OOAD PROJECT

MOTION CONTROL

Simulating mouse through hand gestures

VIKAS YADAV - 13114072
SACHIN AGGARWAL - 13114048
SAURABH VERMA - 13114057
DUDDUPUDI SAI AVINASH - 13114016

B.Tech CSE 2nd Year

OOAD PROJECT - MOTION CONTROL

INTRODUCTION

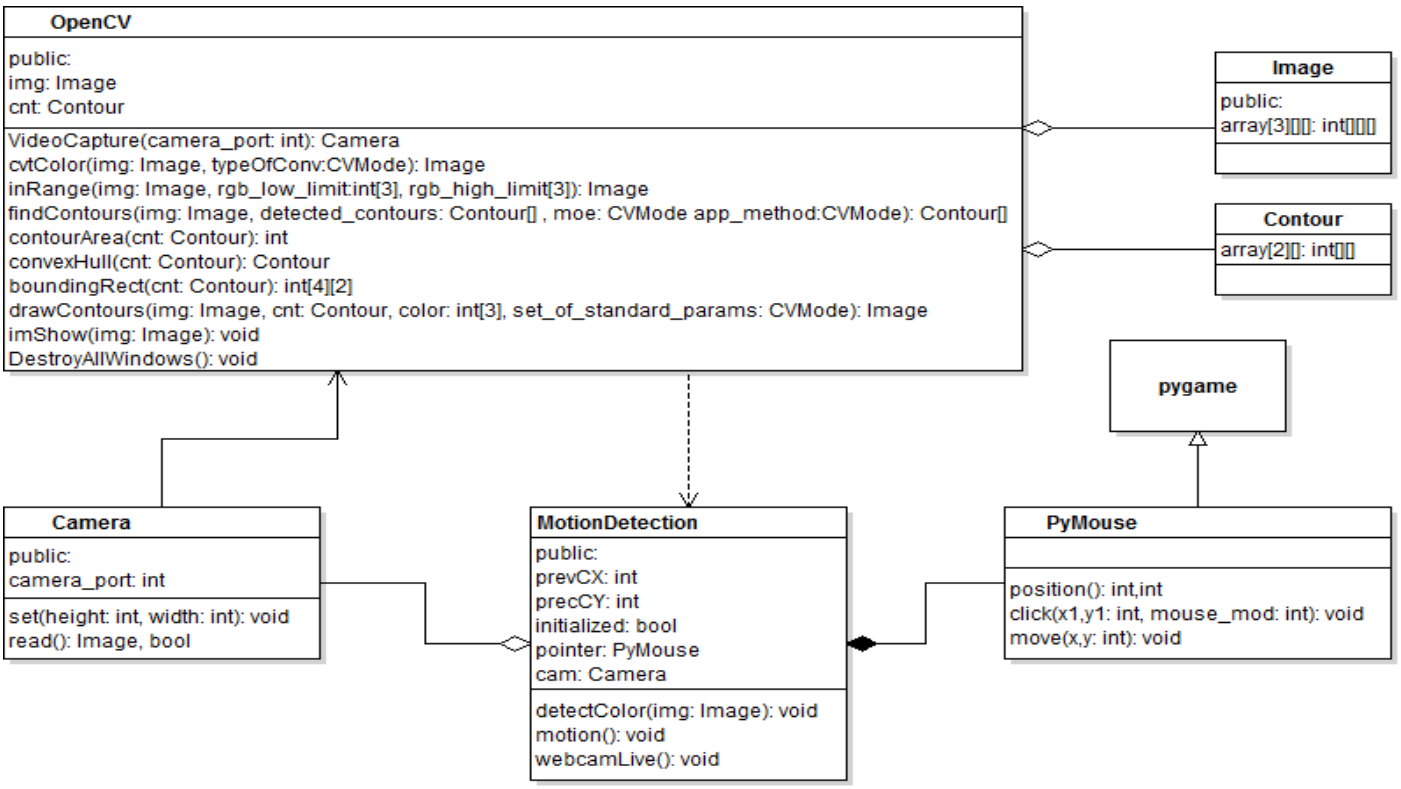
The main purpose of the project is to detect motion of the hand and use it as mouse. The idea of the project can be implemented to many fields in gesture control and motion detection. The project is a smaller step towards building smarter computers and is inspired from the sixth sense by Pranav Mistry.

The python program uses OPENCV and PYMOUSE library along with smaller libraries like NUMPY and SYSTEM to implement the above said functions. The program captures images at a regular interval of time and processes the captured images to find the location of the hand and updates the mouse position accordingly.

This done by first converting the image from RGB to HSV and then finding the blue colour (by defining a lower limit and a higher limit) in the image. The program proceeds by finding the contours(continuous regions having the specified property) having blue colour. The program selects the contour having the largest area (assuming it to be the hand since we assume the hand will be taking most of the space in front of the screen). The program finds the centroid of the contour assumes it to be the centre of the hand. The program also keeps a track of the previous mouse coordinates. All the above function make use of OPENCV library functions. Now the program checks whether a click has been made or not. We define a click as a sudden movement of hand and if the difference between the current coordinates of the mouse and the previous coordinates of the mouse is larger than an assumed value (checked and found out after assumption and testing), then there is a left click. If no click has occurred, then we set the mouse position of the found out mouse coordinates else we perform a mouse click at the current mouse coordinates. The click and mouse movement are implemented using PYMOUSE library. The same process continues till we exit the program.

The program doesn't include a GUI since we assume this to run in the background and thus the current version of the program runs in ubuntu terminal.

• CLASS DIAGRAM



The main classes are described as follows:

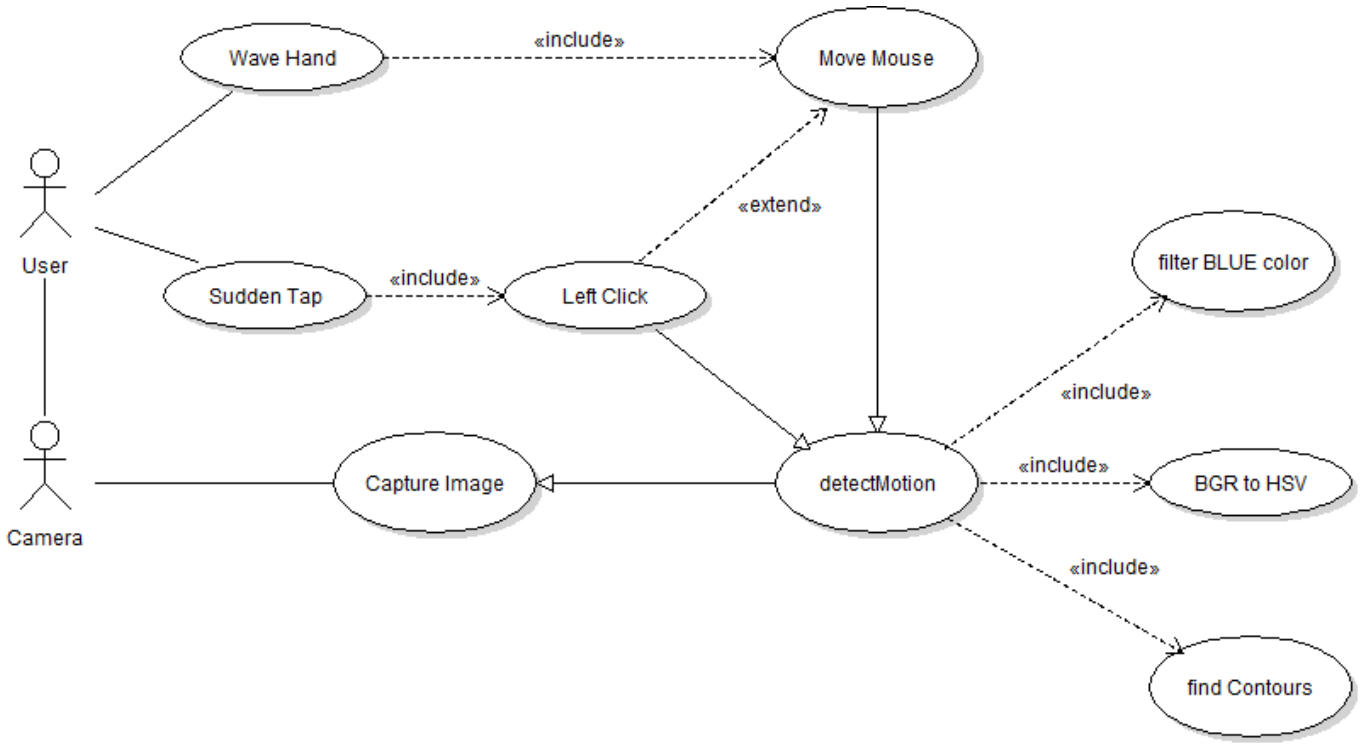
- **MotionDetection** : The main class which has methods for capturing images at regular intervals of time and processing them to find the result and changing the mouse position accordingly.
- **OpenCV** : The library which has most of the functions used for the image processing and filtering out the hand and its position from the image.
- **PyMouse** : The library which has function for changing the mouse position and controlling the mouse.

The relations between the classes are described as follows:

- **MotionDetection** *depends* on **OpenCV** as it is using the functions present in OpenCV for capturing images and processing them.
- **MotionDetection** uses a **Camera** object and thus *aggregation* relation holds between the two.
- **MotionDetection** *depends* on **PyMouse** since it is using its functions for manipulating the mouse positions.

- **OpenCV** also *depends* on **Camera** class since one of the functions uses Camera class.
- **PyMouse** is *inherited* by **pygame**.
- **OpenCV** uses objects of **Image** and **Contour** and thus there exists *aggregation* relation between them.

USE CASE DIAGRAM



There are two main actors in the use case diagram:

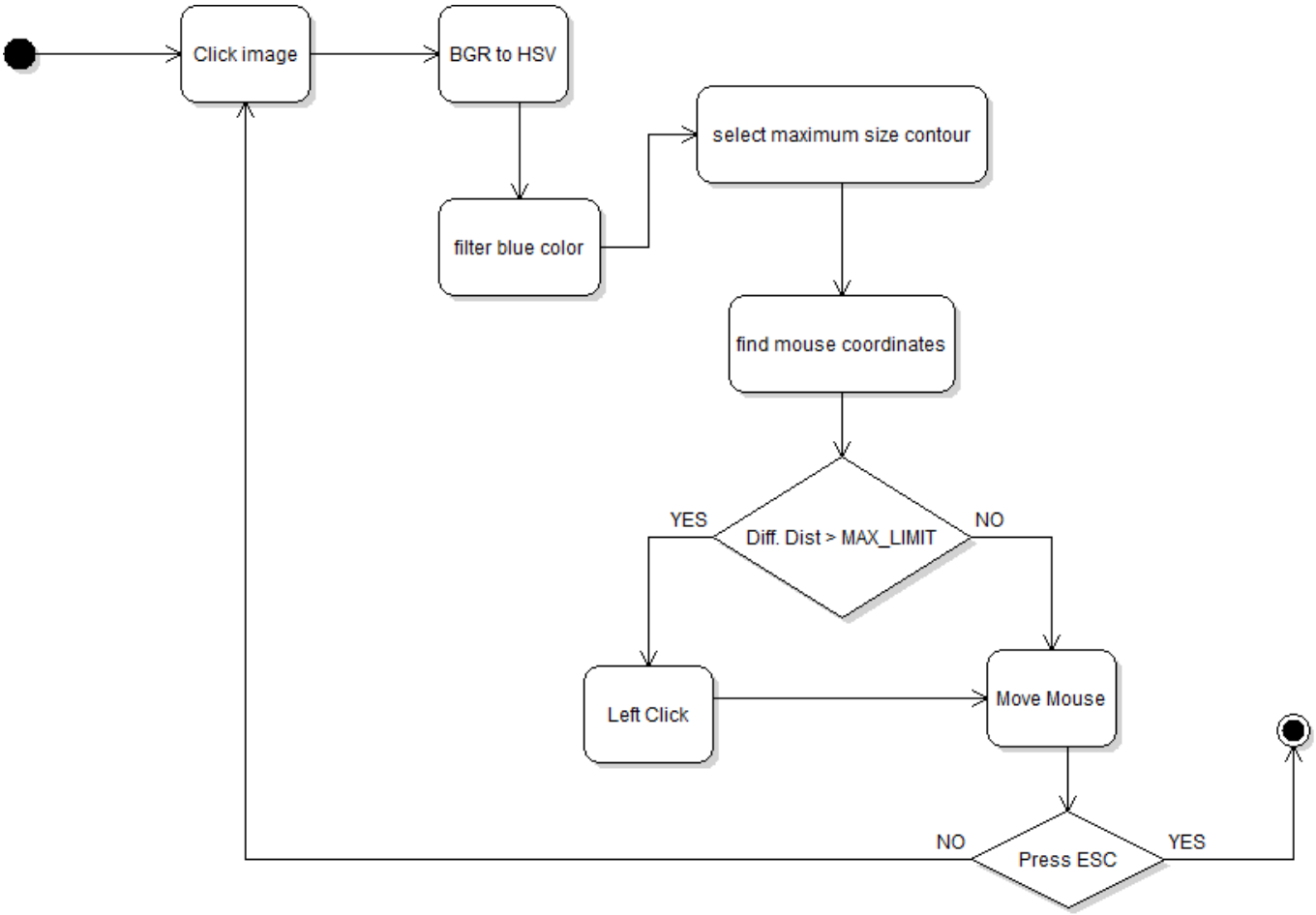
- **USER** : The user can wave the hand in front of the device camera or suddenly tap the hand in the air. The former leads to movement of mouse movement and the latter makes the program perform a left click and then move the mouse to specified coordinates. The mouse is moved using the move and click function from the pymouse library.

- **CAMERA** : The camera has one task, to capture images and the the function detectMotion of the class Motion calculates the new mouse coordinates taking help from the openCV functions performing the below tasks:

- BGR to HSV
- find Contours
- filter blue colour

The two actors mainly control the program and control the workflow.

• ACTIVITY DIAGRAM



The activity diagram describes how the program proceeds and in short explains the whole activities taking place in the program and the decisional conditions in the program.

CONCLUSION

The object oriented methodology really helped us a lot as this was our first experience with pure object oriented programming.

The object oriented approach gave us a lot of benefits like:

- The program became easier to code and debug as we introduced modularity.
- The code is neat, clean to explain and making changes to the program is easier since modules can be changed directly without having to change the code.
- The whole code can be easily seen through by just looking at the UML diagram.

The project can be developed to some really innovative ideas and is a step towards a smarter generation generation of computers. The idea was inspired by "**SIXTH SENSE**" by **Pranav Mistri** and we this has been implemented by **XBOX Kinect** and **Samsung Smart TV**. The further developments to the project include incorporating a right-click and introducing some other smart gestures and the code has been made **OPEN-SOURCE**.