João André Martinho Bolas Soares, n40651

Image Processing Report

Motion analysis Interface for Low End Devices

Starting Idea

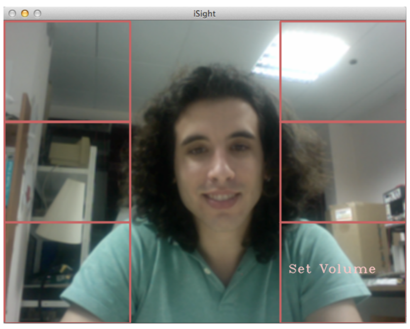
When i came up with this idea , i tough it was interesting having the subject of Image Processing and at same time building something that could be useful for the future. For that reason my idea was basically an interface built on movement by the user, but i wanted to take it a bit further and implement it on a low end device (by low end device i mean a device that doesn't have a lot of hardware resources) and making it usable for common camera's resolution making the analysis in a 2D low (640x480) resolution , this would mean that the code has to be light and effective in order for it to work and don't use much resources for this effectiveness i used OpenCv image Processing Library in C.

This would make a possibility of integrating this system with media centers or other systems, getting common media centers a step further and eliminating the need of remote controls.

So my idea was to build the movement interface and ported to a linux system on a armv6(CPU with performances close to common smartphones).

The several Stages of the Program

I built the interface basing my self on the common notion of motion detection, trough comparing of frames and getting differences between them. Through that image processing method i built several software methods in the program, the first one was a basic Interface to test to functionality of the core of the program, the basic Interface was displayed has in the following image, with the squares being Regions Of Interest and if those were intercepted by a relevant movement it will trigger a command:



<Basic Interface image>

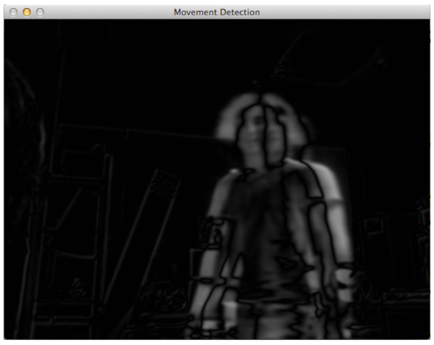
The commands available is the complete volume control like (volume up, volume down and mute) trough Alsa Mixer software and also keyboard keys simulator trough software, this can be used in Linux trough xAutomation making it possible to navigate the display with the interface.

After this i noticed several inconveniences of that interface, starting in the position of the of the several Regions Of Interest being intrusive in the user's movement and forcing the user position to be on the center of the camera frame.

So i begin thinking of a new approach to this, making all the Regions of Interest Dynamic , varying their position has the user moves, to make this possible i needed a tracking system of the user. First i thought of implementing something from my colleges that were working on ASM algorithm but their algorithm needed to be trained and in my case that was not going to be useful , and has a matter of fact probably it would be very resource consuming.

So with the knowledge of motion that i had i started building one from the beginning.

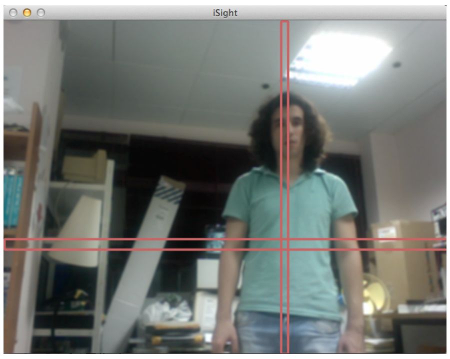
My approach to building this was considered from the study of one motion frame:



<Motion Frame>

So , i though a tracking system should return an X and Y position trough this i could easily make the the regions of interest move dynamically with the user.

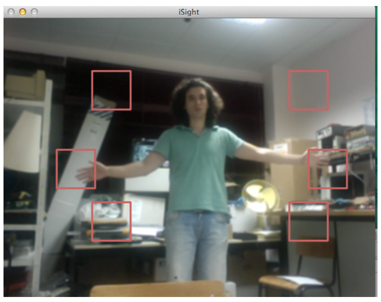
The X position was made slicing the frame vertically in small regions and then analyzing those regions for "motion pixels" (i call motion pixels to pixels that have a value different from zero on the movement frame). The algorithm looks for slice with more motion pixels and then weights with the slices next to that main slice to achieve a final X position.



The Y position of the tracking system was developed always having in mind the average Y position of the user in the frame with this algorithm the Y position is approximately under the chest area of the user.

<Tracking System>

After this implementation i just had to change the size of the Regions of Interest and make them move dynamically with the X and Y positions. Making the final interface completely dynamic shown below :

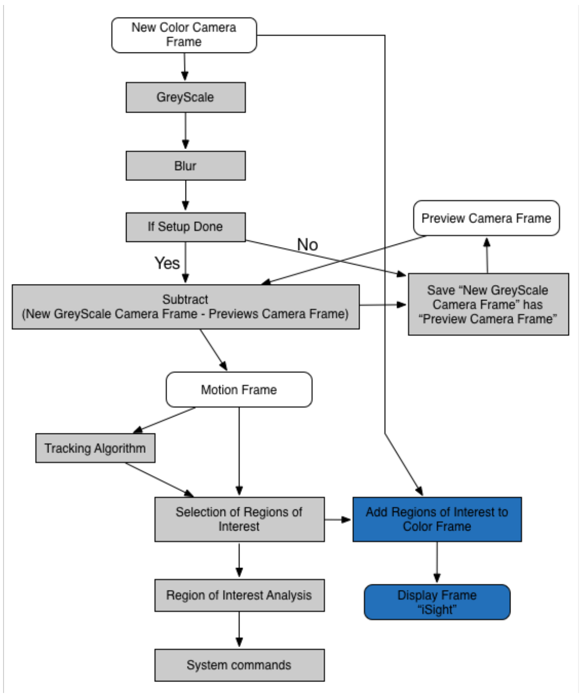


The development of this project was made on Mac OSX system so the control of the system was made on AppleScrypt to test the interface. AppleScrypt is high level language from Apple to control some core functions on the OSX. Has i mentioned before, in order for this to work properly on linux based systems, it needs some third-party software like Alsa Mixer and xAutomation.

The design of the program was to support printable bash commands, although i only had the chance to test the program for OSX i look up and prepared the AlsaMixer/xAutomation commands.

	AppleScript	AlsaMixer/xAutomation
changeVolume: <value> goes from 0 to 100	osascript -e ' set volume output volume <value> '	$ amixer sset 'Master' <value>%
muteSystem: <value> changes from "yes" or "no"	osascript -e ' set volume output muted <value> '	$ amixer set Master toggle
systemKeyPressed: <value> changes from key values on the system	osascript -e ' key down (key code <value>) '	$ xte key <value>

For an easier understanding of the algorithm i made a data flow chart as follows:


New Color Camera Frame
GreyScale
Blur
If Setup Done
No
Yes
Preview Camera Frame
Subtract (New GreyScale Camera Frame - Previews Camera Frame)
Save "New GreyScale Camera Frame" has "Preview Camera Frame"
Motion Frame
Tracking Algorithm
Selection of Regions of Interest
Add Regions of Interest to Color Frame
Region of Interest Analysis
Display Frame "iSight"
System commands

<data flow chart>

In the Data flow chart the white spaces are frames allocated in memory, grey spaces are processes from the program or flux control functions and the blue spaces were methods and frames implemented just for the development phase of the program that could be removed to achieve better performance and making the program run more like an seamlessly interface.

Interface Commands

The interface was built with default commands but it can be easily changed on the code , because of the versatility of the interface, it was implemented a multiplex algorithm on the menus, allowing the Regions Of Interest have different commands on the trigger of only one Region Of Interest.

Main Region Of Interest trigger functions:

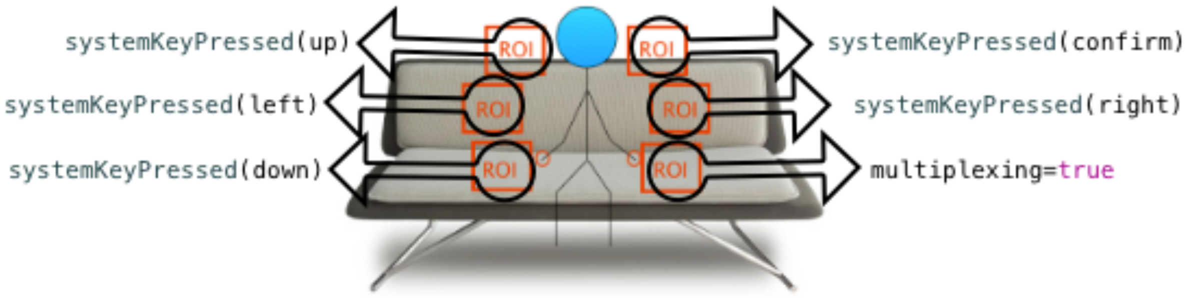
When the user uses the multiplexing ROI the interface commands changes to volume ROI only.

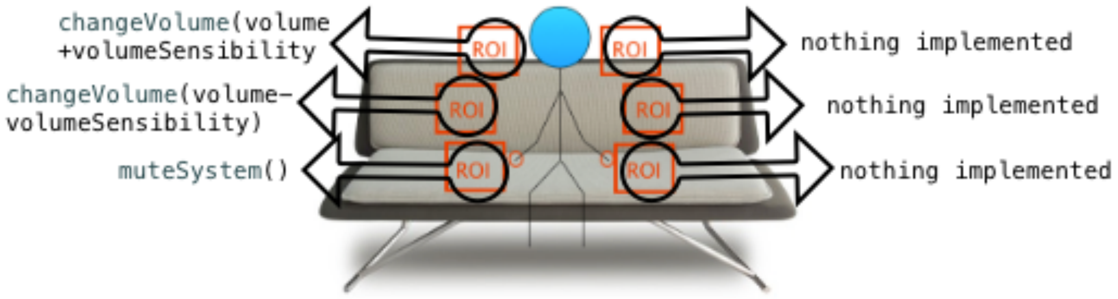
What could be better

There are several things that are not polished on the software, starting with the tracking system that is not perfect, specially on very specific light environments because the motion analysis reads shades has motion and creates a imaginary second user on the frame and it also also has a problem when the tracking system goes to the margin of the frame, the x axis stucks on a collumn near the the margin. Also the use of the key commands couldn't be tested on the mac because of my system but in Linux and trough xautomation i believe it will work.   To achieve good results it is requested to the user to be at least 3 meters away from the camera, it's recommended between 3 meters and 5 meters.

Conclusion

From this project i can surely tell that i have learned a lot from the subject, specially in the motion analysis. This project allowed me to build something a lot more complex than i was expecting at first. I got very familiarized with OpenCV image processing library, and most of all i got a chance to build something while learning how it worked.


systemKeyPressed(up)
systemKeyPressed(left)
systemKeyPressed(down)
ROI
ROI
ROI
ROI
ROI
ROI
systemKeyPressed(confirm)
systemKeyPressed(right)
multiplexing=true


changeVolume(volume
+volumeSensibility
changeVolume(volume-
volumeSensibility)
muteSystem()
ROI
ROI
ROI
ROI
ROI
ROI
nothing implemented
nothing implemented
nothing implemented