AndRo.bot – HTTP and SMS controlled Vision and Voice based Robotic software for Android

Pradeep N., Syed I. Tauhidi

***Abstract* – This project uses Android, along with cloud based APIs and the 80c51 microcontroller, to create a software-hardware system that attempts to address several current problems in robotics, including, among other things, line-following, obstacle-avoidance, voice control, voice synthesis, remote surveillance, motion & obstacle detection, face detection and remote live image streaming. The rover, built upon a portable five-layered architecture, will use several mature and a few experimental algorithms in Web and Mobile Development, Computer Vision and Robotics behind-the-scene to provide its prospective user in defense, entertainment or industrial sector a positive user experience. A minimal amount of security is achieved using MD5, SHA1 and AES.**

***Index Terms* - Voice Synthesis, Voice Control, Robot-User Interaction, Face Detection, Motion and Object Detection, Remote Surveillance, Object Tracking, Line-following and Obstacle-Avoiding Robotics, Computer Vision, Image Processing, Robotic Network Security.**

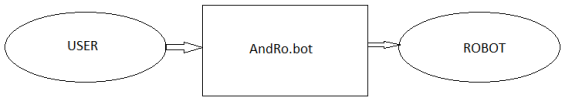
I - INTRODUCTION

It is usually seen in hobbyists' robotics that a highly complex hardware system is created to provide a few basic functionalities. This has resulted in robots that are expensive in price, but limited in functionalities. In our project, we try to build a flexible, robust, secure and high performance software platform that would shift the complexity from hardware to software in hobbyist robotics projects and enable the implementation of a wide range of features.

The AndRo.bot software runs on the Android software stack. It communicates with the user using HTTP or SMS and with the robot hardware using Bluetooth. It makes no use of any sensor, apart from the sensors that comes built-in with the Android smart-phone. Some of the internal features that make this platform preferable over stand-alone solutions are:

1. MD5 and SHA1 based password authentication
2. Built-in HTTP server

3. RIA based Human-Robot-Interaction (HRI) using HTML5 / CSS3 / JQuery
4. SMS based HRI
5. Uses latest version of openCV for Computer Vision
6. Client-side Real-time IP
7. Customizable using Android Preference



[Fig.1 – High Level Architecture]

The software is made as general as possible and it enables the user to implement any type of algorithm that he may require for performing his tasks. The ability of the software platform is demonstrated by implementing these core functionalities:

- Live Image Streaming
- Face Detection
- Line Following
- Object Tracking
- Automatic Guided Vehicle (AGV)
- Voice Synthesis
- Voice Recognition
- Motion/Proximity/Magnetic Detection
- Google Maps based mapping

Live Image Streaming, which refers to images delivered live over the Internet, requires a camera for the media, an encoder to digitize the content, a media publisher, and a content delivery network to distribute and deliver the content.

Face Detection (FD) is a computer technology that determines the locations and sizes of human faces in arbitrary (digital) images. It detects facial features and ignores anything else, such as buildings, trees and bodies.

Line-following robots are some of the earliest Automatic Guided Vehicle (AGVs). They might follow a visual line painted or embedded in the floor or ceiling or an electrical wire in the floor. Most of these robots operated a simple 'keep the line in the center sensor' algorithm. They could not circumnavigate obstacles; they just stopped and waited when something blocked their path.

Object Tracking is the process of locating a moving object (or multiple objects) over time using a camera. It has a variety of uses, some of which are: human-computer interaction, security and surveillance, video communication and compression, augmented reality, traffic control, medical imaging and video editing. Video tracking

can be a time consuming process due to the amount of data that is contained in video.

Tele-operation indicates operation of a machine at a distance. It is similar in meaning to the phrase "remote control" but is usually encountered in research, academic and technical environments. It is most commonly associated with robotics and mobile robots but can be applied to a whole range of circumstances in which a device or machine is operated by a person from a distance. *Automatic Guided Vehicle* is one that can be tele-operated using remote devices.

Speech or *voice synthesis* is the artificial production of human speech. A computer system used for this purpose is called a speech synthesizer, and can be implemented in software or hardware products. A text-to-speech (TTS) system converts normal language text into speech; other systems render symbolic linguistic representations like phonetic transcriptions into speech.

In computer science, speech or *voice recognition* is the translation of spoken words into text. It is also known as "automatic speech recognition", "ASR", "computer speech recognition", "speech to text", or just "STT". Some SR systems use "training" where an individual speaker reads sections of text into the SR system.

Motion detection is the process of detecting a change in position of an object relative to its surroundings or the change in the surroundings relative to an object. Magnetic detection refers to the detection of change in angular orientation or the introduction of any electromagnetic field. Proximity detection detects when an object comes in the vicinity of the device.

Mapping usually refers to map-making and often used instead of cartography.

II – MOTIVATION AND BAKGROUND

Traditionally, the development of robots has required a wide range of complexity in hardware. The reception of a data from external environment required the usage of sensors to detect factors like electromagnetic spectrum, sound, touch, chemical sensors, temperature, range to things in the environment and attitude.

In recent years, the field of computer science has seen rapid developments in the fields of Computer Vision and in the construction of handheld computers (e.g., smart-phones, tablets, etcetera.). This can largely be attributed to Moore’s Law, which is the

observation that over the history of computing hardware, the number of transistors on integrated circuits doubles approximately every two years. This doubling of transistors has roughly made possible the doubling of the computing power of devices and softwares running on the Von Neumann architecture. Examples of this phenomenon are the aforementioned developments in the fields of computer vision and handheld computers.

Computer Vision is a field that includes methods for acquiring, processing, analyzing, and understanding images and, in general, high-dimensional data from the real world in order to produce numerical or symbolic information, *e.g.*, in the forms of decisions. A theme in the development of this field has been to duplicate the abilities of human vision by electronically perceiving and understanding an image. This image understanding can be seen as the disentangling of symbolic information from image data using models constructed with the aid of geometry, physics, statistics, and learning theory. Computer vision has also been described as the enterprise of automating and integrating a wide range of processes and representations for vision perception. Because construction of computer vision algorithms is a system-level task, most application developers use libraries for such tasks. OpenCV is an example of such a library.

OpenCV (*Open Source Computer Vision Library*) is a library of programming functions mainly aimed at real-time computer vision, developed by Intel, and now supported by Willow Garage and Itseez. It is free for use under the open source BSD license. The library is cross-platform. It focuses mainly on real-time image processing. The library is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface. There are now full interfaces in Python, Java and MATLAB/OCTAVE (as of version 2.5). The API for these interfaces can be found in the online documentation. Wrappers in other languages such as C#, Ch, Ruby have been developed to encourage adoption by a wider audience.

Similar developments have also been seen in the construction of handheld computers. A Handheld PC is a computer built around a form factor which is smaller than any standard laptop computer. It is sometimes referred to as a Palmtop. The first handheld device compatible with desktop IBM personal computers of the time was the Atari

Portfolio of 1989. Another early model was the Poqet PC of 1989 and HP 95LX of 1991. Other DOS compatible hand-held computers also existed. Some Handheld PCs use Microsoft's Windows CE operating system. Devices such as smart-phones and tablets running on operating systems like iOS or Android are newer examples of handheld computers. Because of the cost ineffectiveness and the proprietary nature of the devices running iOS, Android has emerged as the better option among the lot.

Android provide a rich API and a user-friendly development environment for application software development. Android 'apps' run on the Dalvik Virtual Machine, which is basically a FLOSS implementation of the Java Virtual machine. This makes possible the usage of a large number of modules for Java. Also, usually Android devices come with several sensors embedded into them.

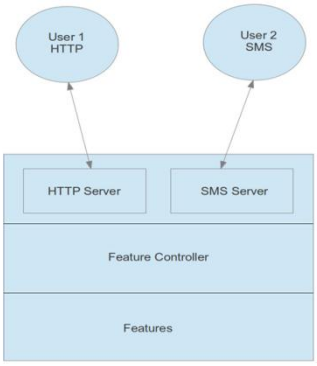
Robotics has traditionally depended, as aforementioned, on sensors and hardware complexity. In the last few years, robots have shifted complexity from hardware to software (e.g., Honda Asimo, Curiosity Mars Rover, etc.). Because of the said developments of in the field of computer vision and handheld computing, we were motivated to experiment with the possibility of using Android only for the construction of a robotic software.

In this project, we use the Java port of OpenCV and nanoHTTPD on Android to create the required vision and HTTP communication mechanisms respectively. We also use Android TTS, Google's cloud-based Voice Recognition APIs and Broadcast Receivers to set up the voice synthesis, voice recognition and SMS communication respectively. The Android smart-phone, which is actually the 'brain' on the robot, communicates with the robot hardware using Bluetooth. The robot hardware contains a HC05 Bluetooth Serial module, a 80c51 micro-controller, an L293 motor driver and two DC motors.

III – SYSTEM DESIGN

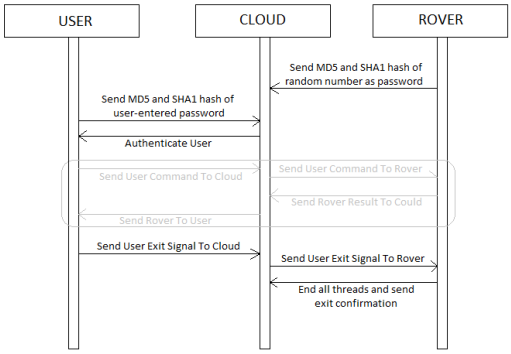
As already mentioned, the system can be accessed by two types of users – using SMS or HTTP. The command received from the user is handled by the 'feature controller', which activates/deactivates the required features.

The figure below shows another interpretation of the high-level architecture, in which the two users are handled by two servers in two processes that receives the user's input and passes the input commands over to the feature controller.



[Fig. 2: High Level System Architecture]

A minimal amount of security, too, is provided. We use SHA1 and MD5 for password authentication and AES for data encryption. The authentication method is demonstrated using the following diagram.



[Fig 3 – Authentication Model]

In our project, we are dealing with two types of inputs – one type from the user to control the robot and the other from the Android's sensors about the environment.

The user can provide input in one of the two ways:

- Textual input using SMS
- Graphical input using the RIA

The SMS server takes three types of inputs that can be provided to get various informations about the device. The legal SMS formats that are taken as inputs are enumerated below:

- GET IP
- GET LOC
- GET STAT

The RIA is a full GUI-based interface that allows the user to provide input by pointing and clicking. The TTS is provided textual input using the standard input.

Apart from the user inputs, the software also makes use of inputs provided by the Android sensors. The main sensors from the Android smart-phone/tablet used for the project are:

- Camera
- GPS
- Compass
- Accelerometer
- Proximity Meter
- Battery Meter
- Thermometer

Voice Recognition uses human voice as input – to convert it to text. In Blob/Object Detection, touch is used to initially pin-point the object to track or blob to follow.

The output is provided in three manners as enumerated below:

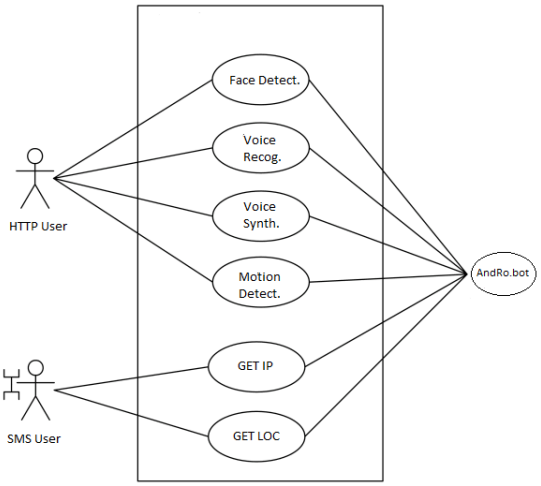
- SMS
- HTTP
- Bluetooth

SMS is used as an output only when the user uses SMS to provide input. If the input is GET IP, the software provides the IP of the device along with port on which the software is running. GET LOC is responded to by replying with an SMS containing the latitude and longitude of the device. Similarly, GET STAT is responded to by providing the battery and temperature status of the device to the sender.

HTTP provides the main interface for the user. It is made as an RIA. The images are sent as text encoded using base64. These images are the rendered on the RIA sequentially using HTML5 Canvas. Texts and other data are provided to the RIA using JSON. JSON or JavaScript Object Notation, is a text-based open standard designed for human-readable data interchange.

Bluetooth is used to provide output to the peripheral robotic device.

A Use-Case Diagram showing the interaction of the two users (actors) with the various features of the system is given below.



[Fig 4 – Use Case for Two Actors**]**

IV – IMPLEMENTATION

1) Live Image Streaming

- Uploads live images from the Android camera to the user
- Images encoded with base64 for transfer over HTTP

2) Face Detection

- Detects a human face using Android camera

- Computer Vision using openCV
- Sets up Feature Detection' using 'Haar Cascade' and draws a rectangle in around the detected face
- Not very accurate, but decent and relatively fast

3) Line Following

- The robot moves over a line of high contrast relative to the background
- Uses openCV for Computer Vision
- Sets up a Color Blob Detector and tries to keep the blob in the center
- Done so by measuring position, length and breadth of contours of the detected colors' and then we find centre of contours:

$$X = X_{POS} + HEIGHT/2$$

$$Y = Y_{POS} + WIDTH/2$$

We try to keep the center on the middle of the image.

4) Object Tracking

- Tracks a colored object in 3D space
- Uses algorithm similar to line following
- Tracks in Z axis by looking at contour size – if increases, object moving near; if decreases, object moving away

5) AGV / RC Robot

- In this the robot can be controlled via. the Android phone remotely using the Android's built-in accelerometer.

6) Voice Synthesis

- The robot reads aloud any text provided remotely by the user
- This is implemented using the Android's TTS server

7) Voice Recognition

- This allows a person to speak words into the Android phone which would be converted into text and displayed to the user
- It is implemented using cloud-based services through Android APIs

8) Motion/Proximity/Magnetic Detection

- Motion means any vibration to the phone and is detected using the built-in accelerometer
- Proximity is detected behind the camera using the proximity sensor
- Magnetic compass is used to detect change in angular orientation of the Android device or the introduction of electromagnetic fields in the vicinity

9) Google Maps based mapping

- Locates the position of the phone using Google Maps
- Uses Google Map JavaScript APIs and Android's built-in GPS sensor
- Centers the map on the GPS provided co-ordinates of the phone.

V – PROSPECTIVE USAGE

Being general-purpose, we can use our software for a wide variety of tasks. But some of the most common tasks that can be easily performed are:

- With extra robotic hardware
    - Military
    - Entertainment
    - Civilian
- Without hardware
    - Photography
    - Security (as an intercom)
    - Research

The usage can be improved by implementing any algorithm that the situation may demand. This is further possible because of the ease with which newer algorithms can be implemented.

VI - CONCLUSION

The software needs high bandwidth connection between user and robot, else performance degrades.

Sensor Processing and Computer Vision are processor intensive. This can be improved using faster, multi-core processors. We can use GPU for even faster processing. We can even run C code directly using JNI, since C is about 10x faster than Java (even with JIT).

Static rules cannot perform in all (esp. unforeseen) situations. This defect can be corrected by replacing static rules them with soft computing techniques (ANNs, neuro-fuzzy, GAs, EAs) for higher automation.

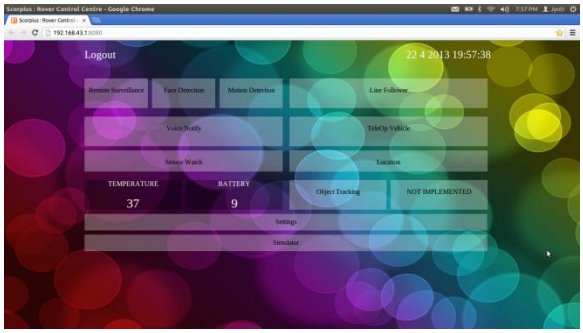
Soft Computing Techniques can easily be implemented in OpenCV using the built-in Machine Learning module.

Another enhancement would be to create a full-duplex link between the software and the peripheral robotic device. This would allow the peripheral robot to provide feed-back to the device. This would result in more accuracy in the algorithms.

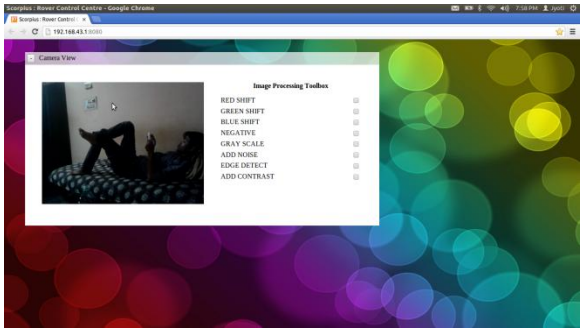
Finally, we can build a custom ROM by forking a version of Android and including OpenCV Manager and our software by default. This would result in a modified Android OS optimized for robotics.

VII – PHOTOS / SCREENSHOTS

[Fig 5 – The Robot Hardware]



[Fig 6 – RIA Control Panel]



[Fig 7 – RIA Real Time Streaming]

VIII – REFERENES

- Green, S.A., Billinghurst, M., Chen, X., Chase, G.J. (2008, *Human-Robot Collaboration: A Literature Review and Augmented Reality Approach in Design*. International Journal of Advanced Robotic Systems, 5(1), pp. 1-18
- *Machine Vision Giving Eyes to Robots: Resources in Technology*. (March 1990, Technology Teacher, 49: 6, 21-28.
- Braggins, Don. (1995, *'A critical look at robot vision'*. The Industrial Robot, 22(6): 9-12.)
- Grimson, W.E.L. and J.L. Mundy. (March 1994, *'Computer Vision applications'*. Communications of the ACM, 37(3): 44-51
- Ken Taylor and B. Dalton(1997, *'Issues in Internet Telerobotics'*. In International Conference on Field and Service Robotics (FSR 97)).
- *Learning openCV Computer Vision and the openCV library* - Gary Bradski and Adrian Kaehlar
- Robert Laganière, *OpenCV 2 - Computer Vision Application Programming Cookbook*

Pradeep N. is a Lecturer in the Dept. of C.S.E. at SSIT, Tumkur. His interests include Computer Vision, Operational Research, Digital Logic, Artificial Intelligence and Soft Computing.

Syed I. Tauhidi is a student of B.E. (C.S.E.) at SSIT, Tumkur. His interests include Robotics, Computer Vision, UI Design, Human Robot Interaction and Artificial Intelligence.