e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:04/April-2022 Impact Factor- 6.752 www.irjmets.com

HAND GESTURE DETECTION USING C++ AND OPENCV

Mehul Singh Negi[*1]

[*1]Dept. Of Computer Science & Engineering, Dronacharya College Of Engineering, Farrukh Nagar, Gurgaon – 123506, India.

ABSTRACT

As nowadays artificial intelligence systems are being used extensively, the ordinary sensor data might not be sufficient for these systems. Thus collecting images from the cameras and building a decision mechanism by Processing them is what organizations are majorly aimed at while designing many such systems. Since interplay of robots and humans are increasing every day, anticipation of human movements by robots/systems is to be expected in the future. Because of this reason, the developed project evaluates the number if fingerprints in real time by trying to Comprehend the movement of the human hand. In doing so, the background need not to be a solid color. This Software, which tries to reduce background noise as much as possible, is created by only signal processing methods and works with high efficiency.

Keywords: Image Processing, Hand Detection, Hand Motion Identification, Number Of Fingers.

I. INTRODUCTION

As the interaction of robots and humans is increasing rapidly with the development of artificial intelligence applications, sensor technology is also developing rapidly. However the sensors are insufficient or costs too much to import non-ordinary data from outside. For this reason, image processing technology is preferred for computers both to reduce costs and to interpret complex data more intelligently. Even if it is a simple webcam, smart systems can be set up with developed algorithms. Detecting hands with image processing and deciphering finger numbers can help people with disabilities interact with technological systems. For example, a severely visually impaired student may show his answer to the camera with his fingers after listening aloud to oral test questions. In addition to facial recognition in security systems, it can be considered to enter passwords with hand movements. In a different way, a system can be developed that interprets sign language with finger movements. Several filters were used to detect the human hand in the project. First, background reduction is performed with the MOG2 algorithm. Later, non-human objects are darkened, using specially designed filters to detect the color of human skin.

After this step, the non-uniform noise on the image signals is reduced as much as possible with the middle filter. Then, HSV color space conversion and gray-scale conversion are performed. As a final step of filtering, the Otsu algorithm was used in conjunction with the binary thresholding method, to mask those below the threshold value with black.

II. IMPORTANT TERMINOLOGY

2.1 Region of Interest:

A region of interest (ROI) is a part of an image that you want to filter or perform some other operation. We define a ROI by generating a binary mask, which is a binary image of the same size as the image we wish to process with pixels having ROI set to 1 and 0 for all the other pixels.

2.2 Contour:

A contour is defined as the line joining all the points along the boundary of the image with the same intensity. Contours are used in shape analysis, locating the shape of the object of interest and object locating.

2.3 Convex Hull:

A convex hull can be defined either as the intersection of all convex sets containing a given subset of a Euclidean space, or equivalently as the set of all convex combinations of points in the subset. For a bounded subset of aircraft, the convex hull can be viewed as a shape enclosed by a rubber band stretched around the subset.

III. BACKGROUND REDUCTION AND SKIN COLOR FILTERING

Each pixel in the image has a different probability density function. The pixels in the new image will have a different probability density function than those in the old image (background). Since the camera is stationary,

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:04/April-2022 Impact Factor- 6.752 www.irjmets.com

we can analyze the part we describe as the background, such as a photograph. After statistical modeling analysis, the variances are determined for the pixel intensity levels. For this, the single Gaussian model [3] specified in equation no. (i) can be used. The position of a pixel relative to time in the RGB color space or any other color space is represented by the notation x(t).

In pixel-based background deletion, we can determine the probability of a pixel belonging to the background (AP) or the probability of belonging to the foreground (AP) with Bayes' rule (R). This process is also used in a single Gaussian model.

$$R = \frac{p(AP \mid X^t) = p(X^t \mid AP)}{p(OP \mid X^t) \quad p(X^t \mid OP)} \qquad - \text{(i)}$$

One method developed to detect the color of human skin is a hybrid method that works as a combination of RGB, HSV, YCbCr color spaces and fixed. The human skin color cannot be distinctly differentiated by the 3 channels (red, green, blue). However, since there are factors such as daylight, we will use the RGB color space to reduce the impact of these factors [1]. HSV (hue, saturation, value) consists of hue, saturation, and brightness components.

IV. HSV AND GRAY SCALE CONVERSION

Before applying the threshold value to the Hand object and masking it (the step just before starting the geometric operations), we need to apply the Hsv transform and the Gray transform to the image.

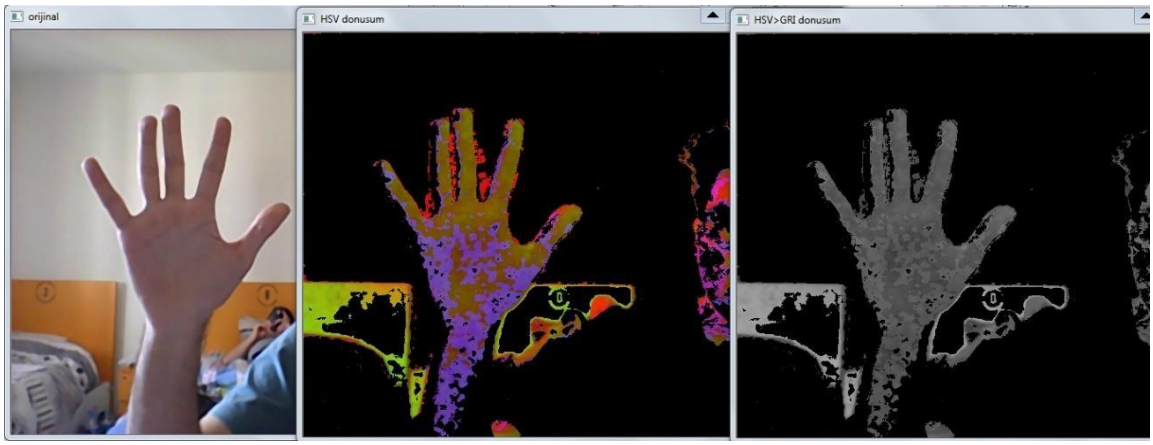

Fig 1: HSV colour space conversion and gray-scale conversion

V. MASKING

The Otsu algorithm takes advantage of the bimodal histogram of the image and assumes that there are two classes of pixels in the image, the background and the foreground. We can think of a bimodal histogram as a normal histogram. However, unlike normal, this is a histogram that reaches a critical peak value at two points.

In this way, the GRASS algorithm can know which value is closer to the mean of the background pixels and in which values are approximately the mean of the foreground pixels. The Otsu method calculates the optimal threshold value that separates these two pixel classes with the equation shown in equation (ii). Thus, within-class variance is minimum and between-class variance is maximum [2].

$$w_1(t)\,q_1(t)^2(t) + w_2(t)\,q_2(t)^2(t) \qquad - \text{(ii)}$$

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:04/April-2022 Impact Factor- 6.752 www.irjmets.com

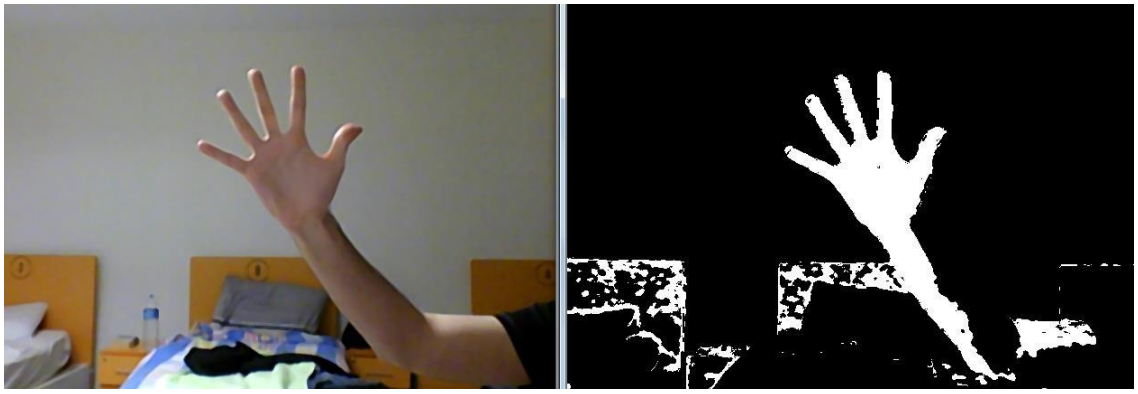

Fig 2: Masking with Otsu Algorithm and binary thresholding

VI. FINDING LARGEST CONTOUR AND APPLYING CONVEX BODY ALGORITHM

To perform geometric operations on our hand image, we need to set the boundaries of our hand on the filtered black and white image. For this we need to be able to quickly understand the boundaries (contour) of our hand.

In this way, we will keep tracking our hand with the software while setting the range of our hand. So, we do not need to touch any particular part of the image, wherever our hand is, the camera will detect it.

In OpenCV it is possible to find the edges of objects with the find Contours command, but since this command finds all edges of the image, if an object appears in the background that the filters cannot prevent, the edges of this object will also be detected. To prevent this, we will consider the object with the largest area among the objects whose edges are detected.

The convex hull algorithm is an algorithm that finds the line segments whose contours connect the outermost points of a known set of points. At the same time, we will use the points where these line segments meet for our operations. These points are called extreme points. By counting the extreme points, we will count the number of fingers present in the image.

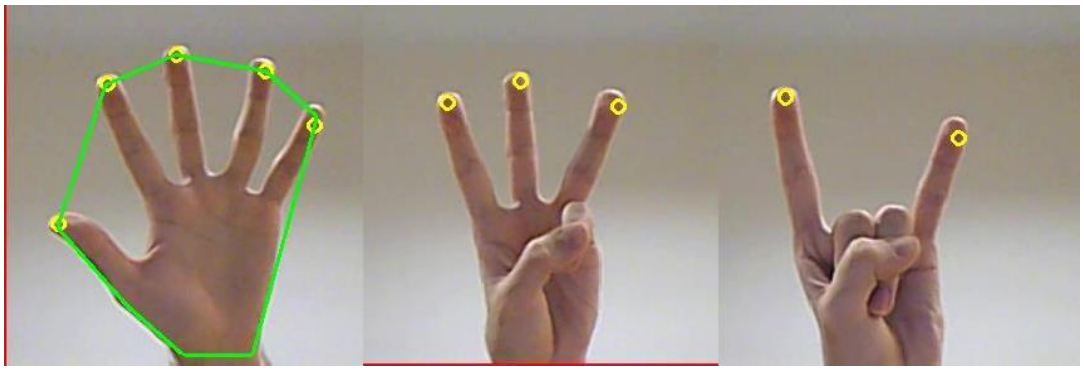

Fig 3: Green lines are convex hull lines and yellow dots are extreme points

When assigning a threshold value, the maximum value is 255, a value of 250 if there are too many objects in the background and a value of 215 when there is a moderate level of variation in the background is the optimal value required for the software to function.

Algorithm:

Step - 1: Capture the video.

Step - 2: Define the region of interest and convert BGR colors into HSV.

Step - 3: Define skin color range in HSV and store the background in a mask.

Step - 4: Dilate and blur the image to decrease noise.

Step - 5: Find contours and approximate them.

Step – 6: Define convexHull to show the coloured border around the hand.

Step - 7: Find defects and remove unrequired defects.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:04/April-2022 Impact Factor- 6.752 www.irjmets.com

Step - 8: Display the image inside the convexHull.

Step – 9: Finish when the esc key is pressed.

Program using C++ & OpenCV:

Apart from the standard library of C++ and OpenCV, we have used (High Gui, Core, Features2d and background_segm libraries along with the imageproc library to implement a basic hand gesture detection program. The following program accepts number of fingers as input through a webcam and processes it to display the count of fingers shown (on webcam as input)

```
include <stdio.h>
#include <iostream>
#include <opencv2\core\core.hpp>
#include <opencv2\highgui\highgui.hpp>
#include <opencv2\core\core.hpp>
#include <opencv2\imgproc\imgproc.hpp>
#include <opencv2\features2d\features2d.hpp>
#include <opencv2\opencv.hpp>
#include <opencv2/video/background_segm.hpp>
using namespace cv;
using namespace std;
void track(int, void*);
Mat originalImage;
Mat figMaskMOG2;
Mat grayImage, eyelash, or2, edges, mirrored;
int thresh = 140, maxVal = 255;
int type = 1, value = 8;
int main(){
Ptr< BackgroundSubtractor> pMOG2 = new BackgroundSubtractorMOG2();
cv::Rect myRoi(288, 12, 288, 288);
VideoCapture cap;
cap.open(0);
while (1)
{
cap >> originalImage;
cv::flip(originalImage, mirrored, 1);
cv::Point(12, 12), cv::Scalar(0, 0, 255));
cv::rectangle(mirrored, myRoi, cv::Scalar(0, 0, 255));
eyelash = mirrored(myRoi);
cvtColor(eyelash, grayImage, CV_RGB2GRAY);
GaussianBlur(grayImage, grayImage, Size(23, 23), 0);
namedWindow("set", CV_WINDOW_AUTOSIZE);
createTrackbar("Threshold", "set", &thresh, 250, track);
createTrackbar("Maximum", "set", &maxVal, 255, track);
createTrackbar("Threshold Type", "set", &type, 4, track);
createTrackbar("Edges", "set", &value, 100, track);
pMOG2->operator()(eyelash, figMaskMOG2);
```

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:04/April-2022 Impact Factor- 6.752 www.irjmets.com

```
cv::rectangle(figMaskMOG2, myRoi, cv::Scalar(0, 0, 255));
track(0, 0);
imshow("Original Image", mirrored);
imshow("Background Removed", figMaskMOG2);
imshow("Grey",grayImage);
char key = waitKey(24);
if (key == 27) break;
}
return 0;
}
void track(int, void*){
int count = 0;
char a[40];
vector<vector<Point> > contours;
vector<Vec4i> hierarchy;
GaussianBlur(figMaskMOG2, figMaskMOG2, Size(27, 27), 3.5, 3.5);
threshold(figMaskMOG2, figMaskMOG2, thresh, maxVal, type);
Canny(figMaskMOG2, edges, value, value * 2, 3);
findContours(figMaskMOG2, contours, hierarchy, RETR_TREE, CHAIN_APPROX_SIMPLE, Point(0, 0));
Mat drawing = Mat::zeros(edges.size(), CV_8UC3);
if (contours.size() > 0){
size_t indexOfBiggestContour = -1;
size_t sizeOfBiggestContour = 0;
for (size_t i = 0; i < contours.size(); i++){
if (contours[i].size() > sizeOfBiggestContour){
sizeOfBiggestContour = contours[i].size();
indexOfBiggestContour = i;
}
}
vector<vector<int> >hull(contours.size());
vector<vector<Point> >hullPoint(contours.size());
vector<vector<Vec4i> > defects(contours.size());
vector<vector<Point> >defectPoint(contours.size());
vector<vector<Point> >contours_poly(contours.size());
Point2f rect_point[4];
vector<RotatedRect>minRect(contours.size());
vector<Rect> boundRect(contours.size());
for (size_t i = 0; i<contours.size(); i++){
if (contourArea(contours[i])>5000){
convexHull(contours[i], hull[i], true);
convexityDefects(contours[i], hull[i], defects[i]);
if (indexOfBiggestContour == i){
minRect[i] = minAreaRect(contours[i]);
for (size_t k = 0; k<hull[i].size(); k++){
```

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:04/April-2022 Impact Factor- 6.752 www.irjmets.com

```
int ind = hull[i][k];
hullPoint[i].push_back(contours[i][ind]);
}
for (count = 0,size_t k = 0; k<defects[i].size(); k++, count++){
if (defects[i][k][3]>13 * 256){
int p_start = defects[i][k][0];
int p_end = defects[i][k][1];
int p_far = defects[i][k][2];
defectPoint[i].push_back(contours[i][p_far]);
circle(grayImage, contours[i][p_end], 3, Scalar(0, 255, 0), 2);
}
}
if (count == 1) strcpy_s(a, "1");
else if (count == 2) strcpy_s(a, "2");
else if (count == 3) strcpy_s(a, "3");
else if (count == 4) strcpy_s(a, "4");
else if (count == 5 || count == 6) strcpy_s(a, "5");
else strcpy_s(a, "SHOW HAND");
putText(mirrored, a, Point(75, 450), CV_FONT_HERSHEY_SIMPLEX, 3, Scalar(0, 255, 0), 3, 8, false);
drawContours(drawing, contours, i, Scalar(255, 255, 0), 2, 8, vector<Vec4i>(), 0, Point());
drawContours(drawing, hullPoint, i, Scalar(255, 255, 0), 1, 8, vector<Vec4i>(), 0, Point());
drawContours(grayImage, hullPoint, i, Scalar(0, 0, 255), 2, 8, vector<Vec4i>(), 0, Point());
approxPolyDP(contours[i], contours_poly[i], 3, false);
boundRect[i] = boundingRect(contours_poly[i]);
rectangle(grayImage, boundRect[i].tl(), boundRect[i].br(), Scalar(255, 0, 0), 2, 8, 0);
minRect[i].points(rect_point);
for (size_t k = 0; k<4; k++){
line(grayImage, rect_point[k], rect_point[(k + 1) % 4], Scalar(0, 255, 0), 2, 8);
}
}
}
}
imshow("Conclusion", drawing);
}
```

VII. RESULT AND PERFORMANCE

Figure 4 shows the result of the method used in this study which performs hand gesture detection in real-time . As you can see, the bounding box is slightly larger as it was intentionally left to cover the maximum part than scattered and all unnecessary information was removed from the image so that we can get a bigger coverage area.

This also helps in the zoom case when the object is very large as it will clearly identify which gesture is being represented and then it returns with the best match count of the fingers from the stored image after processing the captured image.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:04/April-2022 Impact Factor- 6.752 www.irjmets.com

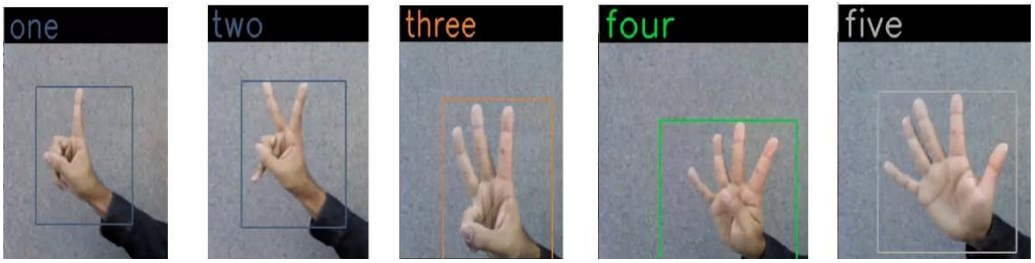

Fig 4: Result of detected gesture

As shown in Figures 5 and 6 below, we can see that the method proposed in this paper has high accuracy. The overall accuracy reached 98.41%, indicating the effectiveness of the method. In addition, we also report the number of frames per second that can be processed. It can achieve real-time gesture recognition with a maximum speed of up to 10fps. Compared to other recent research, this method has high accuracy.

According to [4], in low dimension, the recognition rate of the MAOMP algorithm can still exceed 85% and the recognition rates of SAMP, SP and SWOMP remain at 80–85%, while the recognition rates of ROMP and OMP algorithms is less than 80%. In [5], the paper comes up with a deep learning-based fast hand gesture recognition method using representative frames. The precision of this mechanism is 95.18%.

In [6], the paper uses presence features and a 3D point cloud to recognize gestures. It can apprehend 9 gestures and in general the accuracy can reach up to 94.7%. These results show accuracy and efficiency.

Table 1: Accuracy comparison of different methods

Methods	MAOMP	SAMP, SP, SWOMP	ROMP, OMP	Paprt [14]	Paper [15]	Method in this paper
Accuracy	85%	80% – 85%	80%	95.18%	94.7%	98.41%

VIII. CONCLUSION

Gesture plays an essential role in human-computer-interaction. This paper comes up with a precise and efficient method for gesture recognition. First, the skin is detected using rules based on experience and the image is converted into a binary image. Then expansion and erosion are adopted. After that, all the contours are drawn and the contour of the hand is found. Finally, gesture recognition can be done using the pyramid pooling module and the attention mechanism.

IX. REFERENCES

[1] Abdul Rahman, N., Wei, K. C., & See, J. (2006). RGB-H-CbCr Skin Color Model for Human Face Detection.

[2] Kaehler A. & Bradski G. (2016) Otsu's Algorithm. In A. Kaehler & G. Bradski, Learning Open CV 3 (pp. 258-259). O'Reilly Media, Inc.

[3] Zivkovic, Z. (2004). Improved adaptive gaussian mixture model for background subtraction. In Pattern Recognition, International Conference on Pattern Recognition, 28-31.

[4] Li B, Sun Y, Li G, et al. Gesture recognition based on modified adaptive orthogonal matching pursuit algorithm[J]. Cluster Computing, 2017(3):1-10.

[5] John V, Boyali A, Mita S, et al. Deep Learning-Based Fast Hand Gesture Recognition Using Representative Frames[C]

[6] Chong, Y, Huang, J. and Pan, S. (2016) Hand Gesture Recognition Using Appearance Features Based on 3D Point Cloud. Journal of Software Engineering and Applications, 9, 103-111.