

Real-time Actuation of Cylindrical Manipulator model in OpenGL based on Hand gestures recognized using Open CV

Subash chandar¹, Willson Amalraj², Gnana sambandam³
Department of Electrical and Computer Engineering,
National University of Singapore, 21 Lower Kent Ridge Road, Singapore 119077.

Abstract: This paper proposes a method of actuating robotic cylindrical manipulator model in OpenGL based on hand gestures recognized using OpenCV (Open Source Computer Vision Library) in VC++. A model of cylindrical manipulator is developed using OpenGL (Open Graphics library) in VC++, which is actuated for the recognized hand gestures. The fingers and their orientation with respect to a reference base plane are recognized for different gestures using OpenCV. The actuator in each joint of cylindrical manipulator is selected by specific hand gesture. The movement of the actuator is controlled by orientation of the hand with respect to base plane. The hand gesture recognition and cylindrical manipulator module are developed using Windows Application Program Interface (WINAPI), standard C++ libraries, OpenCV libraries and OpenGL libraries in visual studio 2010. Real time simulation of the prototype model developed in VC++ is demonstrated with successful gesture recognition and actuation of manipulator.

Keywords: Hand gestures, cylindrical manipulator, Open Source Computer Vision Library (OpenCV), Open Graphics Library (OpenGL), Windows Application Program Interface (WINAPI), Visual C++.

I. INTRODUCTION

Human interaction with computers is constantly evolving in order to increase the efficiency and effectiveness by which interactive tasks are performed. Nowadays, devices such as keyboards, mouse, joysticks or gamepads are used for human-computer interaction (HCI). In the recent years a growing interest in methods of natural actions recognition in human-to-human communication such as speak and gesture in a more effective way is seen [1]-[2]. Hand Gestures which has become popular in recent years can be used to develop a Human computer interaction device [3]-[6]. Hand gestures recognition by Computer will provide a more natural human-computer interface, allowing people to point, or rotate a CAD model in a virtual environment, by rotating their hands. Gesture recognition may even be useful to control household appliances [7].

Hand gestures can be recognized by vision-based or data glove based methods. Vision-based hand gesture recognition methods are based on the quality of the captured images [8]-[10]. Because of the limitations of the optical sensors to lighting conditions and cluttered backgrounds, vision based approaches are not efficient in detecting and tracking the hands robustly, which largely affects the performance of hand gesture recognition [11]. Data glove such as Cyber glove can be used to enable a more robust hand gesture recognition which effectively captures the hand gesture and motion using embedded

sensors [12]-[13]. Unlike optical sensors, Cyber gloves are more reliable and are not affected by lighting conditions or cluttered backgrounds. However, it requires the user to wear a data glove and sometimes requires calibration which is inconvenient for the user. Moreover the cost of data glove is much higher when compared to conventional webcam used for vision-based approach of hand gesture recognition. This paper deals with Computer vision based gesture recognition system [14].

The gestures are classified into two categories such as static and dynamic. A static gesture is represented by a single image corresponding to a particular hand configuration and pose. A dynamic gesture is a moving gesture, represented by a sequence of images. In this project dynamic gesture recognition is used to identify the finger counts in the recognized hand, which is done by processing the video acquired from a webcam [15]-[17]. The finger counts obtained for specific hand gestures are used to select the appropriate actuator in a cylindrical manipulator model developed in OpenGL. The orientation angle obtained from the hand gesture recognition system is used to control the movements of actuators in the manipulator model. The selection and control of actuators in manipulator is done effectively in real-time without time delay.

The rest of the paper is organized as follows. In section II, a brief overview of OpenCV is presented. Section III gives an introduction to OpenGL. The proposed architecture for image processing and actuation of robotic cylindrical manipulator is discussed in detail in Section IV. In section V, the results of the proposed system are discussed. Section VI concludes the paper and the future works are presented in section VII.

II. OPEN CV

The OpenCV (Open Source Computer Vision Library) is a cross-platform library of programming functions developed by Intel for real-time computer vision. It is free for use under the open source BSD license. Its main focus is on the real-time image processing. The library will make use of Intel's Integrated Performance Primitives, if available in the system which has optimized routines to accelerate itself. OpenCV aids advanced vision research by providing not only open but also performance optimized code for basic vision infrastructure. The library was originally written in C which makes OpenCV portable to some specific platforms such as digital signal processors. However since version 2.0, OpenCV includes both its traditional C interface as well as a new C++ interface. The number of lines of code for vision functionality is reduced in OpenCV 2.0. It also reduces common programming errors such as memory leaks through automatic data allocation and de-allocation that may arise when using OpenCV in C. wrappers in other languages to C++ is not

developed as opposed to C code. OpenCV libraries can be accessed by Visual C++ in windows operating systems. However the Base Classes from DirectShow SDK is required to access webcams on Windows with DirectX Media SDK prior to 6.0. Facial recognition system, gesture recognition, human computer interaction (HCI), motion tracking, etc. are a few applications of OpenCV.

III. OPEN GL

OpenGL (Open Graphics Library) is a cross-language, multi-platform application program interface developed by Silicon Graphics Inc., for developing and simulating applications, that produce 2D and 3D computer graphics. OpenGL accepts primitives such as points, lines and polygons, and converts them into pixels via a graphics pipeline known as the OpenGL state machine. The interface can be used to draw complex three-dimensional scenes from simple primitives. OpenGL presents a single, uniform interface for different 3D accelerator hardware and supports full feature set, using software emulation if necessary, for all implementations. OpenGL is a low-level, procedural API, requiring the programmer to have a good knowledge of the steps required to render a scene. Hence it provides certain amount of freedom to implement novel rendering algorithms in contrast with descriptive APIs, where a programmer only needs to describe a scene and can let the library manage the details of rendering it. Several libraries are built beside OpenGL to compensate for features not available in OpenGL. Libraries such as GLU, GLUT, SDL and Allegro have been developed for aiding OpenGL for rudimentary cross-platform windowing and mouse functionality. Simple graphical user interface functionality can be found in libraries like GLUT or FLTK. OpenGL is widely used in CAD, virtual reality, scientific visualization, information visualization, flight simulation, and video games.

IV. PROPOSED SYSTEM ARCHITECTURE

The architecture has two different modules, Image processing and cylindrical manipulator module working with synchronization. The flow diagram of the proposed algorithm is shown in Fig.1. The detailed working principle of each module in the algorithm is discussed in this section.

A. Image processing Module

The Image processing module process the image acquired to recognize the gesture. The entire module is developed in VC++ using OpenCV libraries and WINAPI (windows application programming interface). The code is developed in 32 bit, windows7 OS, hence 32 bit version of OpenCV and WINAPI dynamic link libraries. This module consists of different stages with specific functions to identify the hand and to recognize the gesture. The functionality of each stage is discussed in detail in the following sections.

1) Image Acquisition

The image of the hand is acquired through webcam, it can be either integrated or standalone. Webcams can be directly accessed using OpenCV libraries or DirectX libraries present in windows SDK (Software development kit). The webcam is configured such that the frame rate is 30 and the resolution is fixed to be 480 x 640 (VGA). The

capturing resolution can be increased depending on the webcam available, which in turn will increase the processing time of the algorithm.

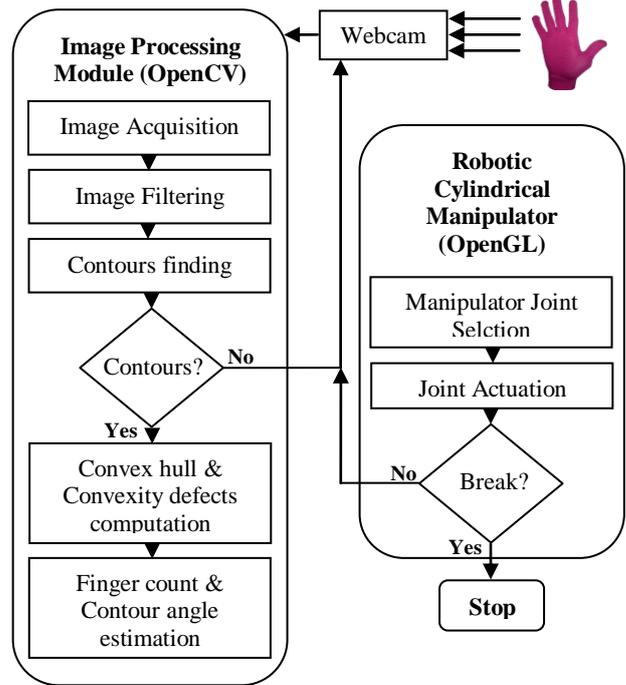


Fig.1 Flow diagram of proposed system

2) Image filtering

The captured image is filtered to extract the pixels retaining the information pertaining to the hand. Attempt was done to recognize the bare hand, but hard coding it, will not be useful, as the complexion differs for different person. Moreover if a range is fixed for filtering the image, it will lead to other objects in surrounding to appear in the filtered image. Hence the natural choice for easy hand detection will be a glove. The program is hard coded to recognize the hand with pink or blue glove.

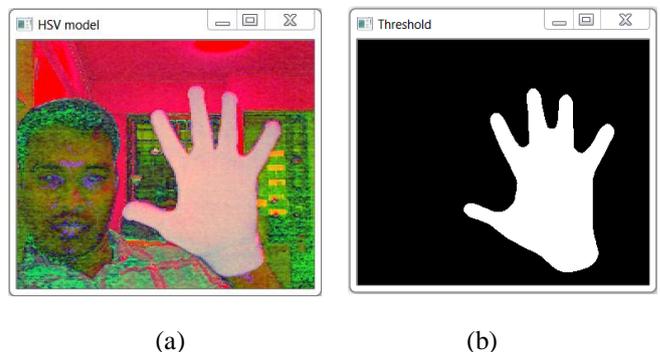


Fig.2 Image captured using webcam (a) HSV format & (b) Final filtered image using CvThreshold

The image captured by webcam in BGR format (3 channels) is flipped to match with the same direction as that of gesture movement in front of webcam. The BGR format is converted to HSV format using cvCvtColor() in OpenCV as shown in Fig.2 (a), which is then filtered to remove the backgrounds except hand, using cvInRange(). cvInRange() compares the source image with the corresponding value in the lower and upper arguments. If the value in the source image is greater than or equal to the value in lower

argument and also less than the value in upper argument, then the corresponding value in destination will be set to 1 otherwise, the value in will be set to 0. Hence the image filtered after `cvInRange()` will be of binary format (1 channel). The noise in the image is then removed by Median filter using `cvSmooth()`. Finally the required pixels with dominant features are separated from the other pixels using `cvThreshold()` as shown in Fig.2 (b).

3) Contours, convexity Hull and Convexity Defects Computation

The final filtered image is used to compute the contours with `cvFindContours()` function. The function `cvFindContours()` computes contours from binary images or images with edge pixels. Contours are represented in OpenCV by sequences in which every entry in the sequence encodes information about the location of the next point on the curve. Once the contour is obtained the next logical step is to comprehend the shape of the processed image.

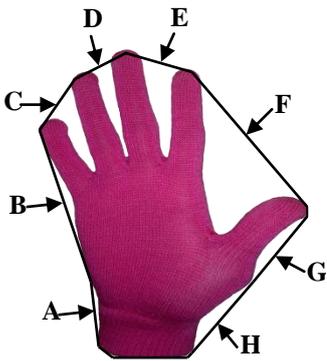


Fig.3 Convex hull and convexity defects of a hand

The convex hull of the contour obtained from recognized hand is pictured as a dark line around the hand in Fig.3, and the regions labeled A through H are each “defects” relative to that hull. Convexity defects characterize not only hand but also its state. Hence the number of fingers can be easily estimated for different gestures.

The number of fingers is used to select the specific actuator in the manipulator. The direction of manipulator actuation has to be controlled by some other controlling parameters. Orientation of the hand contour is chosen as the controlling parameter. The detail of the hand contour formed is summarized with a bounding ellipse using `cvEllipseBox()`. The angle of the bounded ellipse is used to control the direction of actuator movement.

B. Robotic Cylindrical Manipulator Module

A cylindrical manipulator is designed using OpenGL and WINAPI in VC++. The manipulators consist of three joints, a prismatic joint for translating the arm vertically, a revolute joint with a vertical axis and another prismatic joint orthogonal to the revolute joint axis. The each segment of robotic manipulator is designed by a scaled cube using `glutSolidCube()` function in OpenGL as show in Fig.4. If a cube is drawn in OpenGL the center of the cube will be at local coordinate of system. Hence, if the cube is made to rotate, it will do so about its center rather than the base. This is avoided by moving the local coordinate system to one edge of the cube by calling appropriate modeling transformation `glTranslate()`. The same procedure is

repeated for all the segments in the model to orient with respect to base cube which serves the platform.

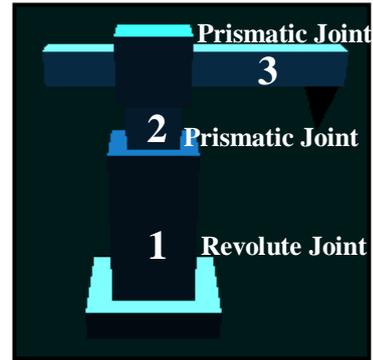


Fig.4 Robotic cylindrical manipulator in OpenGL

The rotation and translational movement is achieved by establishing pivot points for each segments, using `glRotate()` and `glTranslatef()` respectively. In OpenGL the transformations are done by matrix multiplication, hence the position and orientation of the coordinate system for each segment must be saved and restored at appropriate moments. Failure to do so will result in loss of degree of freedom for other segments. This is overcome by using `glPushMatrix()` and `glPopMatrix()` functions in OpenGL. The function `glPushMatrix()`, copies the current matrix and adds the copy to the top of the stack, and `glPopMatrix()`, discards the top matrix on the stack. Whenever a segment is made to move, the contents of the current window will be changed and it has to be redisplayed. It is done by declaring all the routines needed to redraw the scene in the display callback function, `glutDisplayFunc()`.

The material properties of the objects in the scene like, the ambient, diffuse, and specular colors, the shininess, and the color of any emitted light are also declared using `glMaterialfv()` in display callback function. The function `glLightfv()` is used to set the proprieties such as color, position, and direction of light sources in the scene which are declared in a separate one time call function and it remains unchanged.

1) Manipulator Joint Selection

The cylindrical manipulator has three joints capable of translational or rotational movement. Each joint is selected by a specific gesture recognized by image processing module. The gestures and their corresponding joint are depicted in Table I.

TABLE I. HAND GESTURES AND DIRECTION OF ACTUTATION

Hand gestures	Finger Count	Direction of actuation	Actuator
 Contour angle > 135°	1	Counter clockwise	Revolute Joint

 Contour angle < 135°			
 Contour angle > 135°	2	Up	Prismatic Joint 1
 Contour angle < 135°		Down	
 Contour angle > 135°	3	Backward	Prismatic Joint 2
 Contour angle < 135°		Forward	

The primary feature extracted from the gesture for joint selection is finger count. Finger count is computed from convexity defects of recognized hand. The count one corresponds to revolute joint, two for prismatic joint with vertical movement, and three for prismatic joint with horizontal movement as in Table I.

2) Manipulator Joint Actuation

The joint actuation is achieved by computing the hand contour angle computed in image processing module. The angle of hand contour is the angle subtended by major axis of ellipse which bounds the hand contour, with respect to the base plane (x-axis). The angle subtended by hand contour on quadrant-II is considered for actuation. As all the joints in manipulator have two degrees of freedom, it can be made to move in two different directions. The angle subtended is divided into two regions, 135° to 180° for one direction and 90° to 135° degrees for other direction. The joints and their corresponding movements with respect to two different set of angles are shown in Table I.

V. RESULTS AND DISCUSSION

The image processing module (OpenCV) is run as a thread while the robotic cylindrical actuator (OpenGL) module is the main function. Once the program starts execution, the OpenGL module will appear and will wait for response from OpenCV module.

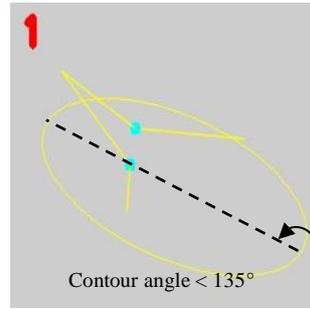


Fig.5 Revolute joint selection and actuation

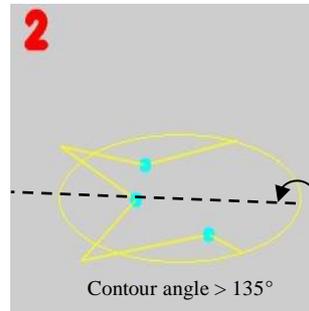


Fig.6 Prismatic joint 1 selection and actuation

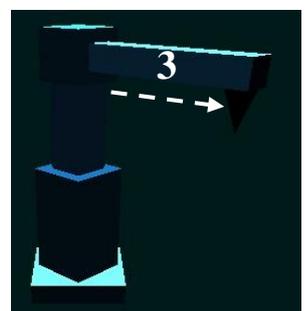
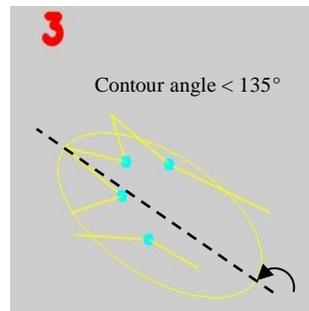


Fig.7 Prismatic joint 2 selection and actuation

When a hand with pink glove is exposed to webcam, contour is formed and the computation of finger count and contour angle will take place in OpenCV thread. Depending on the finger count appropriate joint is selected whose actuation depends on the contour angle. The finger count of hand contour, selected joint and direction of joint actuation are shown in Fig.5 to Fig.7. As the gesture is recognized dynamically the joint selection and actuation takes place without any considerable delay. As OpenCV and OpenGL modules use their specific dynamic linked libraries present in their respective software packages a redistributable package is created using Visual studio 2010, which can be run in other PCs without VC++.

VI. CONCLUSION

The hand gestures are recognized in real-time using OpenCV module developed in VC++. The dynamic gestures are recognized with good accuracy without any time lag. The recognized gesture is able to actuate the robotic cylindrical manipulator designed in OpenGL in real-time. The proposed method of hand gesture recognition and actuation is successful in recognizing and actuating in real-time.

VII. FUTURE WORK

A communication module has been developed for interface with the real robotic cylindrical manipulator in VC++. A serial port interface with a microcontroller is designed to provide communication between software modules running in a PC and the robotic manipulator. The communication module has been tested for different gestures and it is able to generate good result. Eventually the future work will be integrating the OpenCV, OpenGL and communication module with a real robotic cylindrical manipulator. The actuation of real robotic cylindrical actuator will be evaluated.

REFERENCES

- [1] O'Hagan, R. & Zelinsky, A. (1997) "Finger Track – A Robust and Real-Time Gesture Interface". Australian Joint Conference on AI, Perth.
- [2] Agrawal, T. and Chaudhuri, S., 2003. Gesture Recognition Using Position and Appearance Features, ICIP pp. 109–112.
- [3] Von Hardenberg, C. and Berard, F., "Bare-hand human-computer interaction", in [PUI '01: Proceedings of the 2001 workshop on Perceptive user interfaces], 1{8, ACM, New York, NY, USA (2001).
- [4] Malassiotis, S. and Srinivasan, M. G., "Real-time hand posture recognition using range data", *Image Vision Comput.* 26(7), 1027-1037 (2008).
- [5] McAllister, G., McKenna, S. J., and Ricketts, I. W., "Hand tracking for behavior understanding", *Image Vision Comput.* 20(12), 827-840 (2002).
- [6] Ionescu, B., Coquin, D., Lambert, P., and Buzuloiu, V., "Dynamic hand gesture recognition using the skeleton of the hand", *EURASIP J. Appl. Signal Process.* 2005(1), 2101-2109 (2005).
- [7] William T. Freeman and Michal Roth "Orientation histograms of hand gesture Recognition", IEEE Intl. Workshop. on Automatic Face and Gesture Recognition, Zurich, June, 1995.
- [8] Ueda, E., 2003. A Hand Pose Estimation for Vision-Based Human Interfaces, *IEEE Transactions on Industrial Electronics*, Vol. 50, No. 4, pp. 676–684.
- [9] Davis, J. and Shah, M., 1994. Visual Gesture Recognition, *Vision, Image and Signal Processing.* 141(2), pp. 101–106.
- [10] Pavlovic, V.I., Sharma, R. & Huang, T.S. (1997), "Visual Interpretation of Hand Gestures for Human- Computer Interaction: A Review. In *IEEE Transactions on Pattern Analysis and Machine Intelligence.* 19 (7) 677-695.
- [11] Jingling Meng, Junsong Yuan, "Depth Camera Based Hand Gesture Recognition and its Applications in Human-Computer-Interaction", 8th International Conference on Information, Communications and Signal Processing (ICICS) 2011.
- [12] Subash chandar, Vignesh Kumar and Willson Amalraj, "Real Time Data Acquisition and Actuation via Network for Different Hand gestures using Cyber glove", *International Journal of Engineering Research and Development*, ISSN: 2278-067X, Volume 2, Issue 1 (July 2012), PP. 50-54.
- [13] Jiangqin, W., wen, G., yibo, S., wei, L., and bo, P., "A simple sign language recognition system based on data glove", *Signal Processing Proceedings, 1998. ICSP 98. 1998 Fourth International Conference on*, 2, 1257{1260 vol.2 (1998).
- [14] G.R. Bradski, V. Pisarevsky, "Intel's Computer Vision Library", *Proc of IEEE Conference on Computer Vision and Pattern Recognition, CVPR00*, v. 2: 796-797, 2000.
- [15] Oka, K., Sato, Y., and Koike, H., "Real-time fingertip tracking and gesture recognition", *Computer Graphics and Applications, IEEE* 22, 64-71 (Nov/Dec 2002).
- [16] Malima, A. et al. 2006. "A fast algorithm for vision-based hand gesture recognition for robot control", 14th IEEE Int. Conf. on Signal Processing and Communications Applications, Antalya, Turkey.
- [17] Brethes, L. et al. 2004. "Face Tracking and hand gesture recognition for human robot interaction", *Int. Conf. on Robotics and Automation*, Vol. 2, pp. 1901–1906 New Orleans.