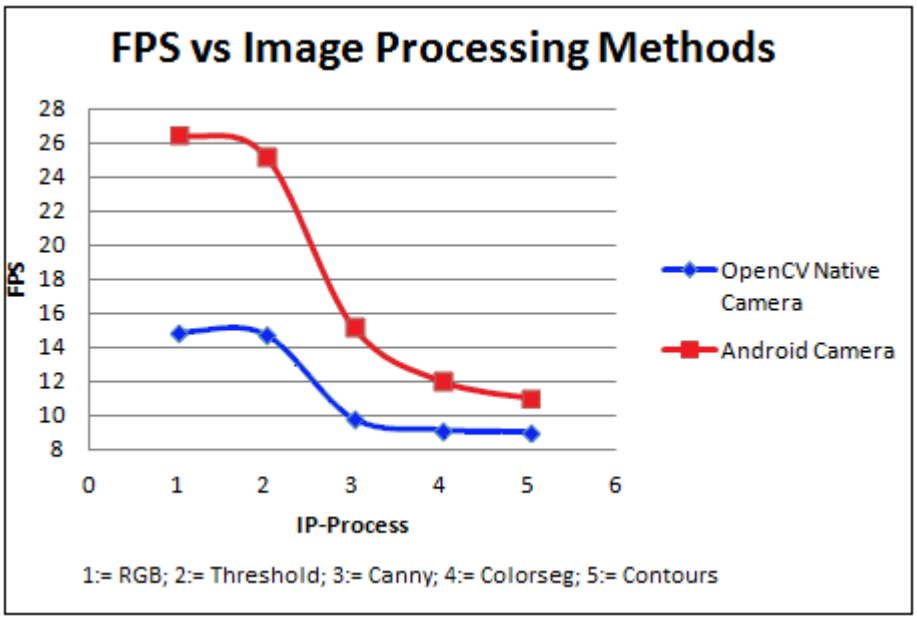
3.4.2 FPS vs Image Processing Methods

5 basic image processing methods are applied to each frame captured from the 5 mega pixels camera of Samsung galaxy S's Smartphone. The image processing methods are tested and categorized into the ascending complexity of the codes that applied to the methods.

Frame Processing	Description
RGB	Convert the original YUV Color Space to RGB Color Space
Threshold	Threshold the grayscale pixel with 70
Canny edge	Detects edges in image
Colorseg	Extract specificied color pixels
Contour	Find contour of an object.

Table 3.1: Descriptions of Image processing methods.

The descriptions of the 5 image processing methods are shown in Table 2. The Image processing methods ColorSeg and Contour are well explained in the chapter 3.5.3 and 3.5.4. The methods are arranged in a way that the method on last row is more complex than method on first row.



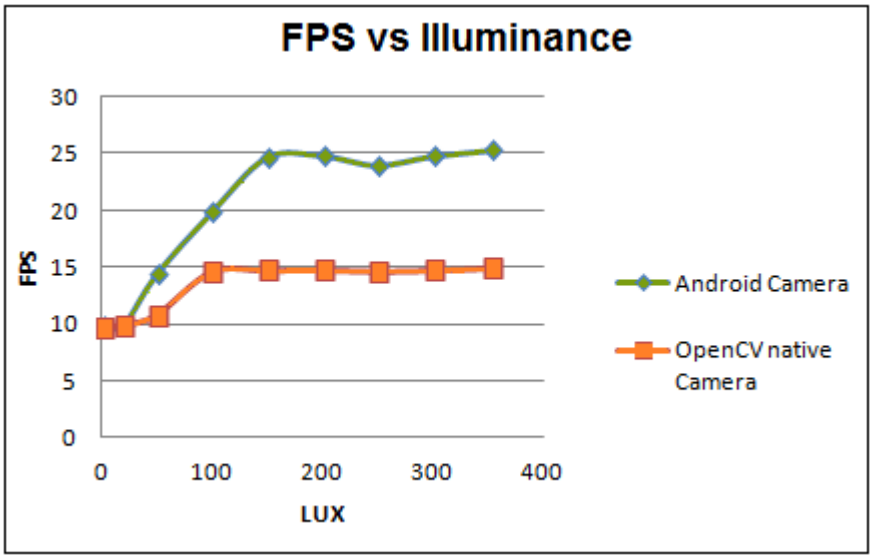
Figures 3. 6: Graph frame rate against Image Processing Methods.

The graph 3.6 shows the frame rate against Image processing methods. The image size was set to 640*480. Again we can see here that the FPS varies against the

complexity of the image processing methods. The more complex an Image processing method is, the lower the frame rate is. In this experiment we can see again the different frame rates between the two camera classes. Again the Android Camera has an advantage against the OpenCV Native Camera.

3.4.2 FPS vs. Illuminance

Illuminance is the total luminous flux incident on a surface, per unit area. It is a measure of how much the incident light illuminates the surface. The illuminance definitely influences the frame rate, because in a darker environment or low illuminance the shutter of a camera waits longer to capture enough lights, thus reduces the frame captured by the camera in a given time. In this experiment, the relationship between frame rate and illuminance is tested.



Figures 3. 7: Graph frame rate against Illuminance

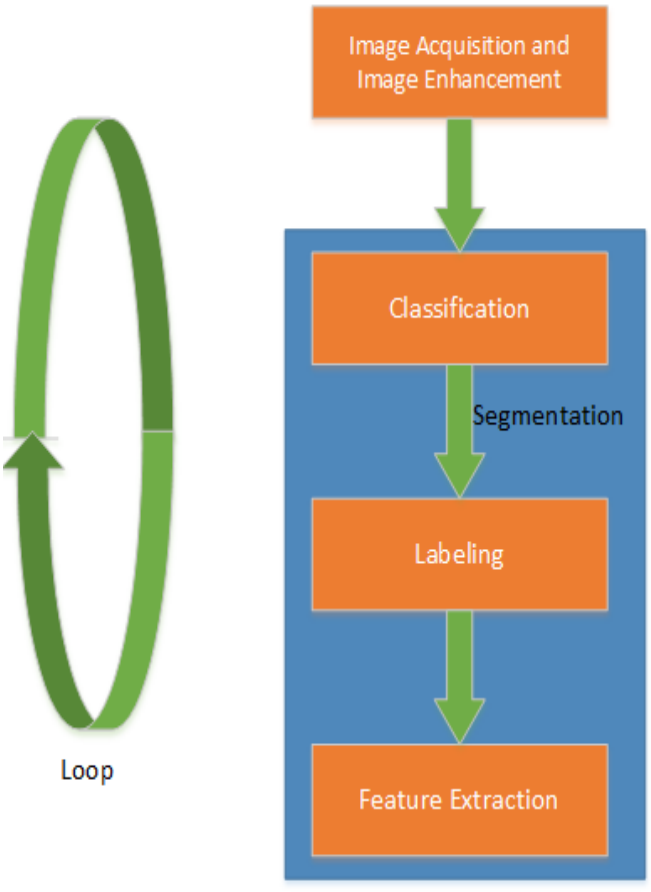
The figure 3.7 illustrates the result of the experiment. The image size was set to 640*480 and the RGB image processing method was used to figure out the frame rates. The illuminance is measured with the unit LUX using PEAKTECH 5065-2 Digital Luxmeter. As shown in the figure 3.7, a minimum illuminance is required to achieve the maximum FPS for both camera classes. The darker environment, which is below 150 LUX influences significantly the FPS. It reduces the fps down to 10 fps. Again the android camera class behaves better in a darker environment than the OpenCV native camera. For the lowest illuminance FPS are therefore identical because the FPS of the camera dominates.

Conclusion

Through all these experiments it is obvious that the Android camera has the upper hand than OpenCV native camera. It is also proved that some factors influence the FPS such as the illuminance, image size, and image processing methods. Therefore a combination of optimized factors is chosen to get better efficiency. However, there are some other factors may influence the FPS such as the autofocus, buffer allocation and etc., that are yet experimented. In a nutshell, the Android camera has advantage than the OpenCV native camera in terms of efficiency and the fps is the function of illuminance, image size and processing methods and etc.

3.5 Object Detection via Color Segmentation

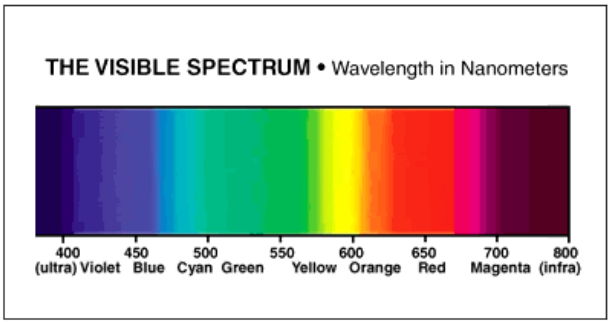
This chapter explains the fundamental steps in Image Processing to process an Image according to a desired task. As the title itself mentions, in this thesis work an object is detected by segmenting its color from the camera images and extract the largest blob or region of the color segment. To achieve this, few fundamental steps in Image Processing such as Image Acquisition, Image Enhancement, and Segmentation are followed sequentially. As they are fundamental, all these steps may have sub-steps such as segmentation, which has classification, labeling and feature extraction as sub-steps as illustrated in figure3.5. Note that these 4 steps are used to process a single image. Therefore these steps are called in a loop for every image received from the camera. The implementation of OpenCV libraries did reduce the programming efforts by inserting one or two lines of OpenCV function for every step. Since we want to do color segmentation it is better to touch the topic color before we discuss these fundamental steps.



Figures 3. 8: The basic steps of Image processing to detect an object via Color Segmentation

3.5.1 Color and Color Models

Color is the visual perceptual results of light in the visible region of the spectrum of light. Light that is visible to humans lies in the range of wavelength from 400nm (violet) to 700nm (red) which consists of colors blue, green, yellow ,orange in between as shown in figure 3.6. Color categories and its physical specification are



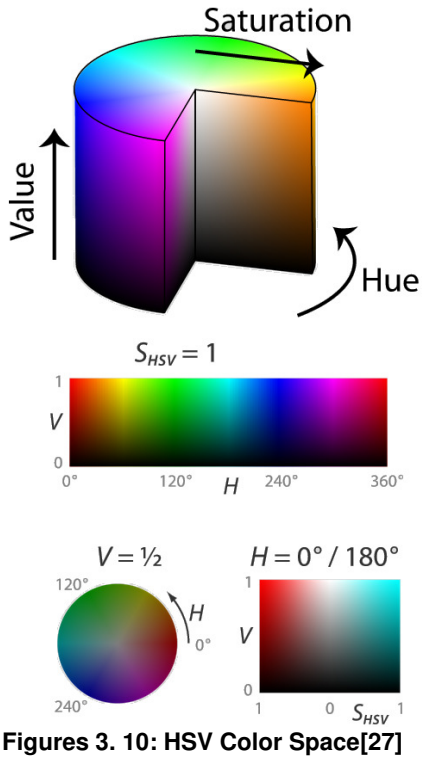
Figures 3.9: The spectrum of visible colors as a function of wavelength in nanometers.[25]

mostly associated with objects, material light sources and etc., depending on their physical properties such as light absorption, reflection, or emission spectra. There are several color spaces or color models exist, which describe a color by numerically identifying its coordinate in the color model [26].

The most notable color model is the RGB color model in which the red, green and blue lights are added together in many ways to reproduce a broad array of colors. The name of this model comes from the three primary colors: red, green and blue. The RGB color model is additive because three light beams are added together to make final color's spectrum. A color in the RGB color model is described by indicating how much of each red, green and blue is contained. The color is defined as an RGB triplet(r,g,b), each component can vary from zero to a defined maximum value.

Another representation of the color is the HSV color model. HSV color model is the cylindrical representation of points in an RGB model. The model rearranges the geometry of RGB in order to have much better information of the color than the cube representation of RGB model. HSV stands for hue, saturation and value and is often called HSB (B for brightness)[26],[27].

In the cyclinder the angle around the central vertical axis is equal to "hue", the distance from the axis is equal to "saturation" and the distance along the axis is the "brightness". This color model is used in computer graphics, computer vision because it is more convenient than the RGB model. In chapter 3.5.2 we will use this color model to extract information from the images [26],[27].



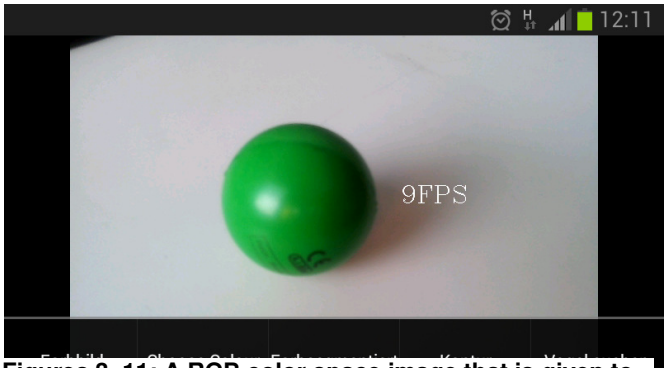
Figures 3. 10: HSV Color Space[27]

3.5.2 Image Acquisition and Image Enhancement

Image acquisition is the first step or process of the fundamental steps of image processing. Image acquisition means giving an Image in a digital form to the process. This first stage usually involves preprocessing such as scaling and etc. All the image processing steps are implemented in the method processFrame (Videocapture capture) in the class OcvViewImgProc.java. Below is the following coding to get an image from the camera using OpenCV native camera:

```
protected Bitmap processFrame(VideoCapture capture) {

.
.
      capture.retrieve(mRgba,Highgui.CV_CAP_ANDROID_COLOR_FRAME_RGBA);
.
.
}
```

The OpenCV function VideoCapture decodes and returns the just grabbed frame. Furthermore it grabs a frame and converts it to RGB color image and saves in a matrix (mRgba).



Figures 3. 11: A RGB color space image that is given to the process stage, Image Acquisition

Since the digital image is grabbed in RGB color space by the videocapture.retrieve () function, we have to convert the image to HSV color space. For that purpose we insert another function of OpenCV (cvtColor),

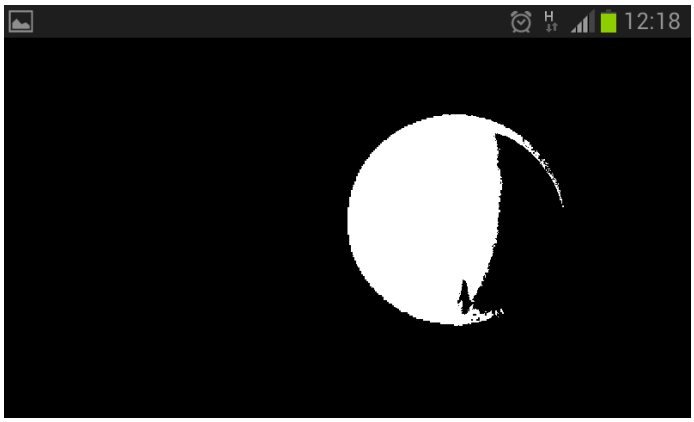
```
Imgproc.cvtColor(mRgba, mIntermediateMat,Imgproc.COLOR_RGB2HSV_FULL)
```

which converts an image from one space color to another. Note that the conversion range values are from 0 to 255, the range that a single 8-bit byte can offer. In the conversion of RGB to HSV, the R,G,B values are converted to the floating point format and scaled to fit 0 to 1 range. Note that the hue value is halved to fit 0 to 255 ranges. For more information check the OpenCV documentation. The converted image is than saved in an intermediate matrix layer (mIntermediateMat). This sort of process is called as Image enhancement, where obscured details are bought out or simply certain features of interest in an image are highlighted

3.5.3 Pixel Classification

A Smartphone camera captures color images in digital formats. The digital image is a numeric representation of a two-dimensional image, which has a finite set of digital values, called picture elements or simply pixels. Pixels are the smallest element in an Image, carrying values that represent the brightness of a given color at any specific point. So we need to extract the information of these pixels to get the value of RGB or HSV to do the color segmentation. Pixel classification categorizes the pixels in a digital image into one of several classes or themes. In our case we categorize the pixel to identify green color pixels from the image. Therefore, the OpenCV function inRange() is used to check if array elements (pixels) lie between the elements of two other arrays.

```
Core.inRange(mIntermediateMat, lo, hi, mIntermediateMat2);
```



Figures 3. 12: Green pixels are extracted from the image

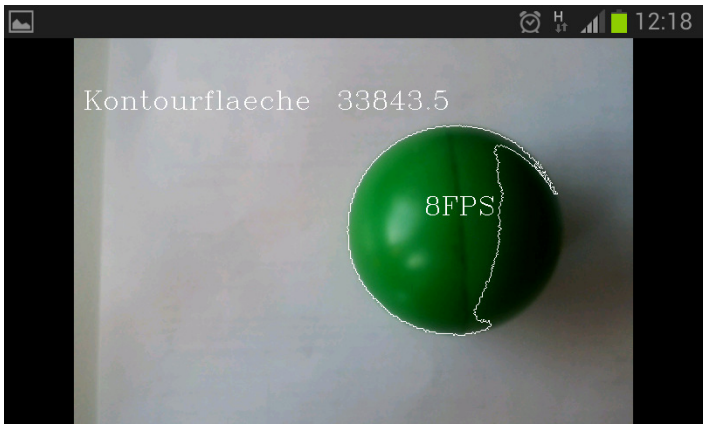
For example, we want to extract only the green pixels in the image. Therefore a tolerance range of green points in HSV color model is defined. The lo (Scalar(85, 50, 20) scalar point indicates the lowest possible green coordinate in HSV model, whereas the hi(Scalar(130,255,255)) corresponds the highest green coordinate in HSV model. Note that the hue values are halved. Once this function is used green pixels are extracted from the image. The figure 3.9 shows the extracted green pixels in white. As you can see some part of the green ball is not recognized, because the pixels are too dark and out of the tolerance range.

3.5.4 Image Labeling

Image labeling is the process to detect connected regions in binary and color images. This works according to the connected-component labeling algorithm, where a pixel

is compare with the neighboring pixels to decide if the pixels are in the same characteristic group. All the connected pixels make regions know as blob. This process is used in this thesis to work to find the connected color regions and the biggest region out of them.

```
Imgproc.findContours(mIntermediateMat2, contours, hierarchy,
            Imgproc.RETR_LIST, Imgproc.CHAIN_APPROX_SIMPLE);

            // Search the biggest contour
            indexMaxArea = -1;
            for (int i = 0; i < contours.size(); i++) {
                        double s = Imgproc.contourArea(contours.get(i));
                        if (s > maxArea) {
                        indexMaxArea = i;
                        maxArea = s;
}
```



Figures 3. 13: Biggest contours via Labeling

The codes above show the OpenCV function findContours(), which retrieve contours from images and saves them in an array. Then the biggest blob is identified to do feature extraction out of it. The figure 3.10 shows the biggest connected green region in the image.

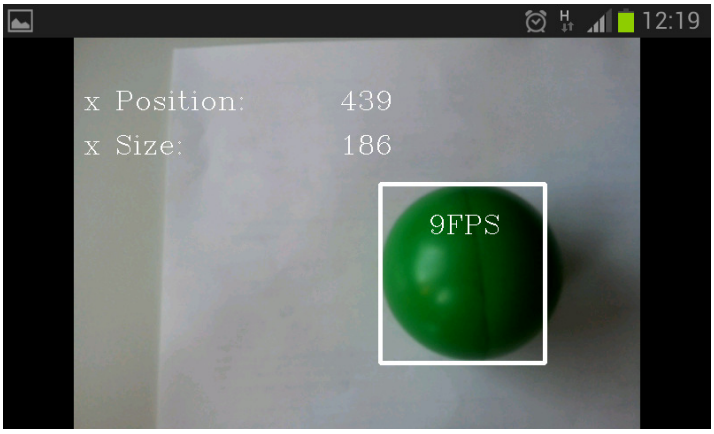
3.5.5 Feature Extraction

This last step namely Feature extraction is to get some characteristic values of the selected region. Feature extraction is normaly used in reduced representation of an image instead of the full size input. As in our case we only focused in our biggest green region to get the values instead of using the full image.

```
MatOfPoint ContuBig = new MatOfPoint();
ContuBig = contours.get(indexMaxArea);

      // for this purpose import org.opencv.core.Rect.must be imported
      Rect vogRect = new Rect();

      // calculate the rectangle for the largest contour
      vogRect = Imgproc.boundingRect(ContuBig);
      // Draw this rectangle into matrix mRgba
      Core.rectangle(mRgba, vogRect.tl(), vogRect.br(), wh, 3);
      // give out the horizontal centre of VogRect
```

```
xVogRect = vogRect.x + (vogRect.width / 2);
// give out the width of the Rectangle
wVogRect = vogRect.width;
```

The above code lines are to create a rectangle boundary around the green blob and extract the position and size of the rectangular. These two values are then given to control the robot, which is explained in chapter 4.0. The figure 3.11 shows the rectangular boundary of the connected green pixels and its x position and size.



Figures 3. 14: Feature extraction from the blob

Chapter 4:

4.0 Controlling NXT via Android Smartphone

There are two options to program or to control the Lego Mindstorms Robot. A user can control the robot by using either the NXT software provided by the Lego or using the direct commands protocol by an outside device. In our case, the direct commands protocol is used to access the control of the robot by using Smartphone. In this chapter we will discuss about the implementation of controlling methods in the Android App, which controls the robot.

Figure 4.1 shows the UML diagram of OpenCV+NXTDemo Apps, which is created for this thesis with the purpose of controlling the Mindstorms robot using the Smartphone. The figure 4.1 is similar to the figure 3.4 of tutorial 0, with some modification in member of the classes such as the methods. The figure 4.1 only shows the created java classes for this Apps, although it has some hidden classes such as surface.Callback and runnable interface classes.

As usual the OcvNxtMainAct.java class is the entry point to this App which defines the operation when the App is created, paused and resumed. It also creates two option menus, the start and stop menu in order to activate and deactivate the robot. Then the OcvViewImgProc.java class is called to perform the image processing, which calls its parent OcvViewBase.java class to set the camera parameters. OcvViewBase.java also has run() method ,which calls the processFrame() method in loops for each frame received from the camera.

There are two additional classes created to control the robot namely the NxtRobotControl.java and NxtBrick.java classes. The purpose of the NxtRobotControl.java class is to provide methods to control the robot and access to the NXT sensors. Meanwhile the Nxtbrick.java is responsible to establish the Bluetooth communication between the NXT Lego Mindstorms robot and the Smartphone as well as provides methods to send and receive Bluetooth messages. The Movemotor(), setUSSensor(), and getUSDistance() methods that discussed in chapter 2.2 are implemented in NXTBrick.java class.

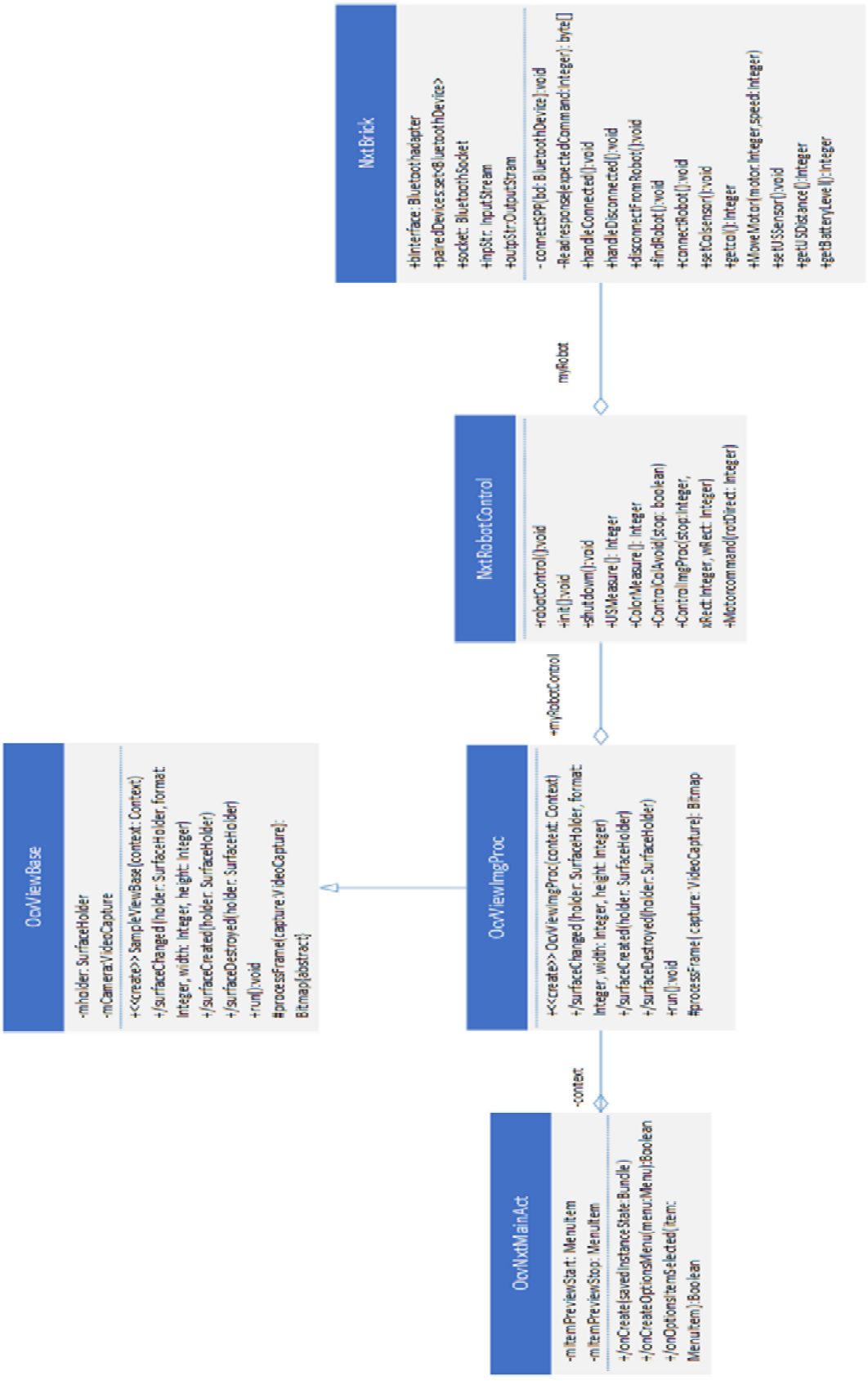


Figure 4.1: UML Class Diagram of OpenCV+NXTDemo App

4.1 Implementation of NXT Sensors

There are two NXT sensors used in this thesis work. The ultrasonic sensor is used for collision avoidance and the color sensor is used to help grab the detected object. As explained in earlier chapters the Ultrasonic sensor uses the I²C communication protocol.

I²C is a half-duplex, bi-directional, synchronous serial data bus developed by Philips in 1982, now a highly popular communication standard.[28] It provides a simple and efficient communication possible between a microcontroller and one or more actuators or sensors.

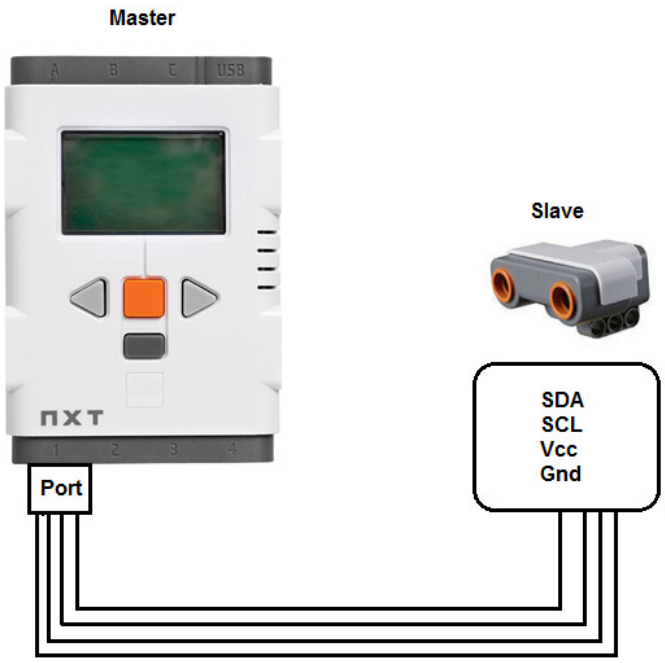


Figure 4.2: NXT Brick's master-slave behavior.

An I²C Communication works on master-slave principle. Only the master initiates the communication and it can send commands to the slave and it can query them for data. As shown in figure 4.2 the intelligent brick is necessarily the master and the Ultrasonic sensor is necessarily the slave.

As in figure 4.2 there are four wires needed for I²C communication between the sensor and NXT brick:

- Gnd(ground) is the reference potential.
- Vcc supplies power to the sensor
- SCL is the clock line, synchronizing the transaction between the two units.
- SDA is the data line, which is the line carrying the data exchanged between the two units.

The NXT connection cable that provided by the Lego contains all these four wires to support I²C communication. The I²C communication of the ultrasonic sensor is unstable when the battery voltage is too low. This is particularly the case if you use rechargeable batteries.

Since the Ultrasonic sensor uses the I²C communication the Lego I²C Communication Protocol with Direct Commands is used to get the value from the sensor and sent it to Smartphone via Bluetooth communication, which is discussed in chapter 2.2.

The color sensor operates by using a single white LED (light emitting diode) to illuminate the target and analyses the color components of the light reflected by the target's surface and calculates a Color Number that is returned to the NXT. It can detect 6 colors and returns the corresponding hexadecimal value:

- 0x01 → Black
- 0x02 → Blue
- 0x03 → Green
- 0x04 → Yellow
- 0x05 → Red
- 0x06 → White

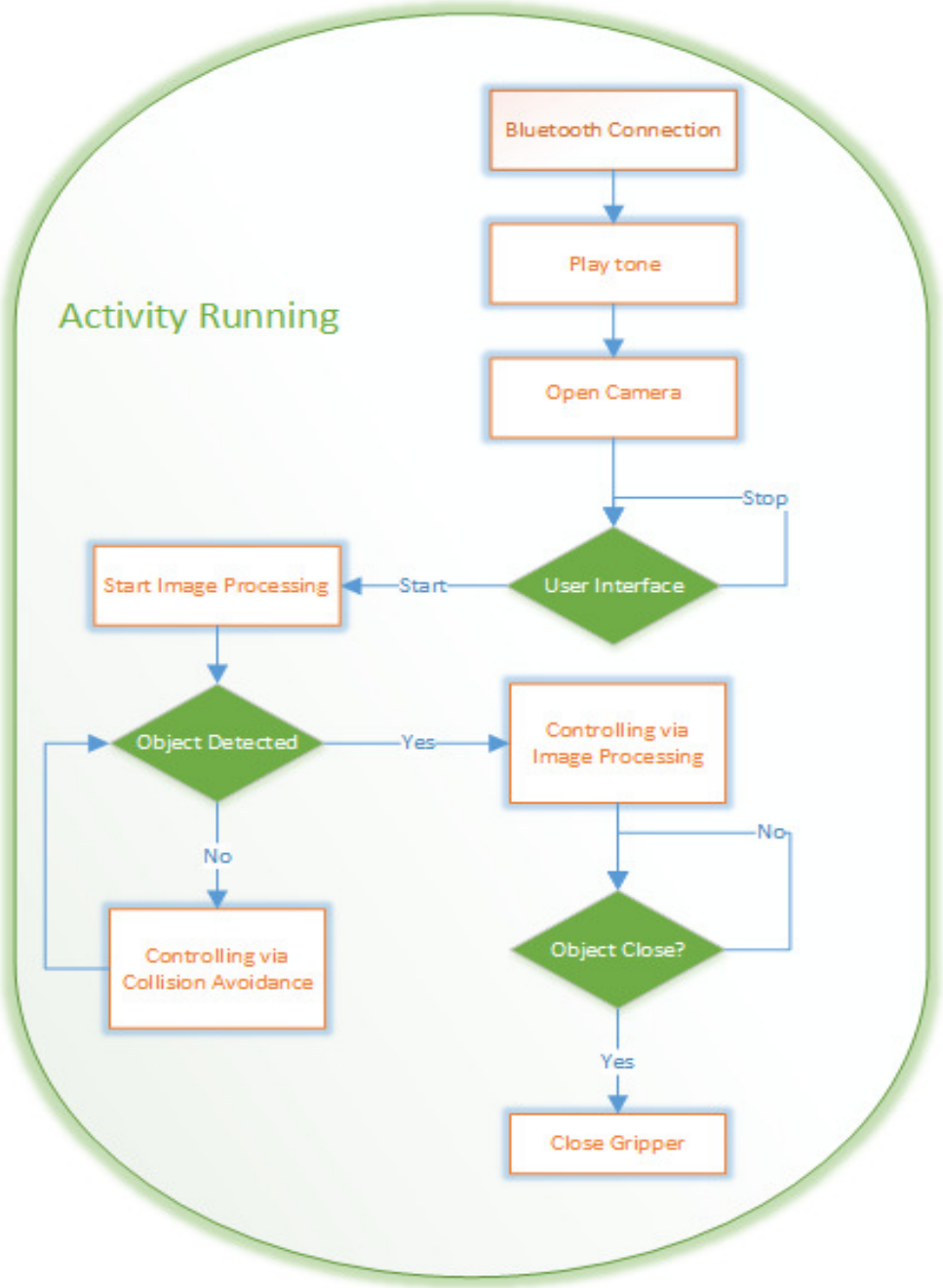
The NxtRobotControl.java class has methods to call these sensor values.

```
public int USMeasure (){
	int Measurement;
	Measurement = myRobot.getUSDistance();
	return Measurement;
}

public int ColorMeasure (){
	int ColMeasurement;
	Measurement = myRobot.getCol();
	return ColMeasurement;
}
```

Above are two methods to receive the sensor values. The USMeasure() is for measuring the distance via Ultasonic sensor, whereas the ColorMeasure() is for identifing the surface color. Both methods call the getter methods that are implemented in NXTBrick.java . `myRobot` is the object of the NXTBrick.java class. To get the sensor value the corresponding sensors must be activated at first by calling the setSensors() setter method in NxtRobotControl.java. Then the getter methods are called to receive values via Bluetooth from the NXT to the Smartphone.

4.2 Application

Once the robot and the Android Apps are built the autonomous robot was tested with an application appropriate to it. In this application the robot should search and detect a colored ball by its own without human external control. To achieve this two controlling methods are implemented in NxtRobotControl.java; controlling via collision avoidance and controlling via Image processing. Below is the flow chart of the application:



Figures 4.3 : Flow chart.

Figure 4.3 shows the application's flow chart, which describes the controlling of the robot. As soon as the Android activity is in running mode or running state the App tries to establish the Bluetooth connection between the Smartphone and the Lego Mindstorms robot.

OcvViewImgProc.java:

```
myRobotControl = new NxtRobotControl();
myRobotControl.init();
```

NxtRobotControl.java

```
public void init() {
// Search for robot and establish bluetooth communication
myRobot.connectRobot();
}
```

The first two code lines in OcvViewImgProc .java create an NxtRobotControl object and calls the init() method. The init() method meanwhile calls the NxtBrick's connectRobot() method to search and establish the Bluetooth communication with the robot. If the connection is successful the Smartphone plays an audio indicating that the Bluetooth connection is successfully established.

The application then opens the camera and sets the camera parameters by calling methods in OcvViewBase.java. The captured images then displayed to the user and the application waits for the input from the user to activate or deactivate the robot. The start and stop menu buttons determine the activation and deactivation of the robot.

Once the start menu button is clicked by the user, the Android App starts the image processing to identify the desired object, which is described in chapter 3.5. The extracted values in chapter 3.5.5 determine the controlling method used to control the robot. Below are code lines in OcvViewImgProc to determine the controlling methods.

```
if (NoContur)
      myRobotControl.ControlColAvoid (stop);
else if(!NoContur)
      myRobotControl.ControlImgProc(stop, xVogRect, wVogRect);
```

`NoContur` is a Boolean variable. It is set to true when the pixel classification does not result in a blob of sufficient size. This allows the Smartphone uses the Collision

Avoidance method to control the robot. If a blob is found, the Boolean variable is set to false and then the Smartphone uses the extracted values from the Image Processing to control the robot. This means as long as the desired object is not in the visible field of the camera, the collision avoidance method will be used otherwise the other controlling method will be used.

4.2.1 Controlling via Collision Avoidance

This chapter will explain the example code to control the robot via collision avoidance. Before we jump to the collision avoidance method I would like to explain an important method in NxtRobotControl.java which is used in both controlling methods.

```
public void Motorcommand(int rotDirect) {
	if ((rotDirect != rotDirectOld) || (motSpeedAd != motSpeedAdOld)) {

		switch (rotDirect) {
		case -1:
			myRobot.MoveMotor(1, motSpeedAd);
			myRobot.MoveMotor(2, - motSpeedAd);
			break;
		case 0:
			myRobot.MoveMotor(1, 0);
			myRobot.MoveMotor(2, 0);
			myRobot.MoveMotor(0, 0);
			break;
		case 1:
			myRobot.MoveMotor(1, -motSpeedAd);
			myRobot.MoveMotor(2, motSpeedAd);
			break;
		case 10:
			myRobot.MoveMotor(1, motSpeedAd);
			myRobot.MoveMotor(2, motSpeedAd);
			break;
		case 20:
			myRobot.MoveMotor(1, -motSpeedAd);
			myRobot.MoveMotor(2, -motSpeedAd);
			break;
		}
	}
	rotDirectOld = rotDirect;
	motSpeedAdOld = motSpeedAd;

}
```

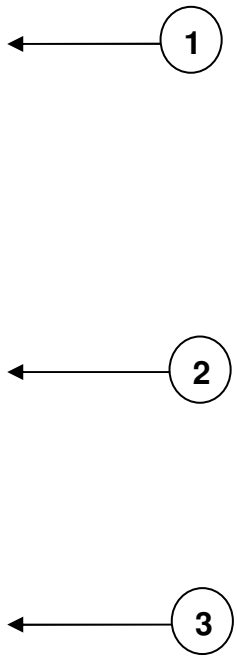
Above is the Motorcommand() method which calls the Movemotor () method from NxtBrick.java to drive the motor according to the values from the image as well as from the ultrasonic sensor. The integer variable rotDirect() determines the direction of

the robot during the drive. Below are the values of the variable and its corresponding direction:

- -1 → Robot turns left.
- 0 → Stop the Robot, All motors stop.
- 1 → Robot turns right.
- 10 → Robot moves straight forward.
- 20 → Robot moves straight backward.

The method calls the Movemotor() method only when there is change in the direction or change in the speed. This is important otherwise the robot reacts slower. Now let's look the example code to control the robot via collision avoidance.

```
public void ControlColAvoid(boolean stop) {

	objectClose = false;

	if (stop)
		rotDirect = 0;
	else {
		if (counter2 < 5) {
			counter2++;

		} else if (counter2 == 5) {
			//Call method to get the Ultrasonic Sensor Value
			Distance = USMeasure();

			if (Distance <= 60 && Distance >= 0) {
				if (Distance <= 40 && Distance > 0) {
					rotDirect = 20;
					motSpeedAd = motSpeed * 2;

				} else {
					rotDirect = 1;
					motSpeedAd = motSpeed *3;
				}


			} else if (rotDirect == 20) {
				counter++;
				if (counter == 2) {
					rotDirect = 1;
					counter = 0;
					motSpeedAd = motSpeed *3;
				}

			} else {
				rotDirect = 10;
				motSpeedAd = motSpeed * 2;
				counter2 = 0;
			}
		}
```



```
        }

        Motorcommand(rotDirect);          <-- 4
}
```

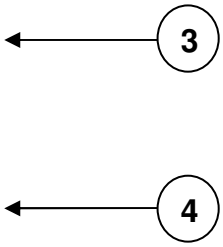
As soon as this method is called, it checks whether the stop menu is clicked or not. If the stop menu is clicked the method set the direction to 0, which stops the motors. Otherwise the ultrasonic sensor measures the distance for every 5 processing frames. If there is hindrance or the distance measured is below 60cm (1) the robot attempts to turn to the right. If the robot is too close to the hindrance or in other words, if the distance is below 40cm, the robot attempts to move backwards in order to turn. Once the direction is set to 20 (2), the robots move backwards a little bit and then turn to right again. If the distance measured is more than 60cm (3) the robot move straight forward. Finally, the Motorcommand() method (4) is called to drive the motors according to the direction.

4.2.2 Controlling Via Image Processing

This chapter explains the example code to control the robot via Image Processing.

```
public void ControlImgProc(boolean stop, int xRect, int wRect) {

            if (stop)
                  rotDirect = 0;

            // adjust the object aligned to robot if the object not close
            // and out of tolerance range
            // turn left
            else if (!objectClose && xRect < (400 - tolValue) && wRect < 200) {
                                                                  <-- 1
                  if (xRect < 200)
                        motSpeedAd = motSpeed*2 ;
                  else {
                        motSpeedAd = motSpeed+10 ;
                  }
                  rotDirect = -1;

            }
            // adjust the object aligned to robot if the object not close
            // and out of tolerance range
            // turn right
            else if (!objectClose && xRect > (400 + tolValue) && wRect < 200) {
                  if (xRect > 500)                                <-- 2
                        motSpeedAd = motSpeed*2 ;
                  else {
                        motSpeedAd = motSpeed+10 ;
                  }
                  rotDirect = 1;
            }
```

```
        // move forward if the object not close
        else if (wRect < 200) {
              rotDirect = 10;
              motSpeedAd = motSpeed * 2;
        }
        // move forward slowly if the object close enough to grab
        else if (wRect > 200 && wRect < 350) {
              rotDirect = 10;
              motSpeedAd = motSpeed - 8;
              objectClose = true;
        }
        // if the object close enough read the sensor value;if the
        // object is green grab the object
        if (objectClose) {

              if (counter < 3)
                    counter++;
              else if (counter == 3) {
                    ColorInt = ColorMeasure();
                    counter = 0;
              }
              if (ColorInt == 3) {
                    rotDirect = 0;
                    myRobot.MoveMotor(0, -motSpeed);

              } else {
                    rotDirect = 10;
                    motSpeedAd = motSpeed- 8;
              }
        }
        if (stop)
              rotDirect = 0;

        Motorcommand(rotDirect);
}
```



The extracted values in chapter 3.5.5 are given (xRect, wRect) to the method above in order to control the robot via image processing. In this method only color sensor value is required to grab the object and the robot is fully drive just with the extracted values from the image. Once this method is called, it checked the stop menu similar to collision avoidance method. If stop menu is clicked the direction is set to 0. Otherwise, the extracted values determine the direction. There are 4 states to determine the direction of the robot. The four states are described below:

1. The object is not close, the detected object located left to the robot and not in line with the robot.
2. The object is not close, the detected object located right to the robot and not in line with the robot.
3. The object is not close and the object is aligned with the robot.

4. The object is close

The four states are illustrated in figure below.

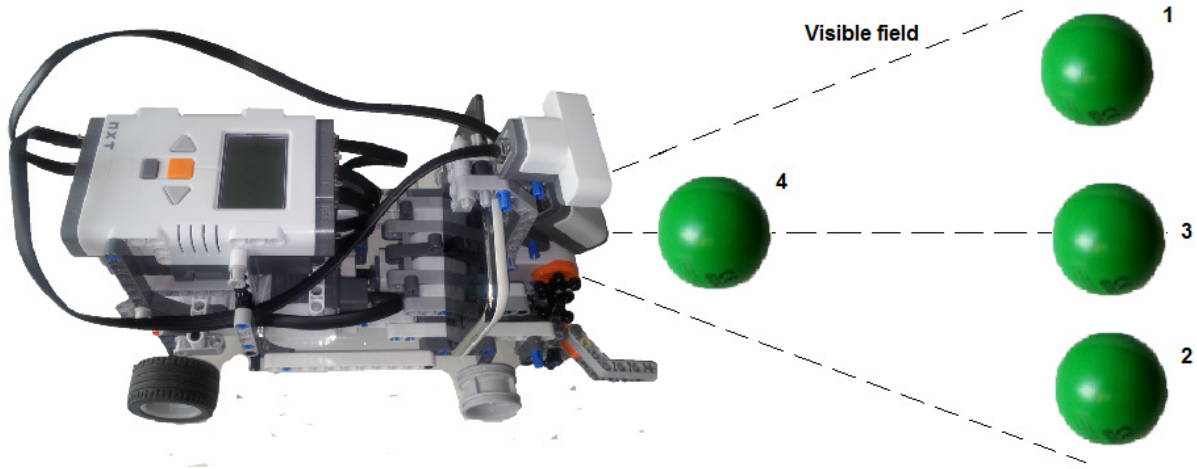


Figure 4.4: The four states to decide the outcome of controlling method via Image Processing.

If the object is at (1) the direction is set to -1; if the object is at (2) the direction is set to 1. If the object is at (3) the robot moves forward. For the 4th state a Boolean variable objectClose is set to true to activate the color sensor. The robot move slowly and for every 3 processing frames the color sensor measures and identifies the surface color. If the color sensor returns the value 3(=green), the gripper closes to grab the object.

4.2.3 Test and Optimization

Initial test result with the controlling methods was not satisfactory as the robot had slow reaction and failed to grab the object when the object was close to it. The problems which caused the robot reacted slower were identified and solution were found to optimize the robot's reaction.

The slow reaction of the robot was caused by two problems:

1. Lower frame rate(FPS)
2. Bluetooth communication

As informed earlier in chapter 3.4, lower FPS is caused by several factors such as the image size, illuminance, and processing methods. Initially the FPS was at 4 FPS. A right combination of FPS factors was chosen and tested to get higher FPS. The image size was set to 680x480, and the image processing methods were optimized to get higher frame rate at 10-13 FPS.

The Bluetooth communication also caused some problems. Sending and receiving a telegram message via Bluetooth needs some times. For an example in collision avoidance method the Smartphone requests the ultrasonic sensor values. To get the sensor values the Smartphone must sent a telegram to the Lego NXT.Then the Lego NXT sent a reply telegram with the values. This process needs some time and the acquisition of sensor values for each frame made the robot reacts slower. Therefore, the program codes were alter in order to measure the distance at given interval or for each 5 frames received, so that the robot not only reacts faster but also identifies the hindrance at the right time.

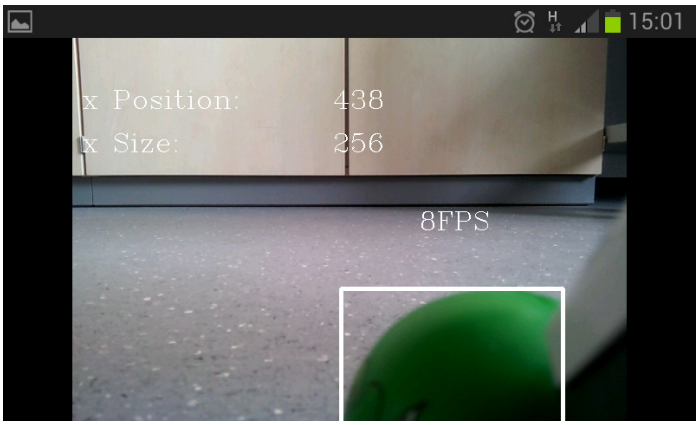


Figure 4.5: Object is partly out of visible field.

Another problem occurred during the test was that the robot failed to grab the object, even though the object closed to it. Initially the values from the image processing were used to grab the object.

When the object is close to the robot part of the object is out of visible field of the camera as shown in the figure 4.5. This leads into misperception and causing problem to grab the object. To overcome this problem the color sensor was attached to the robot to identify the surface color. When the object is close to sensor ,the sensor identifies the green surface color and the robot grabs the object.

Chapter 5:

5.0 Outlook

Although the current version of the autonomous robot has relatively faster reaction compare to its initial reaction, this current version still lack the precision and effectiveness. 90% of the controlling takes place in the Smartphone which means the Smartphone requires the sensor values from time to time and this makes the robot reacts slower. It is a good idea to implement the controlling method in the Lego NXT with its software. By doing this we can save the time penalties which occur during the Bluetooth communication. The Smartphone only performs image processing and sent the extracted values via Bluetooth to the Lego NXT.

As experimented in chapter 3.4, the Android camera class has some advantages over the OpenCV native camera. For further development in the future it is suggested to replace the OpenCV native camera with the Android camera class in order to get better and higher frame rate. Furthermore, the buffer allocation for the android camera may increase the frame rate.

Furthermore the existing application can be expanded, where the robot able to follow arrows and place the grabbed object in a specified location. To achieve this, pattern recognition methods should be implemented with the help of OpenCV so that the Smartphone recognizes the arrows. Those arrows can guide the robot to reach the final destination to drop the object.

Summary

In this thesis work an autonomous robot was successfully built, which is control by an Android based Smartphone. The three main components of the Robot namely the Lego Mindstorms, Android and OpenCV were well studied and combined together in order to create an autonomous robot.

Furthermore, to achieve a satisfactory speed of the robot control, a high image processing FPS rate is very important. For this reason some experiments were conducted on the frame rate which is influenced by the image size, illuminance and image processing methods. Besides that, the differences in frame rate between Android camera and OpenCV native camera were well studied and concluded that the Android camera has some advantages over the OpenCV native camera.

Lastly the controlling methods were written with the help of sensor values and extracted values from image processing. Furthermore the possibility of Bluetooth communication between the Smartphone and the Lego NXT were well studied, implemented and optimized to get an effective communication.

In a nutshell, the objective of this thesis was achieved. Further developments and objectives are recommended to demonstrate the possibility of controlling a robot using Image Processing.

References

[1] ***Lego Mindstorms***
URL: http://mindstorms.lego.com/en-us/products/9844.aspx#8547/ (Last consulted: 24.06.2013)

[2] ***Third Party Sensor***
URL:http://legoengineering.com/thirdpartysensors.html (Last consulted: 24.06.2013)

[3]***Lego Mindstorms EV3***
URL:http://mindstorms.lego.com/en-us/News/ReadMore/Default.aspx?id=476781 (Last Consulted:24.06.2013)

[4] M. Saifizi*, D. Hazry, Rudzuan M.Nor; ***Vision Based Mobile Robot Navigation System;*** International Journal of Control Science and Engineering 2012, 2(4): 83-87

[5] Himanshu Borse, Amol Dumbare, Rohit Gaikwad & Nikhil Lende; ***Mobile Robot for Object Detection Using Image Processing;*** Global Journal of Computer Science and Technology,Neural & Artificial Intelligence,Volume 12 Issue 11 Version 1.0 Year 2012

[6] ***ios vs Android***
URL: http://techland.time.com/2013/04/16/ios-vs-android/ (Last Consulted: 24.06.2013)

[7] ***Creating an Android Project***
URL: http://developer.android.com/training/basics/firstapp/creating-project.html#CommandLine (Last Consulted: 24.06.2013)

[8] ***Activity***
URL:http://developer.android.com/reference/android/app/Activity.html (Last Consulted: 24.06.2013)

[9] ***Starting an Activity***
URL:http://developer.android.com/training/basics/activity-lifecycle/starting.html (Last Consulted: 24.06.2013)

[10] Berthold Klaus Paul Horn: ***Robot Vision***; The MIT Press, McGraw-Hill Book Company, Cambridge, Massachusetts/New York,1987

[11] Gary Bradski, Adrian Kaehler: ***Learning OpenCV, 2nd Edition. Computer Vision with the OpenCV Library;*** O'Reilly Media,Sebastopol, 2012.

[12] Orit Skorka ; Dileepan Joseph;***Toward a digital camera to rival the human eye;*** *J. Electron. Imaging*. 20(3), 033009 (August 19, 2011). doi:10.1117/1.3611015

[13] **URL:**www.opencv.org (Last Consulted: 24.06.2013)

[14] **URL:**http://www.willowgarage.com/ (Last Consulted: 24.06.2013)

[15] ***Lego Mindstroms NXT Communication Protocol; Lego Mindstroms NXT Direct Commands;Lego Mindstorms NXT Ultrasonic Sensor I²C Communication Protocol.***

[16] W.Frank Ableson; Robi Sen; Chris King ;C.Enrique Ortiz; ***Android in Action Third Edition;*** Manning Publication,New York,November 2011

[17] ***ROBOTC for MINDSTORMS-Bluecore Bluetooth Communication***.
URL:http://www.robotc.net/support/nxt/MindstormsWebHelp/index.htm#page=Bluetooth/bluecore_info/Bluecore%20Bluetooth%20Information.htm (Last Consulted: 24.06.2013)

[18] ***Android NDK***
URL:http://developer.android.com/tools/sdk/ndk/index.html (Last Consulted: 24.06.2013)

[19] ***How To Use OpenCV for Tegra;***
URL:http://docs.nvidia.com/tegra/data/How_to_Use_OpenCV_for_Tegra.html(Last Consulted: 24.06.2013)

[20] ***Developing OpenCV computer vision apps for the Android platform;***
URL:http://www.embedded.com/design/programming-languages-and-tools/4406164/Developing-OpenCV-computer-vision-apps-for-the-Android-platform (Last Consulted: 24.06.2013)

[21] ***JavaCV;***
URL: http://docs.opencv.org/java/ (Last Consulted: 24.06.2013)

[22] ***OpenCV4Android SDK***
URL:http://docs.opencv.org/doc/tutorials/introduction/android_binary_package/O4A_SDK.html (Last Consulted: 24.06.2013)

[23] ***Camera***
URL:http://developer.android.com/guide/topics/media/camera.html (Last Consulted: 24.06.2013)

[24] ***Multithreading and Concurrent Programming***
URL:http://www.ntu.edu.sg/home/ehchua/programming/java/J5e_multithreading.html (Last Consulted: 24.06.2013)

[25] ***Properties and Concepts of Light and Color***
URL:http://www.light-measurement.com/properties-and-concepts/ (Last Consulted: 24.06.2013)

[26] Peter Corke; ***Robotics, Vision and Control;*** Springer, Berlin,2011.

[27] ***HSL-HSV models***
URL:http://en.wikipedia.org/wiki/File:Hsl-hsv_models.svg (Last Consulted: 24.06.2013)

[28] ***Analysing I²C communication***
URL:http://www.generationrobots.com/analysing-i2c-communication-with-saleae-logic-analyser,us,8,83.cfm (Last Consulted: 30.06.2013)

[29]Keng T.Tan, Siong Khai Ong, Doughlas Chai; ***JPEG color barcode images analysis: A camera phone capture channel model with auto-focus***;International Journal of Signal Processing, Image Processing and Pattern Recognition Vol.2 No.4, December ,2009
URL:http://www.sersc.org/journals/IJSIP/vol2_no4/4.pdf (Last Consulted: 02.07.2013)

[30]Ive Billiauws, Kristiaan Bonjean; ***Image recognition on an Android mobile phone;*** De Nayer Instituut
URL:http://www.eavise.be/mastertheses/BilliauwsBonjean.pdf (Last Consulted: 02.07.2013)

[31] Samantha Patricia Bail; ***Image Processing on A Mobile Platform;***
URL:http://studentnet.cs.manchester.ac.uk/resources/library/thesis_abstracts/MSc09/FullText/BailSamantha.pdf (Last Consulted: 02.07.2013)