Authors:
Sonal Raj (CS110487)
Shashi Kant (CS110462)
Guide: Mr. Sanjay Kumar
Dept.: Computer Science & Engineering
Institute: NIT, Jamshedpur

SPOCK MANUAL

BTP PROJECT DOCUMENTATION. 2014

This is a layman's guide and the official documentation plus user manual for the Human Computer Interaction System, Spock.

TABLE OF CONTENTS

Contents

INTRODUCTION

Introduction

WHAT IS SPOCK?

SPOCK is the human-computer interaction system which aims to make computer system works at out fingertips literally. SPOCK took inspiration from SIRI, Google Now and Microsoft's Cortana and provides best of all three. But unlike Siri, Google Now and Cortana, SPOCK works on all platforms including mobile, desktop and web. SPOCK widens the possibility of computer system utilization by providing even more modes of interaction viz. speech, gesture etc. It slashes the need of reaching the computer system for its use. It makes computer system more human friendly.

WHY DO WE NEED IT?

SPOCK is the computer system of future. Existing human-computer interaction system are either not efficient enough or they are closed source. Today very few such systems works efficiently (like Google Now and Cortana) and they all are tight closed to their developer companies only. Developers can't modify or customize them for a particular use. This has made such systems available only to people who can afford to burn their pocket and restricted the access more needy people. SPOCK is an open source human-computer interaction system which can be used to by developers all over the world to work on it and make it more feature rich and more users friendly. It can be customized to integrate and work on specific systems and environment. SPOCK is only existing system in the world which is open source and integrates speech synthesis, vision system, text recognition, gesture recognition. There are other promising systems but they lack the richness of the domain which SPOCK covers. SPOCK is also a marvelous project for student to improve its code base and add more features or customize it for a particular environment.

IMPACT OF THE PROJECT

SPOCK draws its users from simple computer science student to computer scientist, a simple layman computer user to an advanced computer system used by major firms and

organizations. SPOCK is destined to be the next big thing in the field of computer technology development. SPOCK gives whole new prospective to use computer system. Leaving the conventional ways of interaction with computer system SPOCK opens up more ways to harness the ever growing power of computer system.

Today, computer systems are generally used by educated and trained personnel. SPOCK gives power to the uneducated, untrained and physically handicapped people to use computer system. People don't need to remember all those difficult commands, shortcuts, tricks and methods to work on a system efficiently. Just a friendly conversation with computer system is sufficient to the computer system to perform all tasks of your need. Open Source nature of SPOCK guarantees the attraction of more developers, more feature addition and more stable system in future.

Since it is open source, we are seeing the widening domain of its uses. SPOCK in the near future is going to give more freedom and power to normal as well as physically challenged people. Even a blind person can use a computer system using SPOKE feeling as he/she is talking to his/her friend while his/her work is being done by the system. SPOKE is not just a common man's tool; it can be used by big enterprises, organizations, companies, government organizations, security firms, vigilance system, threat detection system, network analysis system and as a system control mechanism. For example, it can be used to filter out the telephonic conversation for potential threats by filtering specific words, tones of speech and analyzing person's background history.

PROJECT SUMMARY

Project Summary

In this project, we work on a speech recognition and voice – control engine initially inherent to the UNIX environment. This proposed engine would serve as a framework for several command tools for purposes including File Management, User operations of the OS, Basic Dictation for Word processors or similar software, Web Browsing, etc. The above functionalities are introductory uses for the proposed engine.

This would provide a more elegant way of Human-Computer Interactions on a Unix Platform thereby increasing the User Experience. Moreover, the tools developed with this could enable physically challenged users to manage their operations more easily.

We have based our work mostly in C/C++, Python, JavaScript and Java. The platform used is UNIX. We use OpenCV with encoded custom filters, and PocketSphinx improving the accuracy compared to existing alternatives. The Speech Recognition algorithms and bindings will be based on the work done under the Sphinx project at Carnegie Mellon University. Other tools will be updated as and when required in the future builds.

The project holds great prospects and is definitely a large one, but, we present a working minimal implementation prototype as our BTP work. The code is Open Source and is released under the MIT License.

How to install and configure Spock

OPERATING SYSTEM SETUP

The algorithms developed have been customized and suited for the Linux Operating System, with most of the development and testing work being performed on Ubuntu 13.10. Hence, it is a viable choice for a platform for Spock. Although, it can well be run on any mod of the Linux Operating System or Mac.

Windows binaries are not available as of now but you can still run the system in Cygwin based environment in your windows. But, discrepancies due to device drivers for cameras and microphones have been known to be the sources of error for some devices.

DEPENDENCIES CHECK

The following libraries/ packages are required in the Linux System for the gesture recognition part of the project to function properly.

- gcc, g++, make
- Emscripten
- opencv-doc,
- libcv2.1,
- linhighgui2.1, libcvaux2.1,
- libcv-dev,
- libcvaux-dev,
- linhighgui-dev,
- libx11-dev, and
- libxtst-dev

For the speech synthesis mechanisms to operate in a proper fashion, the following packages need to be installed:

- python-pocketsphinx
- pocketsphinx-hmm-wsj1
- pocketsphinx-lm-wsj

INSTALLATION PROCEDURE

Implementing the code is very simple. However sometimes executable files do not run correctly, in which case the code has to be compiled before running. The packages required for compiling the code are gcc, opencv-doc, libcv2.1, linhighgui2.1, libcvaux2.1, libcv-dev, libcvaux-dev, linhighgui-dev, libx11-dev, and libxtst-dev. These packages can be collectively installed from the Synaptic Package Manager or using individual system commands:

```
sonal@linuxbook-pro# sudo apt-get install [package name]
```

After installing all the packages, we now need to compile the files in the folder, CV. Open the folder, and start a command prompt in that folder. Then, ensure that you have gcc and g++ installed. Then use the following.

```
sonal@linuxbook-pro# g++ install.cpp -o install
```

Running the file install in turn compiles all the other required files, provided the required libraries are installed correctly and up-to-date.

You then need to run the shell script generated to configure your system with the spock system's vision component.

```
sonal@linuxbook-pro# ./install
```

HOW TO INSTALL AND CONFIGURE SPOCK

The Speech Component of Spock can be installed using the following procedures. We need to install the sphinx packages before we can proceed with it:

- python-pocketsphinx
- pocketsphinx-hmm-wsj1
- pocketsphinx-lm-wsj

For this the following commands need to be typed into the command prompt.

```
sonal@linuxbook-pro# sudo apt-get install python-pocketsphinx
```

```
sonal@linuxbook-pro# sudo apt-get install pocketsphinx-hmm-wsj1
```

```
sonal@linuxbook-pro# sudo apt-get install pocketsphinx-lm-wsj
```

Once you are done we can move on with our python script for speech recognition to be used in Spock when you run it.

OpenCV

WHAT IS OPENCV

OpenCV (Open Source Computer Vision Library) is an open-source BSD-licensed library that includes several hundreds of computer vision algorithms. The document describes the so-called OpenCV 2.x API, which is essentially a C++ API, as opposite to the C-based OpenCV 1.x API. The latter is described in opencv1x.pdf.OpenCV has a modular structure, which means that the package includes several shared or static libraries. The following modules are available:

- **core** - a compact module defining basic data structures, including the dense multi-dimensional array Mat and basic functions used by all other modules.
- **imgproc** - an image processing module that includes linear and non-linear image filtering, geometrical image transformations (resize, affine and perspective warping, generic table-based remapping), color space conversion, histograms, and so on.
- **video** - a video analysis module that includes motion estimation, background subtraction, and object tracking algorithms.
- **calib3d** - basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence algorithms, and elements of 3D reconstruction.
- **features2d** - salient feature detectors, descriptors, and descriptor matchers.
- **objdetect** - detection of objects and instances of the predefined classes (for example, faces, eyes, mugs, people, cars, and so on).
- **highgui** - an easy-to-use interface to video capturing, image and video codecs, as well as simple UI
- **capabilities.gpu** - GPU-accelerated algorithms from different OpenCV modules.... some other helper modules, such as FLANN and Google test wrappers, Python bindings, and others.

OPENCV

UTILIZATIONS

OpenCV(Open source Computer Vision) is a library of programming functions for real time computer vision. Its functions are written in C/C++ which make it much faster than other vision library. are closer to directly provide machine language code to the computer to get executed. So ultimately you get more image processing done for your computers processing cycles, and not more interpreting. As a result of this, programs written in OpenCV run much faster than similar programs written in Matlab. So, conclusion? OpenCV is damn fast when it comes to speed of execution. For example, we might write a small program to detect peoples smiles in a sequence of video frames. In Matlab, we would typically get 3-4 frames analysed per second. In OpenCV, we would get at least 30 frames per second, resulting in real-time detection.

OpenCV is highly portable and can be used efficiently on Windows, Unix, GNU/Linux or Mac system. With OpenCV we can get away with as little as 10 MB of RAM.

DRAWBACKS AND SHORTCOMINGS FOR OUR PURPOSE

OpenCV is based on C. As such, every time we allocate a chunk of memory we have to release it again. If we have a loop in our code where we allocate a chunk of memory in that loop and forget release it afterwards, we get what is called a “leak”. This is where the program use a growing amount of memory until it crashes from no remaining memory.

OUR MODIFICATIONS

For a huge and complex system like SPOKE we needed fast, efficient, portable vision library which can work on very low memory. We tried several libraries like MATLAB but since it is programmed on Java it is inherently slower than the library developed using C/C++. OpenCV turned to be our friend in search of apt vision library. OpenCV is highly portable and can be used efficiently on Windows, Unix, GNU/Linux or Mac system. With OpenCV we can get away with as little as 10 MB of RAM. OpenCV is released under a BSD license and hence it’s free for both academic and commercial use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed

for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform. Adopted all around the world, OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 7 million. Usage ranges from interactive art, to mines inspection, stitching maps on the web or through advanced robotics.

We have implemented an additional background filter to the OpenCV module under consideration and then optimized it to operate in noisy areas which will only decipher the gestures mean to be recorded in a 2-dimensional plane of reference that is closest to the point of observation. In this respect, our filtering technique also increases the efficiency of the existing OpenCV algorithms that are being used in this project as a whole.

We have also ported the important aspects of OpenCV to JavaScript so that it can be used in the client side of web applications in modern web browsers. We have integrated it with the Chrome API, and the one for moz/Firefox is in the pipeline.

PERFORMANCE BOOSTS

Compared to the traditional Algorithms of OpenCV to be used in the Biometric Applications, our modified system now improves the sensitivity of the detection system to a high degree along with efficient pattern recognition.

Moreover, the JavaScript run version of the OpenCV libraries can now enhance the functionality of modern websites and web applications giving them extended client side functionality and improved interaction with users. Imagine now you can update your Facebook status by simply saying what you feel. Who needs Emoticons any more ☺

Pocket Sphinx

WHAT IS POCKET SPHINX

PocketSphinx is a small-footprint continuous speech recognition system, freely licensed under a simplified BSD license, suitable for handheld and desktop applications. PocketSphinx is written in C/C++ which works similar to CMUSphinx which is written in java. Lots of research papers and Ph.D. thesis have been written using PocketSphinx as it is actively used in speech recognition research Enhancement engine uses Sphinx library to convert the captured audio. Media (audio/video) data file is parsed with the Content. Item and formatted to proper audio format by Xuggler libraries. Audio speech is than extracted by Sphinx to 'plain/text' with the annotation of temporal position of the extracted text. Sphinx uses acoustic model and language model to map the utterances with the text, so the engine will also provide support of uploading acoustic model and language model.

UTILIZATIONS

- Cross-platform: Linux, Windows, Mac OS X, iOS
- Experimental support for Nokia S60v3 and Windows Mobile
- Support for semi-continuous, phonetically-tied, and fully continuous acoustic models
- Model footprint on disk of about 10MB per language
- Memory footprint under 20MB for medium-vocabulary continuous recognition
- Trigram language models and JSGF finite-state grammars
- Acoustic models for English and Mandarin
- Small language models for English and Mandarin (simplified and traditional characters)
- Python language bindings

- GStreamer multimedia framework integration

APPLICATIONS

PocketSphinx is very small in size and has very small footprint on the system. Because of its lightweight nature, it can be used by systems with low memory and low processing power like mobile devices. PocketSphinx is written in C/C++, which makes it faster, low memory consumer and apt to be used on any system.

WHAT WE CHANGED

PocketSphinx is available for desktop computing environment like Unix, GNU/Linux, Windows, Mac. But People have to install it before they can utilize its enriched potentials. We made it to be used on web also. It has been a long dream to voice-enable websites. However, no good technology existed for this either because speech recognition on the web required a connection to a server or due to the requirement to install binary plugin. No need for installation, no need to maintain voice recognition server farm. This is a really cool technology. Another developer Sylvain Chevalier has been working on a port of PocketSphinx to JavaScript using Emscripten. Combined with the Web Audio API, it works great as a real-time recognizer for web applications, running entirely in the browser, without plug-in. We proceed with the word already done by Chevalier and made few improvements in the JavaScript code of PocketSphinx.

PERFORMANCE BOOSTS

By converting the PocketSphinx into web format, we have already made it more portable and more accessible to users. Now power of PocketSphinx can be harnessed by anyone without going through the hassle of installing CMUSphinx, PocketSphinx and the speech engine. It saves lots of memory on the users system and provides the facility of using speech engine on the fly. Another very major improvement dome by us is slashing the need of speech server as all computations are done on the client's machine. We used Emscripten and Web Audio API to make the PocketSphinx into JavaScript file. Another major

improvement in the JavaScript version of the PocketSphinx is, now it can be used on all major web browsers. We reduced the memory consumption by web browsers while doing computation for speech recognition. We have also reduced the speech dictionary to fill only important words and commands which has reduced the searching and mapping time much lower.

Spock API

WEB APIS FOR ONLINE VISION

Spock uses OpenCV bindings for Node.js JavaScript Framework. OpenCV is the defacto computer vision library - by interfacing with it natively in node, we get powerful real time vision in JavaScript. People are using node-OpenCV to fly control quadrocoptors, detect faces from webcam images and annotate video streams. The Spock API can be used to track live streaming feeds for movements, patterns and gesture which are then interpreted as actions for a given webpage.

WEB APIS FOR ONLINE SPEECH SYNTHESIS

For speech synthesis on the web we have converted the PocketSphinx into its JavaScript version (called JSphinx) so that it can easily be used on the web. Emscripten and Web speech API has been used to convert the C coded PocketSphinx into JSphinx API. JSphinx works on the client side and thus slashes the need of speech synthesis server since all speech synthesis is done of the client machine itself. JSphinx API can be added to any web page as a JavaScript file with few other supplementary files to enable a web page for speech recognition

SYSTEM SPOCK APIS

The System Spock API can be used both as a kernel integrated module for open source operating systems as well can function as a third party application to control random movements, system commands as well as user interaction.

This API, due to its open source nature can be used by application developers to be customized according to the needs of a specific application or a target audience and can be bundled separately along with the application itself. This will led to highly efficient and interactive applications and help them to reach a wider domain of audience.

LIABITITES

JSphinx is aimed to give all speech recognition capability on the web and at the client side. This limits the performance of JSphinx to the limits of user's machine. Any user with slow and old system may experience larger time lag in speech recognition. Since JSphinx is loaded at the time of page load on network, complete speech dictionary of CMUSphinx can't be used efficiently. Complete speech dictionary is very large in size and may consume very large bandwidth to be loaded on user's machine. This limits the size of speech dictionary in the comfortable zone by only incorporating important words and commands in the speech dictionary. Speech recognition also depends heavily on user's way of pronouncing a word. People of different countries speak same words in different fashion.

PERFORMANCE

With ever increasing performance capabilities of processor, memory, bandwidth and other components of computer system, performance of SPOCK is guaranteed to be get better with each passing year. Current performance of SPOCK on system outperforms many other open source speech recognition system and vision control system. However it get a tough competition from Google Now and Cortana but it is important to note that SPOCK is not just a assistant program but it is comprised of whole bunch of different types of applications bundled together. SPOKE API for web used custom made JSphinx API for speech recognition and OpenCV for sensing and implementing gesture recognition. Web API of SPOCK i.e JSphinx may need few improvements but it works perfectly with small speech dictionary.

USING SPOCK ON A LOCAL SYSTEM

INTERFACING WITH OS

Spock has been developed with an objective of being integrated with the system kernel of an Ubuntu distribution. This feature may be a future aspect. As of now, we can run the existing modules of Spock as a third-party application in our operating system.

RUN METHODOLOGIES

We have implemented the Spock System to be not only easy to implement but also, easy to install and run in any local system.

In order to begin the pre-processing you need to initialize the project at first. For the following command is required.

```
sonal@linuxbook-pro# ./initialize
```

The gesture recognition system must then be activated to make the system continuously capture the video feed from an image input source like a file or a live webcam feed. Then to run the program, run the file 'gesture'. This file runs in the background without any window; the only way to know it is running is to check whether the webcam is on.

For that, the command is to run the gesture.sh file generated during the compilation process in the following manner:

```
sonal@linuxbook-pro# ./gesture
```

The file will keep running in the background taking input from the webcam continuously. While making gestures make sure that gestures are quite correct and that the webcam

doesn't move. An arbitrary movement may also be interpreted as one of the gestures and the corresponding action will be performed. All the gestures must be done correctly and slowly. Each gesture must last at least for half a second to be counted as a valid gesture.

Already existing gestures and their functions can be checked by running the command:

```
sonal@linuxbook-pro# ./checkgesture m
Load Successful : scmmd.bin
Gesture : m
Command : firefox
```

New gestures can be added by the command:

```
sonal@linuxbook-pro# ./add-gesture [gesture-character] [custom command]
```

For example:

```
sonal@linuxbook-pro# ./addgesture z google-chrome
```

The [gesture-character] must be a single character like 'w' or 'n', i.e. something like 'star' will not do for the [gesture-character]. Also the system command must be a valid system command. Then following the terminal instructions perform the gesture three times. Each time perform the gesture in a similar way and at the same speed.

If a gesture already exists for the character, it will be overwritten. Some characters have gestures fixed by default like 'u', 'd', 'l', 'r', 's', 'm', 'o' etc., and cannot be overwritten or deleted.

Spock's speech synthesis component can be operated by running the speech.sh file

USING SPOCK ON A LOCAL SYSTEM

USER CUSTOMISATIONS

We offer the user complete customization freedom when it comes to using spock, whether it is the use of different and local natural language speech models or adding custom gestures. This can be done with the help of a training module for vision and speech, wherein the target audience is allowed to configure Spock according to their needs.

Developers and Third Party Application makers are also free to utilize this software according to their own needs and is made possible because of its open source nature.

IMPORTANT LINKS AND RESOURCES

Important Links and Resources

1.1 OpenCV Official documentation: http://www.opencv.org
1.2 CMU Sphinx: http://cmusphinx.sourceforge.net/
1.3 Spock Code Base: http://www.github.com/sonal-raj/spock/
1.4 JavaScript and NodeJS: http://www.nodejs.org
1.5 JavaScript Resources: http://developer.mozilla.org/

FAQ

1.0 What if there are compilation errors in the process of installation.

ANSWER: **** If install runs correctly you dont need to do this ***
Altenetively, you can compile all the files individually using the command:

```
sonal@linuxbook-pro# g++ `pkg-config opencv --cflags` [filename].cpp -o [filename]
`pkg-config opencv --libs` -lX11 -lXtst
```

The files to be compiled are : initialize.cpp, main.cpp, gesture.cpp, addgesture.cpp, checkgesture.cpp and delgesture.cpp.

1.1 Can I contribute to the Spock Project?

ANSWER: Of Course, Yes, We will be happy if you can help.

1.2 How can I contribute to the Spock Project?

ANSWER: Spock is an open source project and its code is hosted on Github. (You can find its link at the useful links section of this manual. You can fork and commit your changes. If you are not a developer, then, your moral support will be appreciated. Trust me, we need it ☺

CONTACT INFORMATION

Contact Information

To know more about this Project, contact the team.

MR. SANJAY KUMAR PROJECT GUIDE	SONAL RAJ (CS110487)	SHASHI KANT CS110462
Sanjay.cse@nitjsr.ac.in	Sonal.nitjsr@gmail.com	supershashikant@gmail.com

Institute Information

National Institute of Technology, Jamshedpur
P.O.RIT, Jamshedpur
http://www.nitjsr.ac.in/