Automation of a Wheelchair Mounted Robotic Arm using Computer Vision Interface

Priyanka Karuppiah, Hem Metalia and Kiran George,

Department of Computer Engineering,
College of Engineering and Computer Science,
California State University Fullerton
Fullerton, California 92831
k.priyanka@csu.fullerton.edu

***Abstract*— Assistive robotic devices have great potential to improve the quality of life for individuals suffering with movement disorders. One such device is a robot-arm which helps people with upper body mobility to perform daily tasks. Manual control of robot arms can be challenging for wheelchair users with upper extremity disorders. This research presents an autonomous wheelchair mounted robotic arm built using a computer vision interface. The design utilizes a robotic arm with six degrees of freedom, an electric wheelchair, computer system and two vision sensors. One vision sensor detects the coarse position of the colored objects placed randomly on a shelf located in front of the wheelchair by using a computer vision algorithm. The other vision sensor provides fine localization by ensuring the object is correctly positioned in front of the gripper. The arm is then controlled automatically to pick up the object and return it to the user. Tests have been conducted by placing objects at different locations and the performance of the robotic arm is tabulated. An average task completion time of 37.52 seconds is achieved.**

***Index Terms*—Assistive Technology, Automation, Computer Vision, Image Processing, Robotic Arm, Vision Sensor**

I. Introduction

Robotic arms mounted on a wheel chair largely benefit people suffering from upper body movement disorders such as Amyotrophic Lateral Sclerosis (ALS), Parkinson's disease, Progressive Muscular Atrophy (PMA). They can enhance the manipulation capabilities for electric wheelchair users and make them feel more independent. But wheelchair mounted robotic arms (WMRA) available in the market such as JACO Robot arm by Kinova Robotics [1] and iARM by Assistive Innovations [2] utilize remote controlled interface such as joystick to control the arm. This might be challenging to operate for users suffering from motor neuron diseases and in many cases it can consume a lot of time and effort to successfully grasp the object. They are also very expensive usually run into thousands of dollars [1].

Many researchers have worked on achieving a flexible interface between the user and the robotic arm like touch screen, joystick and graphic display [3][4] to select the target object. Studies have been performed in the area of human-computer interaction (HCI) to develop assistive technology (AT) for controlling robotic arms. Indika Pathirage et. al. [5] implemented brain computer interface (BCI) for grasping an object by creating a visual grid to select an object from the scene image. The study showed that using BCI to select an object can consume more time and adds fatigue to the user [5]. Hairong Jiang et. al. [6] performed object detection using Speeded Up Robust Features (SURF) algorithm and the results were used for coarse positioning of the arm and gesture based recognition was used for fine localization. Most of the interfaces are semi-automatic and they require certain level of concentration and participation from the user. Their completion times are also usually longer (around 200 to 300 seconds) [6].

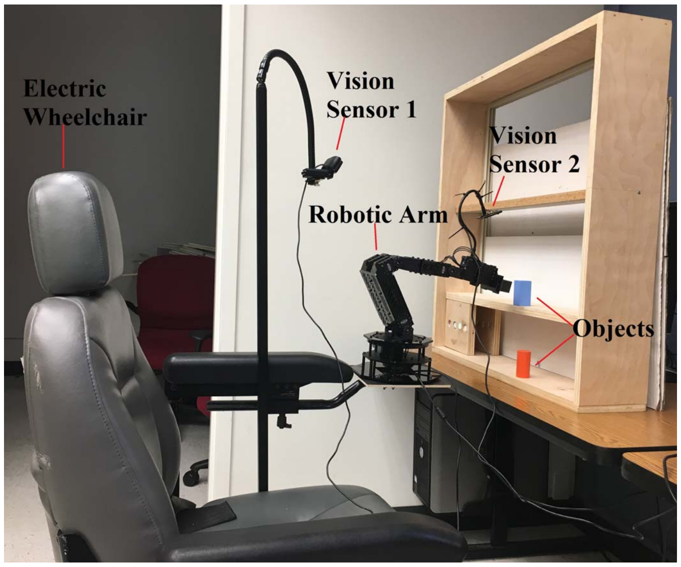

Figure 1. Image of the robotic arm mounted on an electric wheelchair

Taking these drawbacks into account, an autonomous system is designed that will help the electric wheelchair users to pick up objects from a table or a shelf effortlessly in a short amount of time. The estimated cost of the system is around $1500, which is ten times less than the commercially available WMRA [1].

The goal of the research is to develop a system that can deliver the desired object to the user with minimal to no supervision. This experiment is different from the existing ones in a way that the system is designed to be user independent and can figure out the position of the object on its own using an object detection algorithm and can complete the task of picking up the object and returning it to the user under one minute. The coarse location of the target object is calculated by using image obtained from the vision sensor located above the arm and fine localization is achieved by using a vision sensor near the gripper. The obtained coordinates are fed to the robotic arm using a serial communication interface to move the arm in the right direction.

II. SYSTEM DESCRIPTION

The proposed system consists of a robotic arm with six degrees of freedom, an electric wheelchair, a computer system and two vision sensors. Figure 1 shows the image of the complete system. The robotic arm is mounted on the left arm of the wheelchair. One of the vision sensor is placed above the robotic arm which captures the video of the object placed on the shelf. The position of this vision sensor is static and the relative distance between the sensor and the base of the arm is known prior to the start of the experiment to deliver accurate results. The shelf is used to achieve the goal of accessing objects at different heights. Another vision sensor is mounted above the gripper to fine tune the position of the arm's gripper before it can pick the object. This vision sensor is dynamic and the position varies depending on the position of the target object. A computer vision system is developed that detects multiple colored objects placed randomly on the surface within reach by the arm. Figure 2 shows the implementation of the proposed system. The different parts of the system are discussed in detail below.

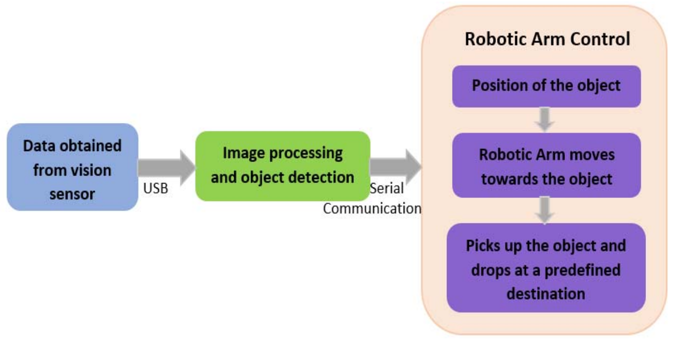
Data obtained from vision sensor
USB
Image processing and object detection
Serial Communication
Robotic Arm Control
Position of the object
Robotic Arm moves towards the object
Picks up the object and drops at a predefined destination

Figure 2. Block diagram of the system

A. *Robotic Arm*

A Trossen Robotics PhantomX Reactor Robot Arm is built using an Arduino compatible advanced microcontroller called Arbotix-M robocontroller [7]. The arm has eight AX-12A dynamixel actuators for controlling different parts of the arm. Each servo has sensors to track its speed, temperature, shaft position, voltage and load. The servo motor at the bottom is used to move the arm in the horizontal direction (left and right). The shoulder has dual servo which moves the arm in the forward and backward direction. The elbow also has dual servo which controls the up and down movement of the arm. The wrist angle and wrist rotate have one servo each. The arm has a parallel gripper controlled by an AX-12 servo which can hold and release the object. The arm is powered by a 12V 5amp power supply and is connected to a computer system via a FTDI cable. A serial connection is made between the computer system and the robotic arm in order to communicate with the robotic arm. This was achieved by using PySerial, a python serial port access library. The serial connection has a baud rate of 38400. The data packet which is 17-byte long was sent serially from the computer to the Arbotix-M robocontroller to control the movement of the arm. The 17-byte data packet includes the header (0xFF/255), X-axis coordinate, Y-axis coordinate, Z-axis coordinate, wrist angle, wrist rotate, gripper, delta byte and check sum. Each of the servo motor can be controlled by varying the 17-byte data sent to the arm accordingly. A short delay is introduced after every serial write command is performed to ensure the reception of the 17-byte data by the robotic arm.

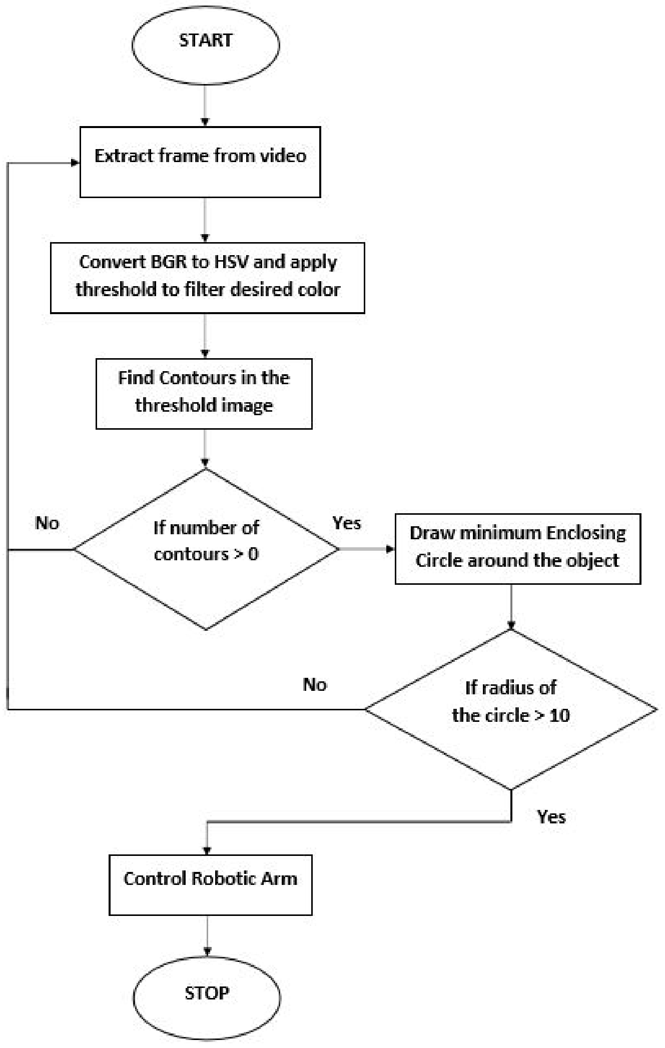
START
Extract frame from video
Convert BGR to HSV and apply threshold to filter desired color
Find Contours in the threshold image
No
If number of contours > 0
Yes
Draw minimum Enclosing Circle around the object
No
If radius of the circle > 10
Yes
Control Robotic Arm
STOP

Figure 3. Flowchart for Computer Vision Algorithm

B. Vision Sensors

The system uses two vision sensors (USB webcams) to perform object detection. A Logitech HD c920 webcam (vision sensor 1) is mounted above the robotic arm facing the shelf located in front of the arm. The vision sensor 1 captures the video of the arm and the shelf in real time. Frames extracted from this video are processed and the position (X, Y) of the target object is calculated. This data is used for coarse positioning of the robotic arm. A robot VGA webcam (vision sensor 2) is mounted above the gripper using a 200 mm gooseneck. The vision sensor 2 captures a close-up video of the target object. Frames from this camera are captured only after the arm is moved to the position indicated by vision sensor 1. The vision sensor 2 is used to position the gripper exactly in front of the object so that the object can be picked up correctly. The resolution of the images in both the camera is 640 x 480 pixels and the webcam captures around 30 frames per second. Figure 4 shows the snapshot of the image captured by the two vision sensors.

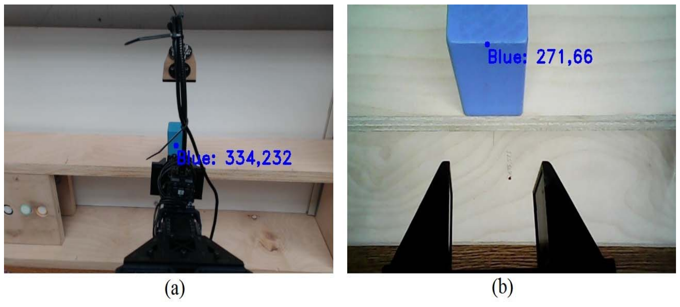

Figure 4. (a) Image captured by the vision sensor 1 (b) Image captured by the vision sensor 2

C. Computer Vision Algorithm

The robotic arm is programmed to move towards the position of a specific colored object. The color detection algorithm is written in Python using the OpenCV library. The vision sensor captures the real time video of the robotic arm and the object.

The video is extracted frame by frame and each of the frame is processed through a series of steps to detect colored objects in the frame. Each frame is converted from BGR (Blue Green Red) image to HSV (Hue, Saturation and Value) image. A specific color is filtered by applying a lower and upper limit threshold to the HSV image [8]. Different colors have different range of hue values. Hence by providing the appropriate range of hue values, different colors can be detected simultaneously. The experiment is performed with blue color chosen as the object of interest. The resulting image is eroded to remove the noise present in the image and dilated to fill the gaps in the image. The threshold image is scanned to check the presence of contours. In case a contour is located the position of the contour is found by calculating moments [8]. A minimum enclosing circle is drawn around the object and the center of the circle corresponds to the X and Y coordinates of the position of the object. The color detection algorithm is extended to capture data from two vision sensors and combined with the robotic arm control code to perform the desired task. Figure 3 shows the flowchart implementation of the computer vision algorithm.

III. EXPERIMENT AND RESULTS

The robotic arm is operated in the Cartesian mode. Initially the two vision sensors start capturing the video in real time. A frame is extracted from the video captured by the vision sensor 1 and the X and Y position of the object is calculated using the color detection algorithm. Before proceeding with the arm control, a condition is checked to see if the position of the object is same as or very close to its previous position ($X \pm 5$). If the condition is true, it means the object has not moved and hence the arm remains in the base position. If it is false, it indicates that the object has been moved to a different location and the arm is controlled to move towards the new location of the object. This way it eliminates the possibility of the arm to move to the same position repeatedly even if multiple frames are captured by the vision sensors. If an object is detected by the vision sensor, the arm is moved to the base position (an inverted L) and the gripper is opened. The arm is then moved up or down depending on the value of the Y coordinate with the threshold measured at 300.

Arm Link software provided by Trossen Robotics is a simple graphic user interface (GUI) to control the InterbotiX line of robot arms [9]. X coordinate range of the arm varies from 212 to 812 and can be controlled manually using the Arm Link software. The pixel distance in the horizontal axis varies between 0 and 640 and is measured using the color detection algorithm. The object is placed at different positions in front of the arm and multiple readings of the X distance in pixels and the corresponding X coordinate of the arm are taken and plotted in the Desmos graphing calculator. A best fit equation of a line is formulated. The factors 0.95 and 150 are calculated using trial and error method such that the line generated passes through a maximum number of points plotted.

$$\text{X position of the arm} = (\text{X object} * 0.95) + 150 \qquad (1)$$

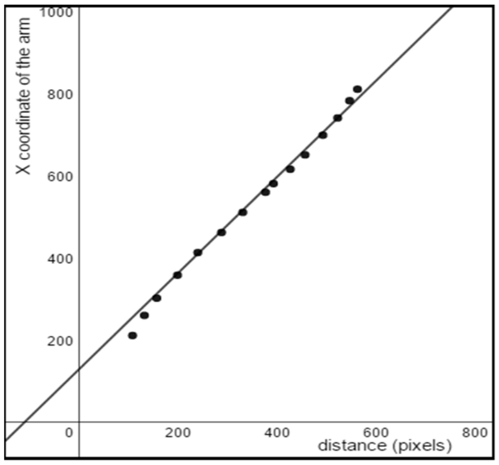

Figure 5. X distance in pixel vs X coordinate of the robotic arm

Figure 5 shows the graph between X distance in pixel and X coordinate of the robotic arm. The arm is then controlled to move in the left or right direction depending on the value obtained from the equation *(1)*. Now the robotic arm is moved to a position which is very close to the location of the target object. There is a possibility that the obtained location might not be the exact location of the object. Hence the vision sensor 2 is used to obtain the accurate location of the object. The X value in pixel is checked to see if it lies in the center of the frame. In the event it does not fall within the specified range, the arm is moved in small steps in the left or right direction until it reaches the center position. If the X value is already in the specified range, no command is given to the arm.

The object is picked up by closing the gripper and the arm then moves in front of the user to deliver the object. Figure 6 shows the flowchart implementation of robotic arm control. Table I and II shows the X and Y position of the target object obtained from the vision sensor 1 and 2. The time taken for the arm to pick up the object and deliver it to the user is noted. It also specifies if the attempt is a success or a fail. Out of the 24 trials performed with the objects in the upper and lower level, 20 were successful in picking up the objects correctly. This gives an accuracy of about 83.33%.

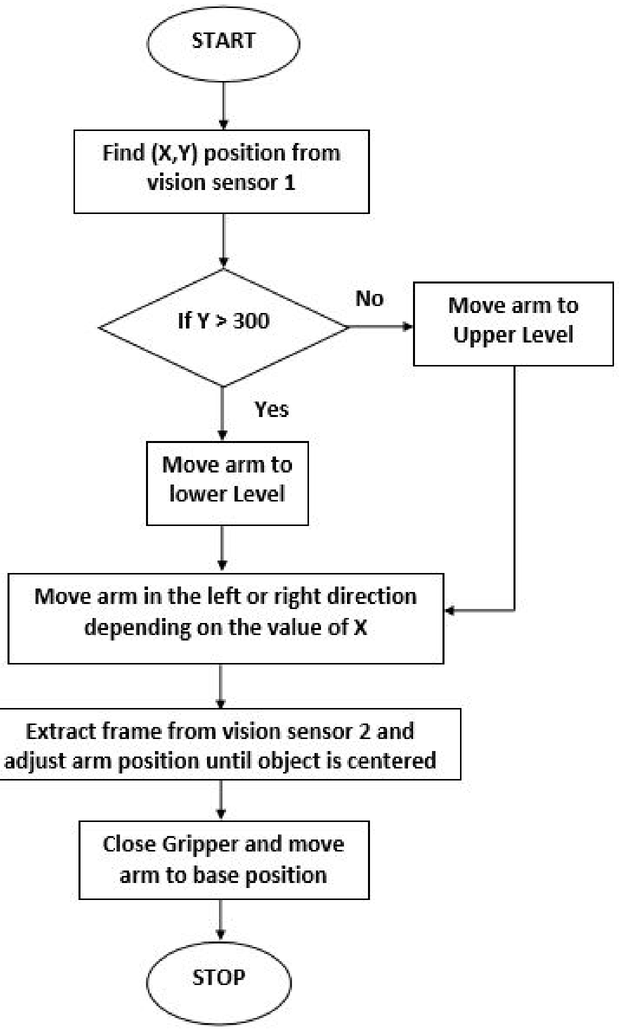
START
Find (X,Y) position from vision sensor 1
If Y > 300
No
Move arm to Upper Level
Yes
Move arm to lower Level
Move arm in the left or right direction depending on the value of X
Extract frame from vision sensor 2 and adjust arm position until object is centered
Close Gripper and move arm to base position
STOP

Figure 6. Flowchart for Robotic Arm Control

TABLE I

Upper level			
Vision Sensor 1 X, Y (in pixels)	**Vision sensor 2 X, Y (in pixels)**	**Task completion time (in seconds)**	**Attempt Status (Success/Fail)**
259, 219	69, 174	39.51	Success
296, 232	121, 158	37.64	Success
326, 222	324, 82	37.80	Success
340, 224	280, 105	36.58	Success
356, 256	143, 139	35.79	Success
391, 234	496, 136	38.07	Success
411, 230	590, 166	37.65	Success
435, 238	579, 163	37.24	Success
615, 244	590, 166	37.65	Fail
630, 229	547, 114	35.28	Success
302, 249	261, 111	37.53	Success
248, 242	63, 145	37.66	Success

TABLE II

Lower Level			
Vision sensor 1 X, Y (in pixels)	**Vision sensor 2 X, Y (in pixels)**	**Time completion time (in seconds)**	**Attempt Status (Success/Fail)**
266, 361	110, 111	37.59	Success
277, 374	87, 138	36.99	Success
338, 372	343, 63	39.20	Success
383, 352	561, 85	37.35	Fail
254, 367	74, 124	37.65	Success
400, 367	537, 120	37.58	Success
416, 363	609, 123	37.42	Fail
390, 360	568, 130	37.77	Success
407, 368	593, 105	37.53	Fail
249, 365	65, 122	37.79	Success
288,367	143, 150	37.36	Success
414, 371	577, 142	37.93	Success

Most of the unsuccessful attempts occur when the target object is placed farther from the robotic arm. From figure 5 we can see that the accuracy is lesser for x values lesser than 200 or greater than 570. Hence the equation *(1)* does not hold good for extreme values. Effective solutions for reducing the number of unsuccessful attempts would be formulating a higher order equation that can hold good for most locations of the object and more fine tuning by repetitively moving the arm until the object is in the center of the frame.

From the above tabulations, it can be found that the time taken to pick up objects from the upper and lower level are almost the same value. The task of picking up an object and delivering it to the user can be completed at an average of 37.52 seconds (37.37 seconds for the upper level and 37.68 for the lower level). This is significantly low compared to the completion times calculated when using brain computer interface [4] or gesture based recognition [5]. It can also be seen that attempts are more successful when the objects are placed at the upper level compared to the lower level. The reason for unsuccessful attempts on the lower level is that the light falling on the lower level is less due to the shadow cast by the shelf. This may pose as a problem for color based object detection since the position of the detected object keeps fluctuating due to insufficient light.

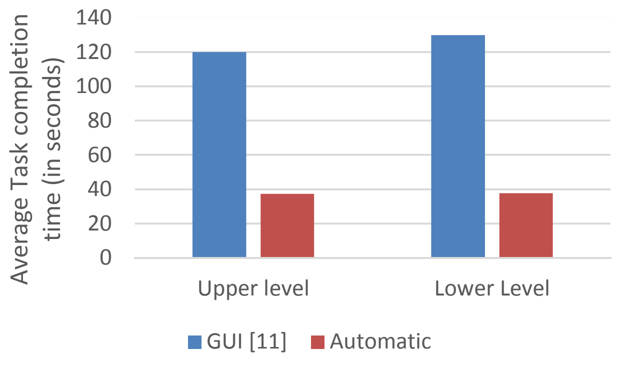

Figure 7. Comparison of task completion times using different control modes

When the arm is controlled using the Arm Link software, it was seen that the completion time for the same task took around 120 to 130 seconds for a skilled user. Figure 7 shows the comparison of the task completion times of picking up the object using GUI and the automated system. The result shows that automating the movement of a robot arm can save a lot of time and it does not require the user to concentrate while performing the task.

IV. CONCLUSION

A computer vision algorithm based on color detection has been developed to automate the movement of the robotic arm. This was implemented by using two vision sensors to accurately find the location of the target object. First sensor was used for obtaining the coarse location of the object and the second one was used for fine localization. The robotic arm has the ability to pick up objects placed at different locations on the upper and lower level with a success rate of 83.33%. The main goal of performing the action of picking up the object under one minute is achieved.

V. FUTURE WORK

Color based object detection was used to determine the position of the desired object. Practical applications will involve objects of various shapes and multiple colors. A user interface can be developed that can allow the user to select a desired object from the frame captured by the vision sensor by incorporating speech recognition. Once the object selection is made, the coordinates of the object can be fed to the arm. Depth sensors can be used in addition to vision sensors to increase the performance of the robotic arm control thereby minimizing the number of unsuccessful attempts. Advanced vision algorithms can be used for pose estimation and size detection to detect same object at different angles. Incorporating the techniques mentioned above will help with detection of daily living objects and assist wheelchair users to perform tasks in a short period of time.

REFERENCES

[1] "Kinova Robotics Jaco Arm". [Online]. Available at: http://www.kinovarobotics.com/service-robotics/products/robot-arms/

[2] "Assistive Innovations Intelligent Robotic Manipulator". [Online]. Available at: http://www.assistive-innovations.com/en/robotic-arms

[3] Camilo Perez Quintero, Oscar Ramirez, Martin Ja¨gersand "VIBI: Assistive Vision-Based Interface for Robot Manipulation", 2015 IEEE International Conference on Robotics and Automation (ICRA)

[4] Katherine Tsui and Holly Yanco, "Development and Evaluation of a Flexible Interface for a Wheelchair Mounted Robotic Arm", Human-Robot Interaction (HRI), 2008 3rd ACM/IEEE International Conference

[5] Indika Pathirage, Karan Khokar, Elijah Klay, Redwan Alqasemi, "A Vision based P300 Brain Computer Interface for Grasping using a Wheelchair-Mounted Robotic Arm", 2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)

[6] Hairong Jiang, Juan P. Wachs, Bradley S. Duerstock, "Integrated Vision-Based Robotic Arm Interface for Operators with Upper Limb Mobility Impairments", 2013 IEEE International Conference on Rehabilitation Robotics, Seattle, Washington USA

[7] "Robotic Arms, Trossen Robotics" ". [Online]. Available at: http://www.trossenrobotics.com/

[8] Adrian Rosebrock, "Ball tracking with opencv", Sp 14, 2015 in Object Detection, Tutorials http://www.pyimagesearch.com/

[9] "Interbotix Arm Link Software, Trossen Robotics". [Online]. Available at: http://learn.trossenrobotics.com/36-demo-code/137-interbotix-arm-link-software.html

ACKNOWLEDGEMENT

This work was supported in part by the National Science Foundation under grant no. ECCS-1150507.