# 2W09_live-gesture-recognition
# FINAL PROJECT REPORT

## Introduction

We wish to a Linux-based application for live motion gesture recognition using webcam input in C++. This project is a combination of live motion detection and gesture identification. This application uses the webcam to detect gesture made by the user and perform basic operations accordingly. The user has to perform a particular gesture. The webcam captures this and identifies the gesture, recognises it (against a set of known gestures) and performs the action corresponding to it. This application can be made to run in the background while the user runs other programs and applications. This is very useful for a hands-free approach. While it may not be of great use for browsing the web or writing a text document, it is useful in media player and while reading documents or files. A simple gesture could pause or play the movie or increase the volume even while sitting afar from the computer screen. One could easily scroll through an eBook or a presentation even while having lunch.

The project essentially consists of four parts:

1. Taking input from the webcam and converting it into a form that can be processed easily.
2. Intercepting the gesture from the input of the webcam.
3. Recognising the gesture from a database of gestures.
4. According to the intercepted gesture, give corresponding commands for the operations.

We have used the OpenCV library for handling and manipulating input from the webcam. Difference between subsequent frames helps detect motion. We included further modifications to eliminate noise so that only the moving object (hand or finger) may be interpreted. The interpreted gesture is scanned against a set of known gesture to find which gesture matches the best. The action, either a system command or a keystroke, corresponding to the keystroke is then performed accordingly. In the project we used the OpenCV library for handling and manipulating videos and images. Few basic gestures are incorporated for basic operations, while the user is given the choice of adding other gestures. For actions corresponding to various gestures, we have made the normal scrolling keystrokes, basic system commands and keystrokes specific for VLC Media Player (like for Play/Pause. To simulate keystrokes we have used the X11 library.

**Various features of the code of the project are:**

- Can detect any kind of gesture which is provided in the database.
- Eliminates the background so can be operated in a place where there is no much movement in the background.
- Code is optimised to reduce noise due to change in light.
- The movements of the head while performing the gesture are eliminated.
- Automatically adjusts to the lighting conditions in the room.

## Project Team

### Mentor

Piyush Kumar

### Team Members

1. Aamod Kore
2. Kshitij Singh
3. Nisheeth Lahoti
4. S S Kausik

## Idea

The code continuously streams video from the webcam and processes the frames of the video to recognise the gesture. This is done by subtracting the RGB value of the pixels of the previous frame from the RGB values of the pixels of the current frame. Then this image is converted to octachrome (8 colours only - red, blue, green, cyan, magenta, yellow, white, black). This makes most of the pixels neutral or grey. This is followed by the greying of those pixels not surrounded by 20 non-grey pixels, in the function crosshair(IplImage* img1, IplImage* img2). The non-grey pixels that remain represent proper motion and noise is eliminated. A database is provided with the code which contains a set of points for each gesture. As the user performs the gesture, a set of points are generated (using the average of x & y coordinates of non-grey pixels in each frame) which are matched with the gestures in the databases to find the best match. To match the gestures the points are appropriately scaled as per their standard deviation and then corresponding points of the user's gesture and that from the database are compared. The gesture which has the least sum of squares of the differences between the corresponding points is returned as the match for the gesture. According to the gesture recognised, certain set of commands are executed, like executing a keystroke or a particular system command.

## Usage and Implementation

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:

```
$ sudo apt-get install [package-name]
```

After installing all the packages compile the file - install.cpp using the command:

```
$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.

```
$ ./install
```

If the file 'install' runs correctly you don’t need to do this Alternatively, you can compile all the files individually using the command:

```
$ 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.

Before beginning run the file initialize:

```
$ ./initialize
```

This will adjust the frames per second (fps) of the webcam and the computer's memory and initialize the standard gestures.
Then to run the program, run the file 'gesture'. This runs in the background.

```
$ ./gesture
```

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

```
$ ./checkgesture m
 Load Successful : scmmd.bin
 Gesture : m
 Command : firefox &

 ... Press ESC to continue ...
$
```

New gestures can be added by the command :

```
$ ./add-gesture <gesture character> <custom command>

    for example:
$ ./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. 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', 'x', 'a', 'v' etc, and cannot be overwritten.

The program is made to run in such a way that it is very difficult to stop it. If the program is run from the terminal, to stop it either go to the terminal, hold down the <*Ctrl*> key and press the ' **C** ' key repeatedly; or just close the terminal (force quit). If the program is not run from the terminal try to end the process through the system monitor. If that doesn't work, God help you! Your best bid then is to restart the computer.

## Expenses

NIL. It's completely a coding project.

## Future Prospects and Modifications

This project has a vast arena of development, notably the Sixth Sense project of Pranav Mistry which completely revolutionises the digital world. The code can be extended to incorporate mouse movements as well as still gestures. Further tweaks can be incorporated in the code to increase the efficiency of the gesture recognition process. The code can be improved for better interpretation and recognition of the gestures and newer gestures maybe incorporated for more functionality. The user interface for adding and checking gestures as well as running the program can be improved greatly, e.g. providing an interactive GUI rather than using terminal commands.

## Functions and Limitations

The final project consists of the files *initialize, main*, *gesture, checkgesture, addgesture and delgesture* whose functions are self explanatory. The file 'gesture' is the main file that runs the live gesture recognition programme. The user can add more gestures to the existing default gestures. The code still has some bugs and occasionally gives Segmentation Fault or Memory corruption errors. We hope to eliminate these errors in the future. The code has other limitations, such as similar gestures like ' c ' and ' o ' are not easily distinguishable and the program may give incorrect result in certain cases.

## Important Links

- Stab Wiki : http://http/stab-iitb.org/wiki/2W9_live-gesture-recognition
- Github Wiki : https://github.com/wncc/live-gesture-recognition/wiki
- Github Repo (project code) : https://github.com/wncc/live-gesture-recognition