The Gesture Replicating Robotic Arm

Sunil Karamchandani[1]
1. Professor EXTC,
2. DJSCOE, Vile – Parle (W)
3. Mumbai, India
E-mail: skaramchandani@rediffmail.com

Satyajit Sinari[2],Amrita Aurora[2] ,Dharmesh Ruparel[2]
1. U. G. Student EXTC,
2. DJSCOE, Vile – Parle (W)
3. Mumbai, India
E-mail: satyajit_789@yahoo.com,
amritaaaurora@gmail.com,
ruparel_dharmesh@yahoo.in

***Abstract*—The Gesture Replicating Robotic arm is a servo controlled robotic arm which replicates gestures in a three dimensional environment. It makes use of cameras which detect the motion of one's hand in three dimensions. The cameras provide frames as input to the software which performs segmentation algorithms like background subtraction, color detection and contour detection. Pixel to angle mapping gives the appropriate commands to the respective servo motors. Thus, replication of the human hand movements is done. The crux of the project is to create software which can detect hand gestures without any involvement of sensors like accelerometers attached to the human arm, with no involvement of artificial intelligence. Thus it can avoid large blocks of code to perform a simple function which humans can perform easily like pouring coffee into a mug. That is robots will be able to move like people and they don't need to be intelligent. Such lifelike robotic arms can be tele-operated to protect human workers in hazardous environments, such as on assembly lines, in space, undersea, and amid nuclear radiation.**

Keywords-Background Subtraction, color detection, contour detection.

I. INTRODUCTION

The Gesture Replicating Robotic Arm is a revolution in technology. It is an answer to the absence of human limbs that occur in space, on land at sea underwater, and other inaccessible hazardous environments. The scope of the project is based on the reality of fully functioning prosthetic limbs that can be used in applications in the near future.

The basic aim is to create software which could detect hand gestures without any involvement of sensors attached to the human arm thus making it extremely user-friendly, with minimum time lag and accurate gesture replication of the human arm. [1, 2]

This paper involves the procedures and algorithms required for a wirelessly tele-operated 6-axis robotic arm with motion characteristics similar to a human arm. It presents an overview on the optimum utility of the Gesture Replicating Robotic arm functioning. A detailed discussion of each concept involved in the process is presented. The working principle is discussed below. The block diagram is a manifestation of the project.

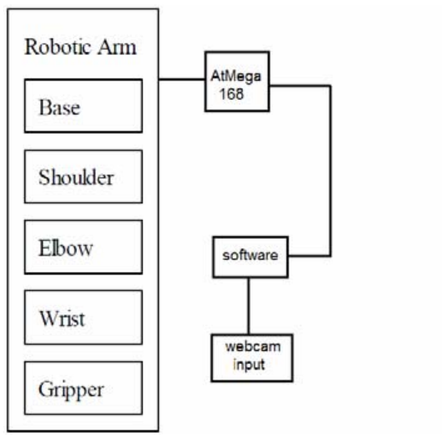


Figure 1. Block diagram of Gesture Replicating Robotic Arm

It consists of the following designed components:

A. The Robotic arm:

Robot Arm has 5 degrees of freedom which includes: Base rotation, Shoulder rotation, Elbow rotation, Wrist pitch and roll.

B. The webcam input:

Two high resolution webcams are used for computer vision, which helps in monitoring the gesture presentation. They detect motion in vertical plane and in horizontal plane.

C. The Software:

The software is written in OpenCv. The platform used is Visual Basic 2010. The data (frames) collected by the webcam will be the input to the program written in this software.

D. The AtMega 168 microcontroller:

Position to angle mapping is performed due to which proper detection of the respective angles can then be performed and then mapped accordingly.

In this technical paper, we present the techniques and drawbacks of each technique experienced while processing frames. A detailed discussion of the algorithms for background subtraction, color detection and contour detection algorithms is thus discussed. Accuracy in image processing enables precise movement of the robotic arm in accordance with the movement of the human arm. Thus the main focus in this technical paper is the effective processing of the image via various techniques mentioned below.

II. Background Subtraction

Background subtraction is a computational vision process of extracting foreground objects in a particular scene. This method of background subtraction is used to detect the wrist and the elbow. [2 & 3]

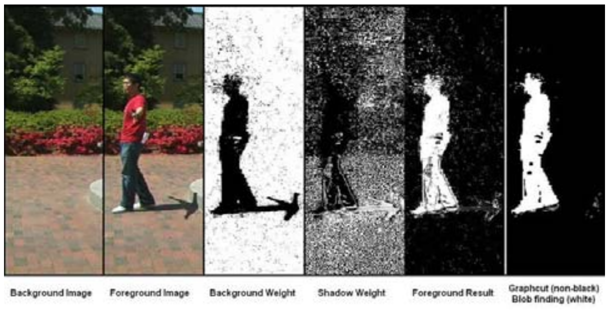
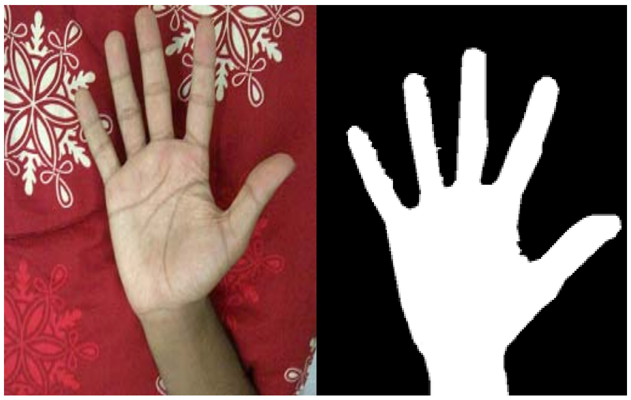


Figure 2. Background Subtraction


(a) (b)
Figure 3.(a) & (b) Implementation of Background Subtraction

The primary colour is then eliminated from the frame using HSV values, leaving the object of interest, the human arm to be processed further. Thus, the frame is simplified. Various approaches were experimented with and then the optimized method was finally implemented. The techniques include the following:

A. Codebook model:
B. Mixture of Gaussians model
C. Color co-occurrence model
D. Skin Detection Model

Skin Detection Model: In this proposed technique, we assign HSV values to each pixel value. The hue, saturation and value are selected such that the color of only the skin is detected. These values range from (0, 50, 50) to (20, 255, 255) for implementation of our code. The RGB values of the image frames are thus converted to HSV and the image is converted to its binary value. This eliminated the drawback of shades of colors since all shades of the same color are assigned a single value. We obtained minimum interference via this method and hence its implementation was easier. The results of all methods are tabulated below:

TABLE I. Results of Background Subtraction Algorithm Implemented

Algorithm	*CPU demanding*	*Training Samples*	*Sensitivity*
Codebook method	40%	15	80%
Gaussian method	50%	10	70%
Color co-occurrence method	60%	10	65%
Skin Detection	30%	No need	40%

III. Wrist and Elbow Detection

A .Thresholding:

This is the conversion of the frame into a binary image (white being the area or color of interest and black being the rest of the frame). In HSV, each “tint” of color is assigned a particular number (the Hue). The “amount” of color is assigned another number (the Saturation) and the brightness of the color is assigned another number (the Intensity or Value). This gives the advantage of having a single number (hue) for the yellow ball despite multiple shades of yellow (all the way from dark yellow to a bright yellow).

B. Position Calculation:

Once the color is detected, the position is found out by mathematical calculations.

C. Moment Calculation:

A memory is allocated to the moment structure and various moments are calculated. Using the moment structure, the two first order moments (moment10 and moment01) are calculated. The zeroth order moment (area) is also calculated. Dividing moment10 by area gives the X coordinate of the color position and dividing moment01 by area gives the Y coordinate. In this way, the position of Wrist and Elbow is calculated. [6 & 7]

IV. Contour Detection

Contour detection is basically the detection of peaks and valleys in the image (i.e. the finger tips and the valleys between them).The techniques include:

A. Convex Hull
B. Quickhull
C. Graham Scan
D. Monotone Chain
E. Pruning (proposed algorithm)

Pruning (proposed algorithm): The pruning happens in two-pass. First, we scan all the points and prune some of them, according to the algorithm described in the next paragraph. Then, we run the same process again on the remaining points to do the final pruning.

For each step, the set of points gets smaller. The advantage of pruning is that we apply the convex hull algorithm (slow) only to a small subset of points, and the pruning algorithm (fast) to all of them. As a result, the total running time should be lower than running the convex hull algorithm on all the points.

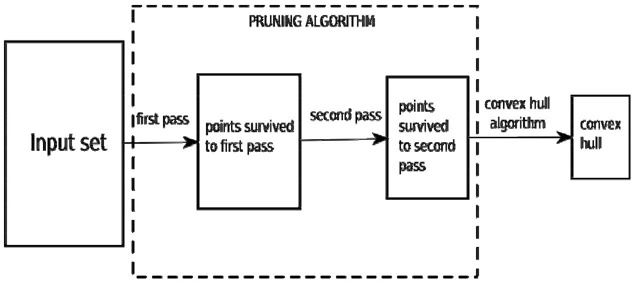


Fig 7. Pruning algorithm

In the first pass, the whole input set is read and the input for the second pass is produced. Since the space is two-dimensional, each point has two coordinates, x and y. Every time we read a new point, we compute the following 4 points:

- A = (Ax, Ay) which maximizes x-y
- B = (Bx, Xy) which maximizes x+y
- C = (Cx, Cy) which minimizes x-y
- D = (Dx, Dy) which minimizes x+y

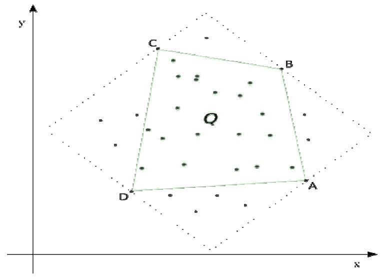


Fig 8. Pruning of points using quadrilateral ABCD (Q)

Let's call Q the quadrilateral having the vertices A, B, C, D. The dashed diamond (DD) is constructed with 2 edges inclined at 45° and 2 at -45°. It's easy to see that none of the read point can be outside DD, otherwise the definition of either A, B, C or D would be contradicted. Q will expand when we update one of A, B, C, D. For example, if we update B, it means that we read a new point with a greater (x + y) than B. The new B will be outside the diamond, hence Q will expand. Finally, any point contained in Q cannot be part of the hull; otherwise the hull itself would not be convex. Therefore, we could theoretically prune all the points inside Q. But the problem is that it's not efficient because it requires the computation of the 4 edges of Q and many floating points operations to decide whether a point is internal to the edges or not. So we find a better approach.

- x = max(Cx, Dx)
- x = min(Ax, Bx)
- y = max(Ay, Dy)
- y = min(By, Cy)
- Build the rectangle R with vertices : (x , y), (x , y), (x , y), (x , y).

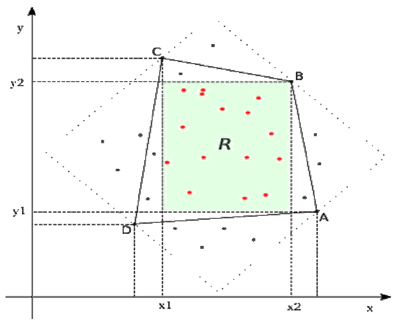


Fig 9. Pruning of points using quadrilateral R

Since all the points inside Q can be pruned, and R is inside Q, it follows that we can prune all the points contained in R, marked in red. Also, R has the same behavior of Q, in that it changes every time we update A, B, C or D. Now we are in a much better position to prune points. In fact, checking whether a point is inside R is a very fast operation which only requires basic comparisons and only integer mathematics if the coordinates are integers. So, in the first pass, if we read a point which is inside R, we can feel free to prune it, because it will never be part of the hull. On the contrary, if the point is outside DD, it is necessary to update one of A, B, C, D, which also implies to update Q and R.

The need of a second pass: At the end of the first pass, after having pruned a (hopefully) large number of points, Q and R cannot change any more, because we found the final values of A, B, C and D. But we can do even better, running a simplified version of the algorithm once again. The points which have survived to the first pass become the input for the second pass. This time, R no longer needs to be updated and we just have to check whether each point is inside R or not. The first pass does not prune all the points simply because we find the final value of R only at the end of it.

The function *prune()* takes a set of points S as input and removes some of the points which cannot be part of the hull. For simplicity, the function changes the content of the passed arguments, rather than returning the new pruned set. The content of S after invoking the function *prune()* is the set of points survived to the pruning and can be passed as input to any convex hull algorithm. Since we had to choose the optimized one, we opted for the Monotone Chain algorithm, because it was simple to implement, but probably a different one like the Graham scan would not have changed significantly the overall performance, since it only runs on a small subset of the input, thanks to the pruning.

TABLE II. RESULTS OF CONTOUR DETECTION ALGORITHMS

Algorithm	*Compile Time*	*Max fps* support*	*Delay (msec)*
Convex Hull method	20	10	100
Graham Scan method	14	40	25
Quickhull method	13.2	40	25
Monnotone-chain method	10	50	20
Pruning method (proposed)	0.18	200	5

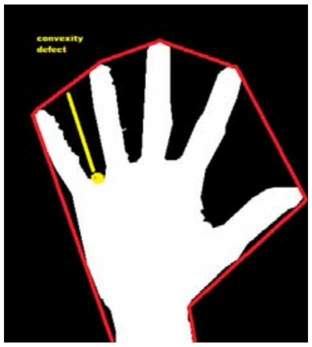


Fig 11. Convex Hull applied to the hand

The red lines are the convex hull of the hand. After this, we get 'defects', i.e. the valley points within the convex hull (the yellow dot is a defect). These are called convexity points. These defects are used to calculate the number fingers that are unfolded.

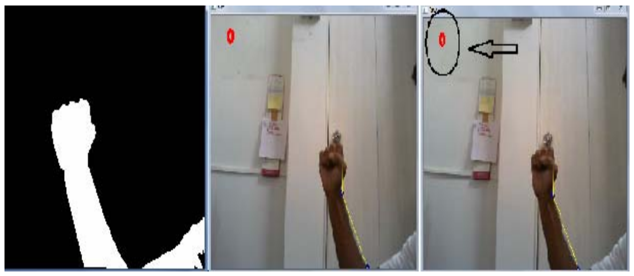


Fig 12. Output for defect '0'

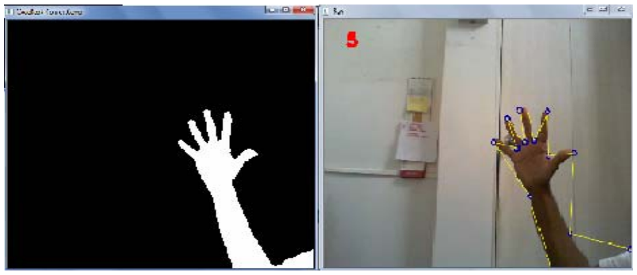
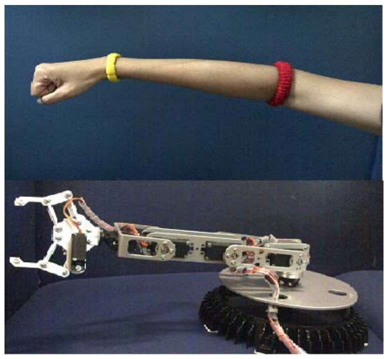


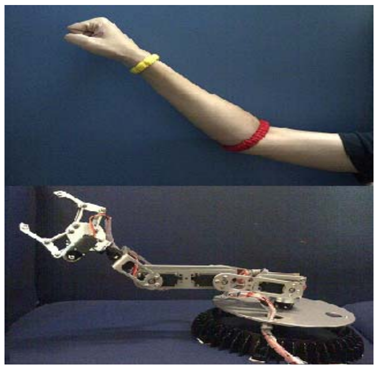
Fig 13. Output for defect '5'

V. PRACTICAL IMPLEMENTATION

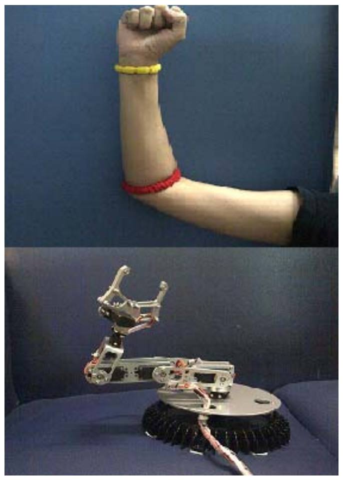
Some examples of the positions of the human arm and the corresponding position of the robotic arm implemented practically are shown below:



(a)



(b)



(c)

Fig 14.(a)Position 1 of robotic arm & human arm horizontal.
Fig 14.(b) Position 2 of robotic arm & human arm at 45 degree
Fig 14.(c) Position 3 of robotic arm & human arm at 90 degree.

VI. Applications

- Substitute of a human arm in inaccessible environments: In areas where it is practically impossible for humans to survive. E.g. in extremely high or low temperatures. The robotic arm can be used and various actions can be performed with the human controlling those actions miles apart.
- In medical science: "Neuro-arm" uses miniaturized tools such as laser scalpels with pinpoint accuracy.
- Coal mines: In hazardous environment, it becomes practically impossible for workers to dig through coal mines with extreme conditions of heat.
- Nuclear or radioactive experiments: Due to the use of metal instead of flesh, it becomes easier to perform radioactive experiments with the robotic arm, ensuring no loss of human life.
- Bomb defusing: Bomb defusing is an extremely useful application. Bombs can be defused by special squads without risking their lives and maneuvering and controlling the robotic arm to disarm the bomb efficiently. [11,12]

References

[1] Chris Carter,Brian Dwyer, Rachel Lutsic,"Wireless User Controlled Robotic Arm",Central Michigan University, Mt. Pleasant, MI 48859

[2] Amithash E. Prasad, Murtuza Tambawala ,"Human Controlled Remote Robotic Arm"(HCRRA)

[3] N. Martel-Brisson & A. Zaccarin's , "Moving Cast Gesture Replicating Robotic Detection from a Gaussian Mixture Gesture Replicating Robotic Model".Graphcut Background Subtraction.

[4] T. Bouwmans, "Background Subtraction For Visual Surveillance: A Fuzzy Approach", Handbook on Soft Computing for Video Surveillance, Taylor and Francis Group, Chapter 5, March 2012.

[5] T. Bouwmans, F. El Baf, B. Vachon, "Statistical Background Modeling for Foreground Detection: A Survey", Handbook of Pattern Recognition and Computer Vision, World Scientific Publishing, Volume 4, Part 2, Chapter 3, pages 181-199, January 2010.

[6] T. Bouwmans, "Recent Advanced Statistical Background Modeling for Foreground Detection: A Systematic Survey", Recent Patents on Computer Science, Volume 4, No. 3, pages147-176, September 2011.

[7] T. Bouwmans, "Subspace Learning for Background Modeling - A Survey", Recent Patents on Computer Science, Recent Patents on Computer Science, Volume 2, No 3, pages 223-234, November 2009.

[8] T. Bouwmans, F. El Baf, B. Vachon, "Background Modeling using Mixture of Gaussians for Foreground Detection - A Survey", Recent Patents on Computer Science, Volume 1, No 3, pages 219-237, November 2008.

[9] J. R. Parker,Algorithms for Image Processing and Computer VisionChristopher M. Holt , Alan Stewart , Maurice Clint ,Ronald H. Perrott, "An improved parallel thinning Algorithm", Communications of the ACM, v.30 n.2, p.156-160, Feb. 1987

[10] T. Y. Zhang , C. Y. Suen, "A fast parallel algorithm for thinning digital patterns", Communications of the ACM, v.27 n.3, p.236-239, March 1984

[11] D. Ballard and C. Brown *Computer Vision*, Prentice-Hall, 1982, Chap. 8.

[12] E. Davies *Machine Vision: Theory, Algorithms and Practicalities*, Academic Press, 1990, pp 149 - 161.