SUMMER PROJECTS '13, IITK

Gesture Recognition using a Webcam.

This software uses a webcam to record and detect gestures with good accuracy on a Linux based system. Basic system commands are run thereafter depending on the gesture.

Team Members:

Arpit Shrivastava

- Roll no. 12161

Pranav Maneriker

- Roll no. 12497

Shubhangee Verma

- Roll no. 12710

Anurag Sharma

- Roll no. 12146

Gesture Recognition
USING A WEBCAM.

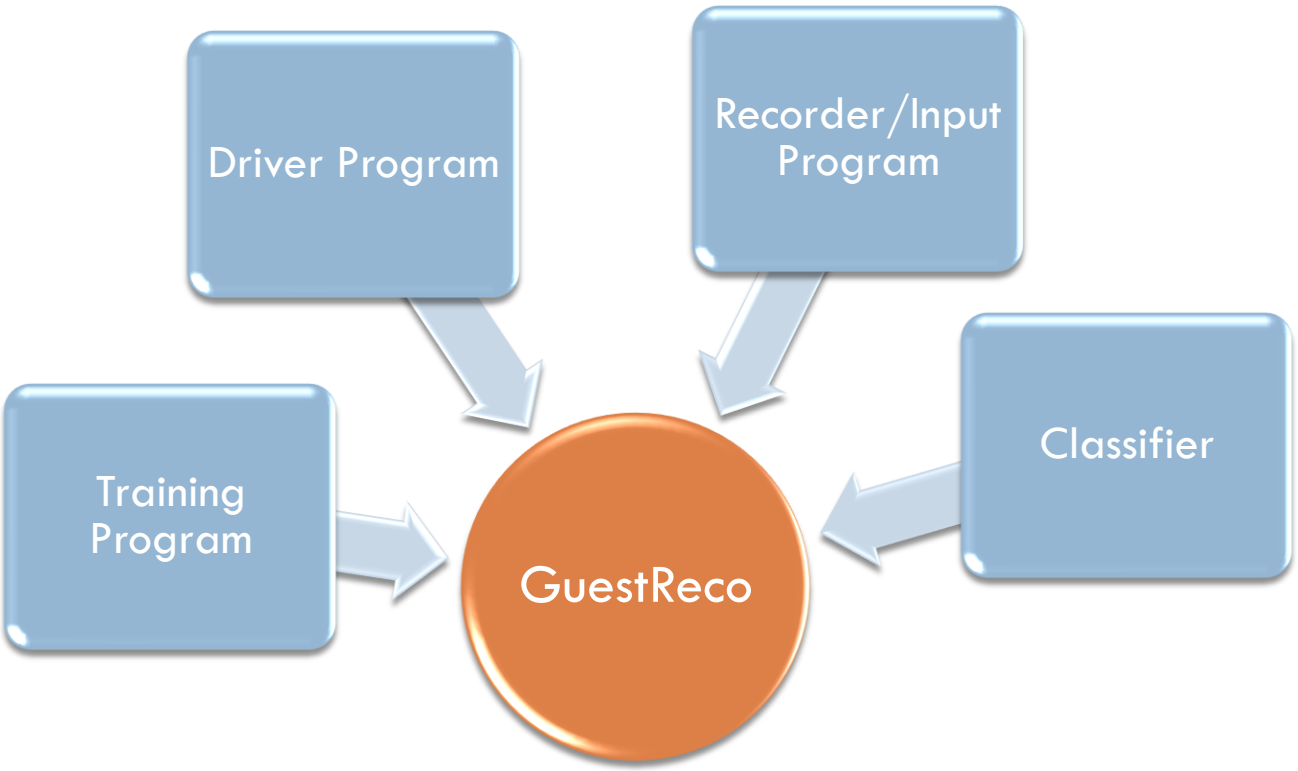
Aim:

The aim of the project is to develop an application that can detect a hand gesture using a webcam and to perform corresponding actions such as system commands or simple keystrokes after properly identifying it against a set of known gestures. The software can identify both one-finger and two-finger gestures. This application can be made to run in the background while the user runs other programs and applications.

Salient features of the application:

1. Recognizes any gesture provided in the database with good accuracy.
2. It recognizes only dynamic gestures i.e involving motion. It does not recognize static gestures.
3. The noise in the background is easily eliminated using color-based filters.
4. New gestures can be added and the existing gestures can be modified to do different tasks which can be specified by the user.
5. The user has to wear thimbles of specific colors while performing a gesture. Of course these colors can be changed by the user by running a specific program.

The structure of the program:



Driver Programs:

1. "Run.cpp"

- It calls and executes "Main.cpp".
- "Main.cpp" returns a value which identifies the gesture and returns its identification number.
- Then the corresponding shell file is executed.

2. "Main.cpp"

- "Gest-reco.cpp" has a function gest-reco() which records a gesture and saves it in a file "recorded" by default.
- int count_files() counts the number of files in the folder – gestures.
- #include<dirent.h> - It's required to count the number of files in a directory.
- In int main() gest-reco() records the gesture and imwrite() saves the file with a rescaled name.

Input Program

1. "Gest-record.cpp" and "Color_2.cpp"

- Includes "cv.h" and "opencv.hpp apart from usual C++ header files.
- Filter defines the HSV filter values.
- The gesture recording time is set for one and a half seconds.
- clear() clears the matrix for the image.
- For filtering we use hsv values rather than rgb values because it is better suited for color based feature extraction. filter() filters the image with hsv values. It shows the image pixels which has hsv value between h.min-h.max, s.min-s.max and v.min-v.max. These pixels are colored white. While choosing the colors for the thimbles care should be taken to choose them as different from the background as possible otherwise background noise hampers the filtering and subsequent operations of the program.
- centroid() plots the centroid of the above obtained filtered pixels.
- rescale() calculates the x.min-x.max and y.min-y.max. It then stretches/shrinks the rectangular array into 32x24 matrix.
- rec_gest() is called for recording the image. It sleeps for 400 milliseconds and then calls rec_gest_image() to start recording the image for next two seconds.

Training Program

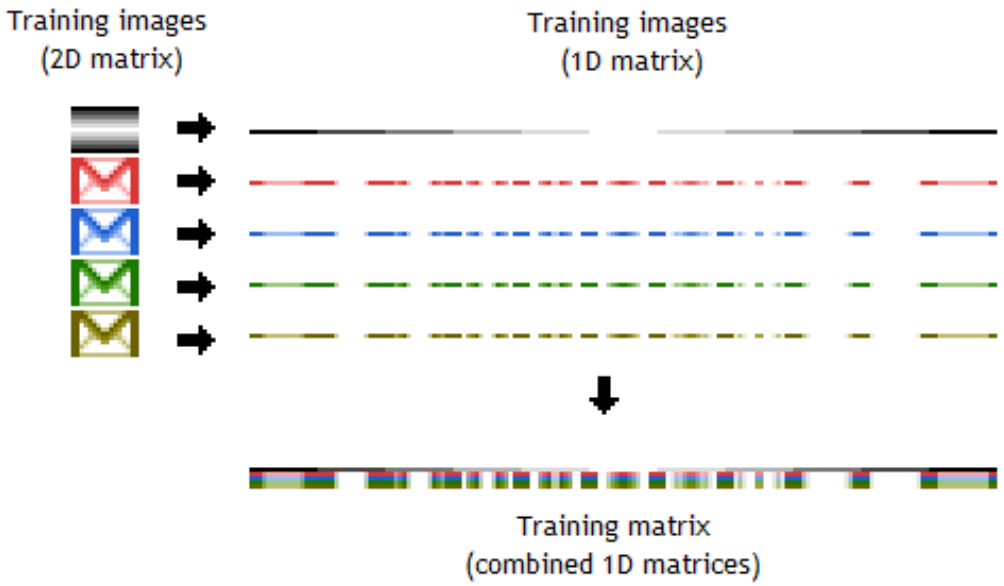
1. "Svm_train.cpp" and "2_col_train.cpp"

- count_gestures() counts the files in ./gestures specified in #define PATH and count_files() does that for /gestures/gesture1 because all gesture'n' folders have same number of test data files.
- To train SVM on a set of images, first one has to construct the training matrix for the SVM. This matrix is specified as follows: each row of the matrix corresponds to one image, and each element in that row corresponds to one feature of the class -- in this case, the color of the pixel at a certain point. Since the images are 2D, one will need to convert them to a 1D matrix. The length of each row will be the area of the images (note that the images must be the same size).
- This has to be done for every image in the training set. After this, one should have the training matrix set up properly to pass into the SVM functions. SVM parameters also need to be set which can be found easily on net.

Classifier

1. "Svm_class.cpp" and "2_col_class"

- Mat_img_1D converts the 2D image into 1D image which is sent to svm.predict for classifying.
- It uses the svm.predict(img_mat1d) which returns a value based on the labels . If the gesture is recognized the function returns 1 otherwise -1. Using this we are able to classify the gestures. Input for this function is the image file which is cut and arranged in 1D array using the above function.



Possible extension of the present project hopefully in the next summers:

- Thimbles can be done away with and the program can be modified to recognize finger movements directly.
- Real-time movement of mouse pointer can be achieved to serve the purpose of the track pad.
- A Flutter-like application can be developed easily