CS-308-2015 Final Report

Motion Sensing TV

Aditya Raj 110050025
Alok Yadav 110050043
Nishit Bhandari 110050026
Tanmay Randhavane 110050010

Table of Contents

1. Introduction

We build a system to enhance user's TV watching experience by incorporating user's movements and his availability adjudged by his activity on phone. The motivation for this comes from automated audio/video control of television like devices. Our system aims to control audio and video of the device automatically using the input from video camera and smartphone. This system can be employed in TVs, Computers, Audio Systems, etc.

2. Problem Statement

The problem centers around designing a smart automated audio/video system that adapts to viewer activity. Viewer activity and corresponding actions include -

- Smartphone conversations

  When user gets a phone call on his smartphone, the device should be muted and the volume is turned up again when the phone call ends.

- Movement in and out of the Field of View (FOV)

  The device adjusts its display to the motion of the person in the field of view and rotates itself so that the person in focus is always facing the display.

- Distance from the device

  Volume of the device is adjusted according to the distance of the user from the device. The farther the user the louder the volume and vice versa.

3. Requirements

3.1 Functional Requirements

The system should be able to

1. detect call on smartphone to adjust volume
2. movement of the user (away/towards the monitor) to adjust volume
3. rotate firebird according to user position for better user experience

3.2 Non-Functional Requirements

The system must be able to respond to changes in topology fast enough otherwise functionality of the system would be of no use.

1. Call functionality must be instantaneous as otherwise audio muted after call ends.
2. User movement must be adapted to in a few seconds otherwise the user would change his position again.

3.3 Hardware Requirements

- Display device (laptop, TV, etc.)
- Android Client (Smartphone)
- Server (connected to display device)
- Camera
- Rotating System (Firebird)
- Xbee

3.4 Software Requirements

- OpenCV: for video input and processing
- ALSA: for adjusting volume of the device
- Android Studio: Phone call input
- AVR Studio : IDE for firebird
- AVR Bootloader: for loading the program into firebird
- Serial Terminal: for Xbee

4. System Design

Our system consists of two independent components.

1. Smartphone Control

   The design of this component is described in the following diagram.

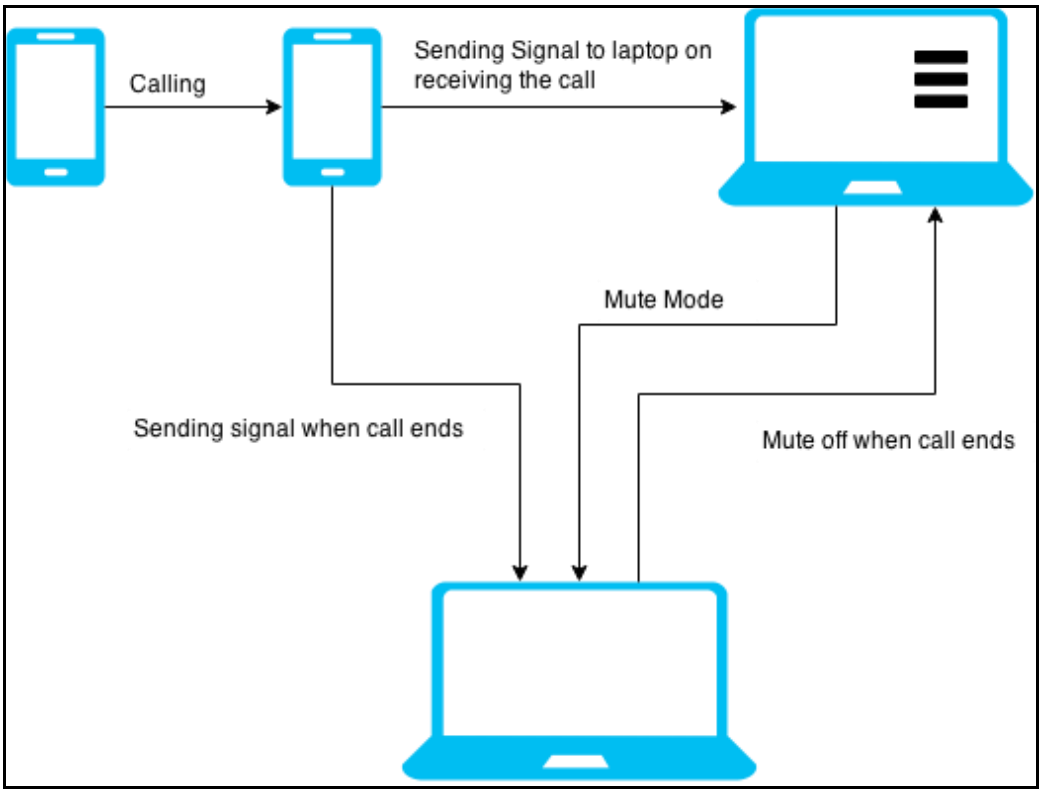

Figure 1 Smartphone control schematic

2. Motion Control

   This component performs two functions

   a. Volume Adjustment: According to the distance of the user from the device decrease or increase its volume.

   b. Change Display Angle: According to the location of the user, rotate the display by a certain amount.

   The schematic for this component is shown in the following diagram.

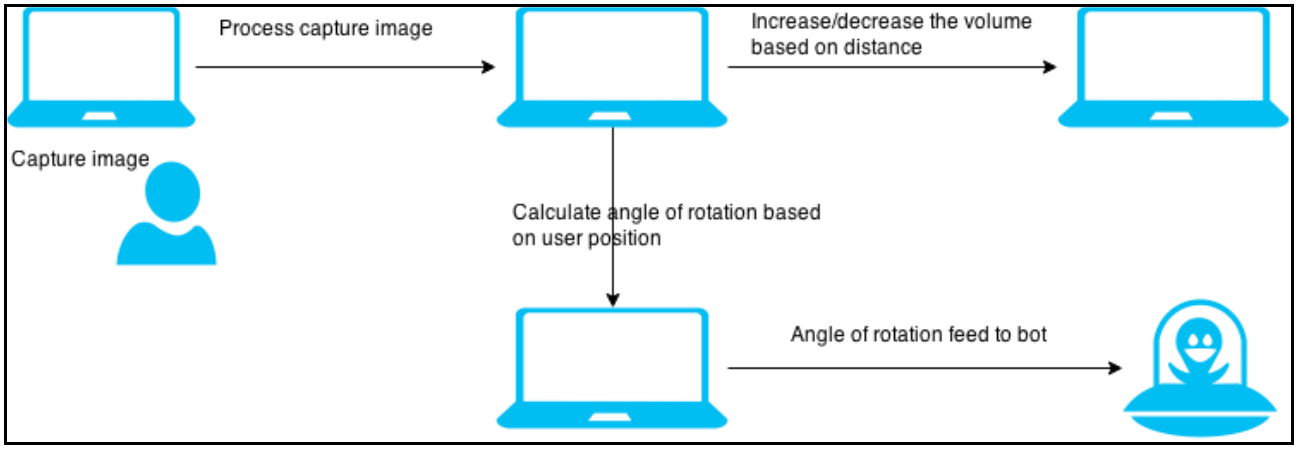

Figure 2 Finding distance and rotation angle from video

5. Working of the System and Test results

- Movement in and out of the Field of View

  To detect the motion and location of the user following algorithm is used. Output of the algorithm was changed by manual inspection and with multiple users. The system performs well with a delay of a few seconds.

```
Function getRotationAngle(video_capture)
        while(1){
                image = video_capture.next_image;
                found_rectangles = HaarFaceDetector(image);       //Face Detection
                found_rectangles += HoGDetector(image);           //Humanoid Detection
                smallest_rectangle = findMinAreaRectangle(found_rectangles);
                draw smallest_rectangle;
                distance = area(smallest_rectangle)/total_area
                angle = arctan(smallest_rectangle.x/distance)
                return angle;
        }
```

- Smartphone Control

  During a call, the audio must be muted and not otherwise. We tested this by placing a call on the smartphone and the audio was muted and as soon as the call was over, the audio went to full. We did this a multiple times. The system performed without any delay.

- Firebird rotation
  After a lot of troubles, we were able to rotate firebird by the amount specified. All the quantized angles were tested and verified with the user position.

6. Discussion of System

a) What all components of your project worked as per plan?

- Smartphone Control
  We were able to control the audio of the device when user got a call on smartphone. An android client runs on the smartphone which sends a signal to the device which in turn adjusts the voice accordingly.
- Movement in and out of the Field of View

  Our system is able to detect user's motion and location from video input and is able to generate the target angle by which it should be rotated.
- Distance from the device

  Our system is able to correctly increase or decrease the volume according to the distance gauzed from the video input.

b) What we added more than discussed in SRS?

- Multiple users
  Along with adjusting the volume and screen rotation to the movement of a single user our system can also work with multiple users. If multiple users are in the field of view, our system assigns a primary user (usually the farthest user) and adjusts the volume according to his position. The screen is rotated if primary user goes out of the field of view and new primary user is chosen.

c) Changes made in plan from SRS:

- Location of the user
  The location of the user was judged using humanoid detection. This proved to be less accurate for laptop because the field of view of the webcam is too small to capture the entire humanoid figure. Also, usually user watches the laptop from close enough that only his face is visible to the device. So, we updated the algorithm to also detect faces and adjust the volume according to combined results from both the detectors.
- Rotating disk system
  Initially, we planned to have a rotating disk platform on which we can hoist our device and it can be rotated by a given angle. This involved many mechanical parts and the system was too complex to construct in the given timeframe. So, we decided to use a prototype device mounted on a Firebird.
- Rotation of the Firebird
  The plan was to send angle of rotation via Xbee which accepts one byte at a time. So we planned to send 3 bytes for angle and direction. But, it turns out that firebird wasn't stopping after the action.
  For example- If we give a command of rotate left by 30 degrees then it moved in a circular

fashion and never stopped. After a lot of unsuccessful experiments, we finally decided to just send a coded byte for rotation. To incorporate all varied angles in one byte, we decided to quantize the angles (+45, +30, +15, 0, -15, -30, -45). If we rotate more than this, then the user goes out of the FOV (Field of view).

7. Future Work

Possible extensions

1) Combining the openCV part and firebird to get a fully functional unit.

2) Rotating firebird with all angles (not just a few quantized ones)

3) Rotating an actual device kept on rotating disk system powered by stepper motors.

4) Implementing this on a smart TV.

5) Age detection of user and change channels accordingly.

6) Reducing frame rates and using better algorithms for faster processing of video.

8. Conclusions

1) This is a promising area of research which has a lot of potential to grow.

2) Implementing the tasks mentioned above can make this a very feasible and useful system.

3) Figuring out the working of firebird was a lot of trouble. With proper documentation, we could have achieved a lot more.

4) We enjoyed a lot working on this project and the support which we got from Sir and the TAs was encouraging and refreshing.

9. References

[1] TCP Sockets Tutorial, http://www.cs.rpi.edu/~moorthy/Courses/os98/Pgms/socket.html

[2] Blocking socket http://stackoverflow.com/questions/26470140/socket-server-that-waits-for-message-from-client-to-read

[3] Reading from sockets in C, http://stackoverflow.com/questions/666601/what-is-the-correct-way-of-reading-from-a-tcp-socket-in-c-c

[4] Robot Manuals, https://www.dropbox.com/sh/ikqp7bi24cf2fm1/AACPrwbNKk5ks00AwHv7LbS3a?dl=0

[5] HoG person detector, https://chrisjmccormick.wordpress.com/2013/05/09/hog-person-detector-tutorial/

[6] Face Detection using Haar Cascades, http://docs.opencv.org/master/d7/d8b/tutorial_py_face_detection.html

[7] Haar-like Features, http://en.wikipedia.org/wiki/Haar-like_features

[8] ALSA volume control, http://stackoverflow.com/questions/6787318/set-alsa-master-volume-from-c-code