ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 7, Issue 7, July 2020

Application of Computer Vision in Human-Robot Interaction Using Hand Gesture for Robot Control

Samson Z. Damtew1*, Prabhakar.S2
1* Dean, Mechatronics Department, Engineering College, Kombolcha Institute of Technology, Wollo University, Ethiopia.
Email: samsonzerihun@kiot.edu.et
2 Associate professor, Mechanical department, Automotive Engineering Stream, Kombolcha Institute of Technology, Wollo University, Kombolcha-208, Ethiopia. Email: praba.rockson@gmail.com, prabhakar@kiot.edu.et, prabhakar@avit.ac.in

Abstract: This paper aims about implementing a computer vision application that can be used to control a robot through simple hand gestures. The main motivation of our paper is the need to develop a simple robot controller that can interact smoothly with humans without having any special devices. The purpose of this paper is to analyze the different hand gesture and finger representations which are captured from webcam from different users in a real time so that the robot will detect, recognize and interpret the meanings for the respected commands so that take action to control the robot motion or movement directions based on the current situations into consideration. During the project, four gestures were chosen to represent four navigational commands for the robot, namely Move Forward, Turn Left, Turn Right, and Stop. The result shows that this program is robust to color contrast and in detection and recognition of different hand skins from different users while further improvement and optimization of memory and storage allocations needs to be done using different open CV versions and can be also implemented for other related application that needs a human machine interactions.

Index Terms—Computer vision, Convex hull, Gesture recognition, Human and robot, HMI, Hand gestures.

I. INTRODUCTION

Gesture recognition has been a very interesting problem in Computer Vision community for a long time. This is particularly due to the fact that segmentation of foreground object from a cluttered background is a challenging problem in real-time. Nowadays robot are used successfully in many areas, particularly in industrial production, military applications and space explorations. This success drives the interest in the feasibility of using robots in human social environments. In human social environments communications can be done using verbal language (audio) or gesture. Robots can also be controlled using speech but due to the complexity of speech signal the control is a more difficult using verbal language.

Early approaches to the hand gesture recognition problem in a robot control context involved the use of markers on the finger tips [1]. An associated algorithm is used to detect the presence and color of the markers, through which one can identify which fingers are active in the gesture. The inconvenience of placing markers on the user's hand makes this an infeasible approach in practice. Recent methods use more advanced computer vision techniques and do not require markers. Hand gesture recognition is performed through a curvature space method in [2], which involves finding the boundary contours of the hand. This is a robust approach that is scale, translation and rotation invariant on the hand pose, yet it is computationally demanding. In [3], a vision-based hand pose recognition technique using skeleton images is proposed, in which a multi-system camera is used to pick the center of gravity of the hand and points with farthest distances from the center, providing the locations of the finger tips, which are then used to obtain a skeleton image, and finally for gesture recognition. A technique for gesture recognition for sign language interpretation has been proposed in [4]. Our focus is the recognition of a fixed set of manual commands by a robot, in a reasonably structured environment in real time. Therefore the speed, hence simplicity of the algorithm is important. We develop and implement such a computer vision procedures in this work in our detection and recognition of the four gestures and their translation into the corresponding commands for the robot. The appropriate OpenCV functions and image processing algorithms for the detection and interpretation of the gestures were used. Then after, the program was tested on a webcam with actual hand gestures in real-time and the results to be observed.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 7, Issue 7, July 2020

II. MATERIALS AND METHODS

As the first step in hand gesture recognition is obviously to find the hand region by eliminating all the other unwanted portions in the video sequence, the techniques used in our research project involves the use of several algorithms commonly used in computer vision. These include those used in color segmentation, contour drawing, convex hull, feature extraction, and gesture recognition. A computer vision application written in C is used integrated with the computer webcam as a camera for the robot.

So, our first step to find the hand region from a video sequence involves three simple steps.

- Background Subtraction
- Motion Detection and Thresholding
- Contour Extraction

Having segmented the hand region from the live video sequence, we will make our system to count the fingers that are shown via a camera/webcam. Here, we will use ideas from region-based segmentation [5, 6] to alleviate this problem. Our assumption is that as the largest connected white region corresponds to the hand, we will use a relative region size threshold to eliminate the undesired regions. Then after, to count the fingers, we will introduce a faster approach to perform hand gesture recognition as proposed by Malima et.al. Therefore, our methodology to count the fingers (as proposed by Malima et.al) will be as shown in the figure 1 below.

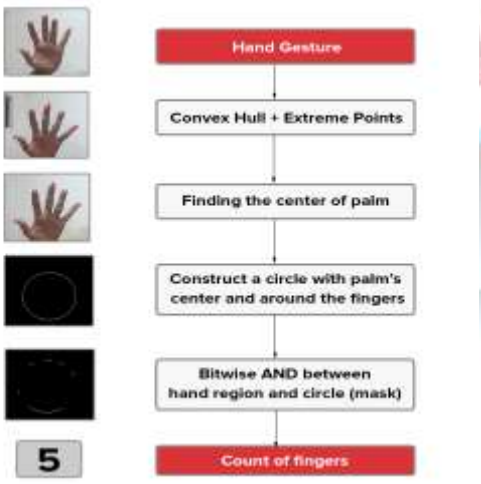

Figure 1: Methodology of Hand-Gesture Recognition algorithm to count the fingers

As you can see from the above image, there are four intermediate steps to count the fingers, given a segmented hand region. All these steps are shown with a corresponding output image (shown in the left) which we get, after performing that particular step.

1. Find the convex hull of the segmented hand region (which is a contour) and compute the most extreme points in the convex hull (Extreme Top, Extreme Bottom, Extreme Left, Extreme Right).
2. Find the center of palm using these extremes points in the convex hull.
3. Using the palm's center, construct a circle with the maximum Euclidean distance (between the palm's center and the extreme points) as radius.
4. Perform bitwise AND operation between the thresholded hand image (frame) and the circular ROI (mask). This reveals the finger slices, which could further be used to calculate the number of fingers shown.

Once we have got the segmented hand region, we can calculate its centroid, or center of gravity (COG), $(\bar{X}, \bar{Y})$[7], as shown in (1):

$$(\bar{X} = \frac{\sum_{i=0}^{k} x_i}{k} \quad and \quad \bar{Y} = \frac{\sum_{i=0}^{k} y_i}{k}) \tag{1}$$

Where x_i and y_i are x and y coordinates of the i^{th} pixel in the hand region, and k denotes the number of pixels in the region.

Furthermore, below is our implementation section that present the entire function used to perform the above four steps.

III. IMPLEMENTATION

Color segmentation using Thresholding

Hand segmentation deals with separating the user's hand from the background in the image. This can be done using various different methods. The most important step for hand segmentation is thresholding which is used in most of the methods described below to separate the hand from the background such that each pixel is either classified as a hand pixel or a background pixel.

Thresholding can be done using HSV or YCrCb color spaces. In this project both color spaces are used in combination for robust hand segmentation.

The algorithm used for the color segmentation using thresholding should consider the following points:

1) Capture an image of the hand gesture from the camera.
2) Determine the range of HSV and/or YCrCb values

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 7, Issue 7, July 2020

for skin color for use as threshold values.

3) Convert the image from RGB color space to HSV and YCrCb color space.
4) Convert all the pixels falling within the threshold values to white.
5) Convert all other pixels to black.
6) Save the segmented image in an image file.

A static threshold value is selected from 0 to 255. This threshold value is chosen in such a way that the white blob of the hand is segmented with minimum noise possible. The system use the following ranges in HSV and YCrCb color spaces.

H[0,20], S[45,255], V[0,255] &
Y[60,255], Cr[140,210], Cb[90,130]

This values work best for a constant lighting condition and lighter skin color. To adjust the threshold value according to the usage scenario track bar control is also provided. This method is useful where the intensity of the hand is almost similar whenever the system is used. Also the background intensity should be similar every time the system is used. But even in constant lighting conditions during every system use the system might fail depending on the user's hand color. If the user's hand is also darker in color, the system might not be able to separate the user's hands and the dark background. After segmentation a Gaussian and median smoothing techniques are used to reduce noise due to lightning conditions. Figure 2 shows a sample segmented image after all smoothing techniques are applied.

Figure 2: Sample image, image after color segmentation.

IV. HAND DETECTION

1) Contour

A contour is the curve for a two variables function along which the function has a constant value. A contour joins points above a given level and of equal elevation. The contour is drawn around the white blob of the hand that is found out by thresholding the input image. There can be possibilities that more than one blob will be formed in the image due to noise in the background. So the contours are drawn on such smaller white blobs too. Considering all blobs formed due to noise are small, thus the large contour is considered for further processing specifying it as a contour of hand.

In this implementation, after preprocessing of the image frame, white blob is formed. Contour is drawn around this white blob.

For finding contours in OpenCV using C we use the following function.

IntcvFindContours(CvArr* image, CvMemStorage* storage, CvSeq** first_contour, int header_size=sizeof(CvContour),

Int mode=CV_RETR_LIST,int method=CV_CHAIN_APPROX_SIMPLE, CvPoint offset=cvPoint(0,0))

This function returns a sequence of contours from the segmented image. To draw the contours on the image we use the draw contour function which draws external contour and holes using different colors.

void cvDrawContours(CvArr img, CvSeq* contour, CvScalar externalColor, CvScalar holeColor, int maxLevel, int thickness=1, intlineType=8)*

We then can filter the smaller blobs using the contour area so that the blob with the maximum contour area can be considered as hand contour as depicted in figure 3.

Figure 3: Hand Contour

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 7, Issue 7, July 2020

2) Convex Hull

The convex hull of a set of points in the Euclidean space is the smallest convex set that contains all the set of given points. For example, when this set of points is a bounded subset of the plane, the convex hull can be visualized as the shape formed by a line stretched around this set of points.

Convex hull is drawn around the contour of the hand, such that all contour points are within the convex hull. This makes an envelope around the hand contour.

An approximate polygon is drawn before using the convex hull function. A sequence of points is then extracted from the approximate polygon to get the corresponding convex hull as depicted in figure 4. For detecting the approximate polygon and convex hull of a contour the following functions are used.

CvSeq cvApproxPoly(const void* src_seq, int header_size, CvMemStorage* storage, int method, double eps, int recursive=0)*

CvSeq cvConvexHull2(const CvArr* input, void* hull_storage=NULL, int orientation=CV_CLOCKWISE, int return_points=0)*

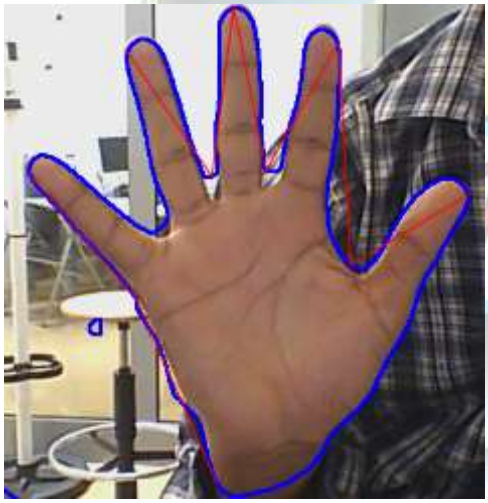

Figure 4: An approximate polygon to extract the convex hull (red line)

3) Convexity Defect

When the convex hull is drawn around the contour of the hand, it fits set of contour points of the hand within the hull. It uses minimum points to form the hull to include all contour points inside or on the hull and maintain the property of convexity. This causes the formation of defects in the convex hull with respect to the contour drawn on hand. A defect is present wherever the contour of the object is away from the convex hull drawn around the same contour. Convexity defect gives the set of values for every defect in the form of vector. This vector contains the start and end point of the line of defect in the convex hull. These points indicate indices of the coordinate points of the contour. These points can be easily retrieved by using start and end indices of the defect formed from the contour vector. Convexity defect also includes index of the depth point in the contour and its depth value from the line. Due to this requirement on the minimum depth required for consideration as the space between two fingers, the system may not detect some fingers if the user's hand is far away from the web camera. The convexity defects are used to count fingers on feature extraction stage. Figure 5 shows the start and end defect points using different colors.

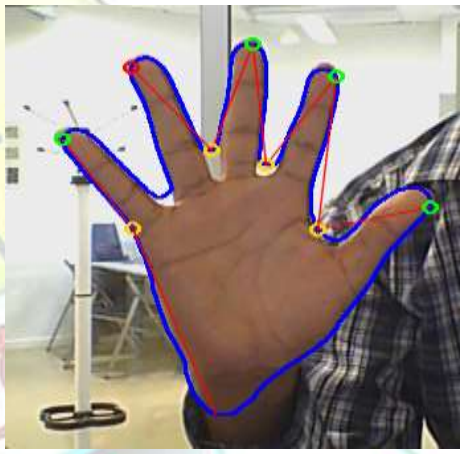

Figure 5: Convexity defects points

Feature extraction

After all pre-processing on the input image frame mentioned in hand segmentation and hand detection is done, useful information is extracted from the user's hand for input purposes. For extracting useful input, two main things that are mentioned in hand detection contour and convexity defects in the image frame are mainly used for extraction of inputs. In this project we used number of fingers and hand orientation to distinguish the hand gestures.

4) Number of fingers

In this method, the count of fingers from the user's hand is extracted. It makes use of convexity defects for detecting the finger count. As mentioned above, the data structure of convexity defect gives the depth of the defects. Such defects can occur in a general hand image due to wrist position and orientation. But some defects have far greater depth than others. These are the gaps between the two fingers. As shown in the figure 6, count

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 7, Issue 7, July 2020

given by the user is two. There are many defects that are formed, but the depth of defect formed due to the gap between two fingers is much greater and thus can be separated from other non-important defects. For two adjacent fingers, there is one such defect.

This technique fails when there is no large defect in the input image frame. This situation occurs when there is one finger or no finger in the image. To overcome this issue we design the commands from count of 2 to 5. The count of 1 and 0 can also be used as a command but they both represent the same command.

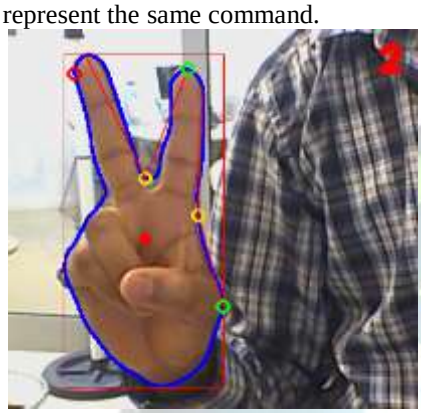

Figure 6: Extraction of finger count

5) Hand Orientation

Three possible orientations can be recognized such as up, right and left. To get the hand orientation we use bounding rectangle around the contour. The center of the bounding rectangle and centroid of the contour are used as distinguishing features. The tip of the finger which is a point farthest from the center of the contour is extracted by comparing the distance from the centroid to all convex defect points. The tip of the finger is used to know the direction of the pointing finger. The tip of the finger is the solid purple point as shown in figure 7.

Comparing the width and height of the bounding rectangle, we can differentiate up, left and right gesture. If the height is greater than the width the hand is in upright position which can be forward or stop gesture. The two gestures can be differentiated using finger count. If finger count is one or two considering the error in finger count, Forward gestures is selected. If the number of fingers is greater than three, Stop gesture is selected. However if the width is greater than the height the gesture must be Turn left or turn right. To further distinguish the gestures we use the position of fingertip with respect to the center of the rectangle. i.e. if the fingertip is to the right of the center, the gesture is turn right and if it is to the left of the center ,the gesture is turn left. Figure 7 shows the actual result of our gesture recognition.

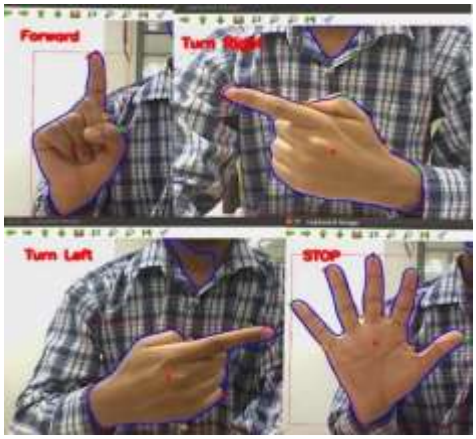

Figure 7: Gesture recognition results

V. Experimental Results

We have conducted experiments based on images we have acquired using a 4 Mega-Pixel digital camera as well as a simple webcam. We have collected these data on uniform as well as cluttered backgrounds. Figure 6 shows a sample result for a hand image displaying the count “two”. We showed the output of various stages of our algorithm.

When the hand gestures were made before the webcam, the program successfully implemented the OpenCV track bars, the segmentation of images into black and white, the detection of blobs, the recognition of the images, and the output of the control commands.

The results also showed that the gesture recognition application was quite robust for static images. However, the video version was enormously affected by the amount of illumination, such that it was necessary to check and adjust the HSV values for skin color when starting the program to get the proper output. Sometimes the adjustment was difficult to do because of the lighting conditions and the amount of objects in the background.

Moreover, the application was very susceptible to noise on the video stream. Slight hand movements could affect gesture recognition. Nevertheless, if the hand is steady enough, the program outputs the correct command. It was also observed that while the program was executing there were memory leaks. Attempts to remedy the problem were made by using the OpenCV functions to release memory. Despite this, the leaks continued. Perhaps the leaks were due to the implementation of OpenCV functions for the sequences behind the scenes.

VI. Conclusion

We proposed a fast and simple algorithm for a hand gesture recognition problem. Given observed images of

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 7, Issue 7, July 2020

the hand, the algorithm segments the hand region, and then makes an inference on the activity of the fingers involved in the gesture. We have demonstrated the effectiveness of this computationally efficient algorithm on real images we have acquired. Based on our motivating robot control application, we have only considered a limited number of gestures.

Our algorithm can be extended in a number of ways to recognize a broader set of gestures. Such applications lack accuracy to be implemented using webcams mounted on mobile robots since the movements of the robot and hand could affect the gesture recognition. However it would be convenient for remote control of a robot from stationary platform. A lot has to be done to get a robust hand detection algorithm which cannot be affected by lightning conditions and movements of the hand.

As this work could also be done for a computer human interaction to replace a computer mouse or touchpad using hand gesture. Finding the tip of the finger could be promising to replace the computer cursor. The clicking, dragging and scrolling action can also be done using a combination of other hand gestures with fingertip position. This application could be more robust than robot control due to the possible constant lighting and limited relative movement of the hand.

REFERENCES

[1]. J. Davis and M. Shah "Visual Gesture Recognition", IEE Proc.-Vis. Image Signal Process, Vol. 141, No.2, April 1994.

[2]. C.-C. Chang, I.-Y Chen, and Y.-S. Huang, "Hand Pose Recognition Using Curvature Scale Space", IEEE International Conference on Pattern Recognition, 2002.

[3]. A. Utsumi, T. Miyasato and F. Kishino, "Multi-Camera Hand Pose Recognition System Using Skeleton Image", IEEE International Workshop on Robot and Human Communication, pp. 219-224, 1995.

[4]. Y. Aoki, S. Tanahashi, and J. Xu, "Sign Language Image Processing for Intelligent Communication by Communication Satellite", IEEE International Conf. On Acoustics, Speech, and Signal Processing, 1994.

[5]. R. C. Gonzalez and R. E. Woods, Digital Image Processing, Prentice-Hall, 2nd edition, 2002.

[6]. R. M. Haralick, and L. G. Shapiro, Computer and Robot Vision, Volume I, Addison-Wesley, 1992, pp. 28-48.

[7]. Malima, Asanterabi Kighoma and Özgür, Erol and Çetin, Müjdat (2006) A fast algorithm for vision-based hand gesture recognition for robot control. In: IEEE 14th Signal Processing and Communications Applications, Antalya

[8]. N.M. Oliver, B. Rosario, and A.P. Pentland, "A Bayesian computer vision system for modeling human interactions," IEEE Trans. on Paftern Anal. and Machine Zntell., vol. 22, no. 8, pp. 831-843,2000.

[9]. Nir Friedman and Stuart Russell. "Image segmentation in video sequences: A probabilistic approach," In Proc. of the Thirteenth Conference on Uncertainty in ArtzJcial Intelligence(lJA I), Aug. 1-3, 1997.

[10]. M. Deepan Raj, I. Gogul, M. Thangaraja and V. S. Kumar, "Static gesture recognition based precise positioning of 5-DOF robotic arm using FPGA," 2017 Trends in Industrial Measurement and Automation (TIMA), Chennai, 2017, pp. 1-6, doi: 10.1109/TIMA.2017.8064804.

[11]. Christof Ridder, Olaf Munkelt, and Harald Kirchner. "Adaptive Background Estimation and Foreground Detection using Kalman-Filtering," Proceedings of Intemational Conference on recent Advances in Mechatronics, ICRAM'95, UNESCO Chair on Mechatronics, 193-199, -1995.

[12]. [lo] M. Piccardi, T. Jan, "Efficient mean-shift backgonmd subtraction", to appear in Proc. of IEEE 2004 KIP, Singapore, Oct. 2004.

[13]. Z. Cao, Q. Yin, X. Tang, and J. Sun, "Face Recognition with Learning based Descriptor", Proceedings of Computer Vision and Pattern Recognition (CVPR), 2010.

[14]. OpenCV documentation http://docs.opencv.org/

[15]. Hand Tracking And Gesture Detection (OpenCV) http://s-ln.in/2013/04/18/hand-tracking-and-gesture-detection-opencv/

[16]. OpenCV Tutorials for Hand Gesture Detection and Recognition http://www.intorobotics.com/9-opencv-tutorials-hand-gesture-detection-recognition/#UI32pLcIkQkVJyEd.99