Vision Based Object Classification with Scorbot-ER 4U

*Md. Hazrat Ali and N. Mir-Nasiri

Abstract— A Robot is a mechanical device that can perform preprogrammed physical tasks. A robot may act under the direct control of a human (eg. the robotic arm) or autonomously under the control of a pre-programmed computer. One of the most frequent tasks done by the robotic arm is to pick and place. The objective of this research is to come up with a solution to model and simulate the Intelitek Scorbot ER-4 Plus using the RoboWorks and RoboTalk simulation software with the integrated vision system. Scorbot ER-4 does not have integrated camera. This research focusses on adding a camera with the robot and performs tasks such as object detection, classification, pick and place. For instance, it captures image of the 3 test blocks, identify the block with the largest area and find its centroid coordinate in order to calculate the biggest object. In addition, it is also shown that the robot is able to successfully communicate through ACL controller, RoboWorks and RoboTalk software in order to perform other more advance tasks .

Keywords---Robot; pick and place; image processing; detection, classification

I. INTRODUCTION

SCORBOT-ER 4 plus is manufactured by Intelitek. It is designed for laboratory and training applications with fast and accurate five-axis vertically articulated robot. The robotic system is controlled by ACL Controller-A, enables continuous path control, linear and circular interpolation, path optimization, different control profiles, event driven programming and full interrupt capabilities. The controller's 300 parameters are user accessible and allow definitions for servo control, speed, velocity profile, smoothing, position error, thermal, impact and limit protection, homing, encoder interface as well as Cartesian calculations. SCORBASE pro for Windows enables intuitive robotic programming and operation. Optional RoboCell 3D graphic simulation software helps users create, program and execute an unlimited number of simulation based industrial applications. This Scorbot is able to monitor and control up to 11 axes, 16 inputs and 16 outputs and is an ideal robot for FMS and basic CIM applications. With its broad range of accessories and software, this system provides a powerful yet safe didactic environment. Even though the robot has very attractive features, it still lacks of vision system as many other industrial robots. To make the robot work in real time and offline mode, it is crucial to have integrated camera so that it works more intelligently and optimize the grippers travelling path as well as travelling time.

*Md. Hazrat Ali, School of Engineering, Nazarbayev University, Astana, Kazakhstan *Email: md.ali@nu.edu.kz

N. Mir-Nasiri, School of Engineering, Nazarbayev University, Astana, Kazakhstan

II. LITERATURE REVIEW

One of the researches presented the development of a controller for a Scorbot Er-4u didactic manipulator, using a vision system communicating via the USB port. Considerate that nowadays, robotics is an essential element for the automatization in manufacturing process [1]. The Scorbot-Er 4u robot is a versatile and reliable system for educational use, it has a mechanical structure, vertically articulated, open frame with five rotational axes and gripper [2]. Several projects have been accomplished that involve the control of the Scorbot by means of the use of Matlab Software or incorporating a system of vision. A project called cut laser [3] where the authors describe that the robot will work like a printing machine where the end effector will have a laser and it will draw or cut metal with the power of laser beam. The game named tic tac toc where the robot by means of a system of vision can select someone of the recorded poses to follow the game [4]. The selection of nuts, screw and keys are used in another project. The system of dynamic guide for Scorbot-Er IX robot arm by means of artificial vision is applied in [5], it also involves in the inverse kinematics for obtain the desired displacement.

In [6], the kinematics problem is defined as the transformation from the Cartesian space to the joint space and vice-versa. The solution is through model of representation Denavit-Harbenterg. The workspace density function is described with both planar revolute and variable-geometry-truss manipulators [7]. A focus reaching subtask which involves computing trajectory for an arm manipulator, integrated approach to inverse kinematic and path planning [8]. Self-calibration [9], [10] of any system is the capability of performing calibration without any external expensive calibration apparatus setup. These are useful when a system is functioning in a dynamic environment. Mainly, self-calibration techniques can be classified into two broad categories: redundant sensor approach and motion constraint approach. However, both these approaches are having certain advantages and limitations [11]. The main drawback of sensor based approaches is that some of the kinematic parameters are not independent of the error models, and therefore, position and/or orientation of the tool on the platform cannot be calibrated. In the later category, it is not possible to calibrate the position and/or orientation of the tool resulting that the error in locked passive joints may become unobservable. In this research, we proposed a vision based object classification model based on attached camera with its end effector.

III. WORKING PRINCIPLES

A. MODELING AND ANALYSIS

i) Forward Kinematics

The essential concept of forward kinematics is that the positions of particular parts of the model at a specified time are calculated from the position and orientation of the object, together with any information on the joints of an articulated model. So for example if the object to be animated is an arm with the shoulder remaining at a fixed location, the location of the tip of the thumb would be calculated from the angles of the shoulder, elbow, wrist, thumb and knuckle joints. Three of these joints (the shoulder, wrist and the base of the thumb) have more than one degree of freedom, all of which must be taken into account. If the model were an entire human figure, then the location of the shoulder would also have to be calculated from other properties of the model. **Table I** shows the calculated D-H parameter with integrated camera system.

TABLE I
D-H PARAMETER CALCULATION

Link	Angle θ	Displacement d (mm)	Length l (mm)	Twist α
1	θ_1	343	0	90º
2	θ_2	0	220	0º
3	θ_3	0	220	0º
4	θ_4	0	0	-90º
5	θ_5	150	0	0º

ii) Inverse Kinematics

Inverse kinematics is the process of determining the parameters of a joint flexible object in order to achieve a desire pose. For example, with a 3D model of a human body, what are the required wrist and elbow angles to move the hand from a resting position to a waving position? This question is vital in robotics, where manipulator arms are commanded in terms of joint angles. Inverse kinematics are also relevant to game programming and 3D modeling, though its importance there has decreased with the rise of use of large libraries of motion capture data. We have calculated θ_1,θ_2 and θ_3 for the robot using mathematical formula.

iii) Functional Flow chart

Figure 1 shows working flow chart of the robot based on integrated camera inputs. Camera input

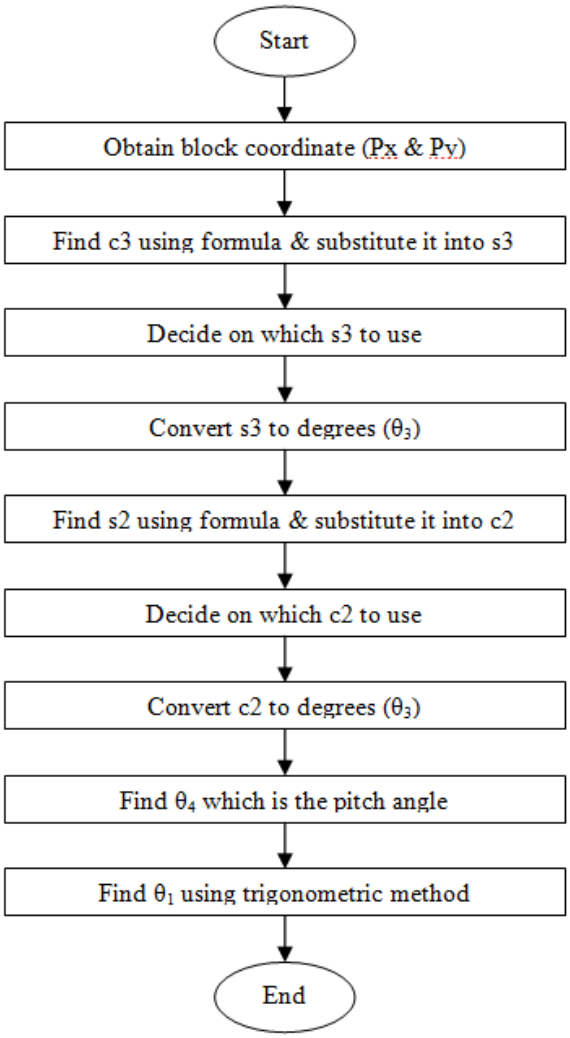

Fig. 1. Working flow chart

B. IMAGE PROCESSING

Figure 3 shows the image processing flow chart. The function in this file performs image capturing and the image processing steps to determine all the image properties. Area of the captured image is calculated and determined the centroid of the object with the biggest area to classify the biggest object.

Step 1: Image Capturing

First, the image is captured and stored in an array format in 640x480 pixels resolution. Then, the RGB image is converted into grayscale image. The reference image is shown in **Fig. 3**.

Step 2: Thresholding

The grayscale image is shown in **Fig. 4**. Image conversion takes place from grayscale to binary by applying threshold method which is done on the grayscale image in order to mask the shadow around the block. This helps in reorganization on the edge of the block in an image. The code *IM2BW* produces binary images from indexed, intensity, or *RGB* images. To do this, it converts the input image to

grayscale format (if it is not already an intensity image), and then converts this grayscale image to binary by threshold method. The pixels are converted to black color (1) if the output binary image BW has values less than 0.42 whereas all other pixels is converted into white (0) in color if the output binary image BW has values more than 0.42. We must convert the connected components in binary image. The output image will only contain 0 (White) or 1(Black).

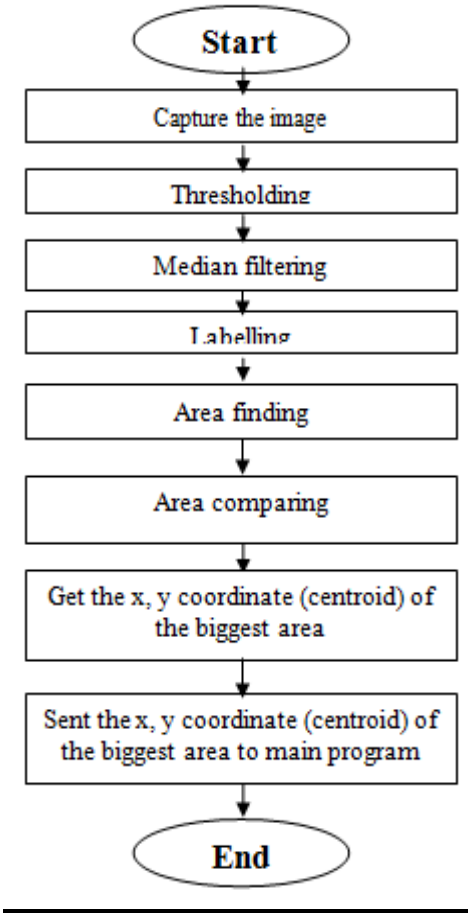

Fig. 2. Image processing flow chart

BWLABEL supports 2-D inputs only. **Figure 5** shows the resulting image after thresholding process. **Equation 1** is used to calculate threshold values.

$$h'_f(z) = \frac{1}{2K+1} + \sum_{j=-K}^{K} h_f\ (z+j) \qquad (1)$$

Where, *K* is a constant representing the size of the neighborhood used for smoothing and its value =128, z refers to the count 0 to 255, j applies to the pixel values, and h_f refers to the total histogram values for the image function where Mean = 0.40 and Standard Deviation =10.12. This value white pixel count is used as motion level. This signifies the magnitude of the motion. Greater magnitude of motion produces more white pixels thus giving higher motion level.

Step 3: Median Filtering

Figure 6 highlights the image after applying median filter. It is necessary to apply 2-D median filtering after converting the images to black and white. The code *B = MEDFILT2* (A, [MN]) performs median filtering of the matrix A into two dimensions. Each output pixel contains the median value in the M-by-N neighborhood around the corresponding pixel in the input image. Code MEDFILT2 pads the image with zeros on the edges, so the median values for the points within [M N]/2 of the edges sometimes appear as distorted.

Step 4: Labelling

It is necessary to perform labelling which will enhance image to recognize the blocks and to find the total number of blocks in the image.

Step 5: Area Calculation and Centroid Finding

The code *REGIONPROPS* plays the most important role in finding the area and centroid. *REGIONPROPS* measure properties of image regions. IMAGE = regionprops (L, PROPERTIES) measures a set of properties for each labeled region in the label matrix L. After the area of each object has been known, it compares the area to get the biggest area then sent the x, y coordinate (centroid) of the biggest area to the main program. **Figure 7** shows the resulted biggest object.

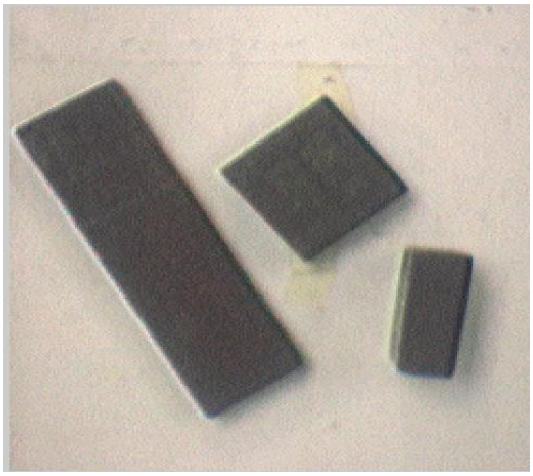

Fig. 3. Capture the image

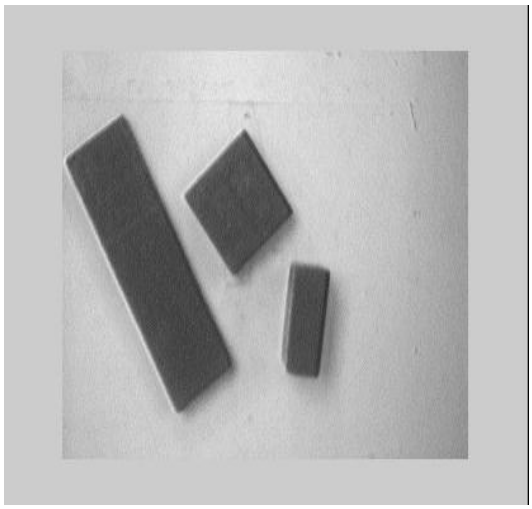

Fig. 4. Image in grayscale

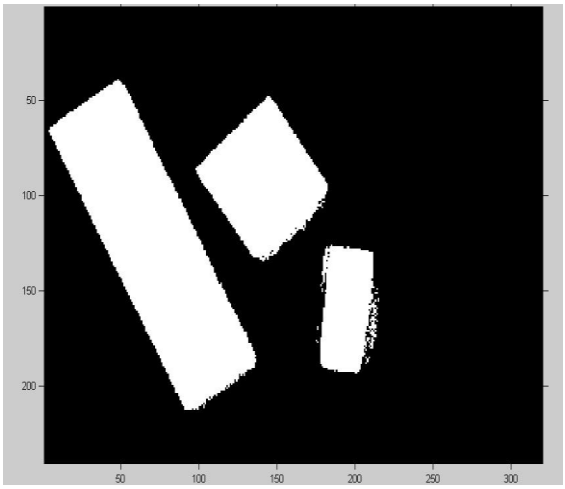

Fig. 5. Image after thresholding

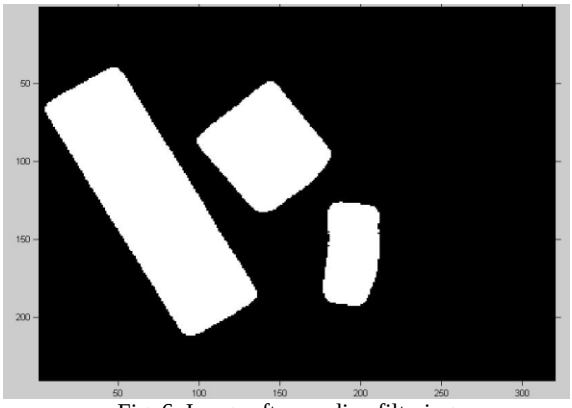

Fig. 6. Image after median filtering

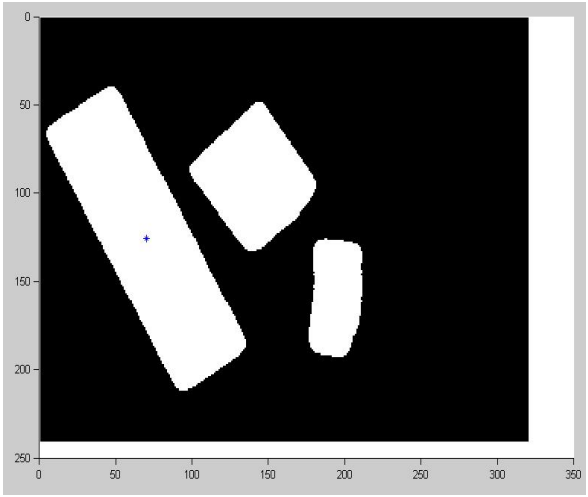

Fig. 7. Image with x, y coordinate (centroid) of the biggest area

C. MAPPING AND INTERFACING

i) Mapping

Once the biggest centroid has been found, the X and Y coordinates of object will be passed to the interface program. In the Interface program, the conversion of the X and Y coordinate into millimeter scale is accomplished. The parallel port is able to send with value from 0-128. The relationship between the values within the millimeter scale needs to establish so that it will fit the length value of X and Y into the range of the parallel port in term of the maximum value of 128. Another conversion is in between the length of X and Y (in millimeter) into the range of 128 which is the maximum data that can be sent through the parallel port. **Figure 8**, shows the path planning of the end effector of the robot.

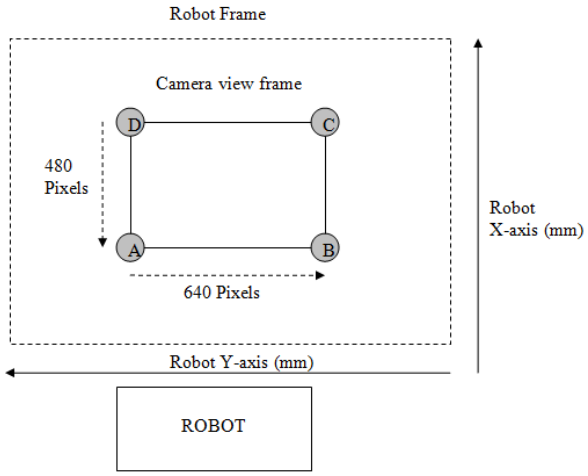

Fig. 8. End effector's path planning

The end effector's path planning sequences are:

a) Robot is controlled to move its tool center point to the four corners of the camera frame – A, B, C, and D. (A = 215x,108.6y) (B = 227x, -167y) (C = 431x, -164y) (D = 430x, 100.8y)

b) The X and Y coordinates of the robot at point A, B, C and D will be recorded down as A : $\{Y_1,X_1\}$, B: $\{Y_2,X_1\}$, C: $\{Y_2,X_2\}$, D: $\{Y_1,X_2\}$.

c) The range of Y-axis (mm) $Y_2 - Y_1 = 640$ pixels. The range of X-axis (mm) $X_2 - X_1 = 480$ pixels.

Conversion for X axis = (272/640)

=0.425 mm/pixels

Conversion for Y axis= (205/480)

=0.427 mm/pixels

ii) ii) ACL Interface

In order to send the values of X and Y coordinate to ACL in 7 bits (128 in decimal), the coordinates are compressed into 128. XX and YY is the X and Y coordinates sent to the ACL controller. **Figure 9**, explains image compression.

XX = (128/272) x (centroid of X in milimeter)
YY = (128/205) x (centroid of Y in milimeter)

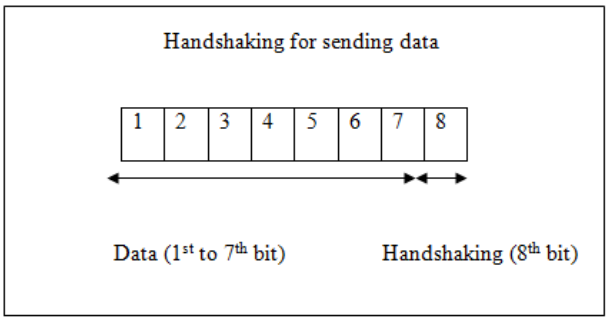

Fig. 9. Image compression

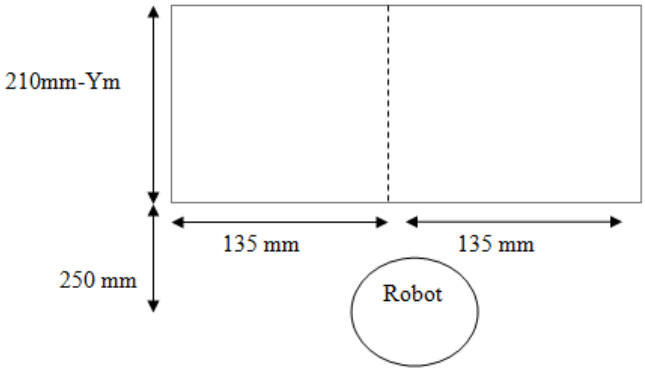

Fig. 10. Image after thresholding

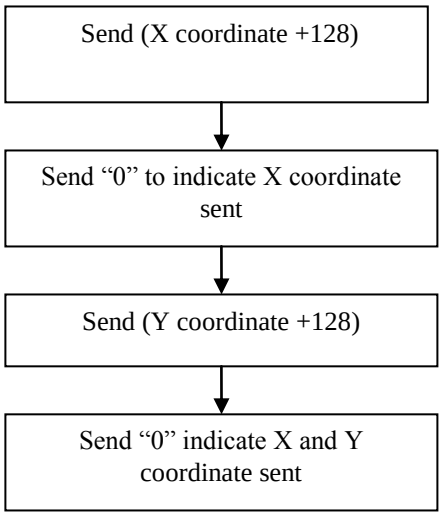

Fig. 11. Image conversion sequences

Bit 8 is used as the handshaking in order to activate the data needs to be added with 128. The X coordinate will send compressed value (1st to 7th bit) in 128 followed by adding 128 to indicate that it is handshaking. After that, sending signal '0' to indicate X coordinate has been sent. The same process is carried out for sending Y coordinate. **Figures 10** and **11** highlight image conversions sequences.

IV. Simulation and Development

A. RoboTalk-RoboTalks Simulation

RoboTalk is used in order to interact with RoboWorks. It is the most powerful mechanism for real-time interactive control of a RoboWorks model. RoboTalk uses the TCP/IP network protocol to allow an external program to make a connection with RoboWorks to control the model. The tag names that are defined by the user in their model are used by RoboTalk. RoboTalk comes in a package as *DLL* file to use on the Win32 platform. Also, LabVIEW is available with RoboWorks to support interfacing with RoboWorks. RoboTalk source is running on a platform other than Win32. This source can be compiled on platforms such as Linux for real-time communication with RoboWorks. RoboTalk can be used as a means of interfacing with RoboWorks to interactively control the simulation from another program. External programs that can interactively control a RoboWorks model include 'c' programs, Visual Basic, Matlab, LabVIEW, etc. Matlab is chosen for easy interaction with other written programs. The robot has six degrees of freedom including the end gripper. **Table II** describes the joint and tag assignment.

Table II
Image After Thresholding

Joint No:	Assigned Tag Name
1	R1
2	R2
3	R3
4	R4
5	R5
6	R6

The tag names are assigned earlier in RoboWorks modeling. Matlab uses these tag names to control each joint.

B. Design of RoboTalk

After analyzing the given tasks, RoboTalk program is written based on the two tasks defined earlier. All joints (Tag values) are controlled by RoboTalk. A certain sufficient steps will be used to run the "For" loop, such as 100 times. During each time, the tag value will increment or decrement accordingly until it reaches the target coordinate. Later, the gripper will be closed to pick up the target object. To complete the task, RoboTalk is designed in such a way so that the picked object is moved to the final destination before releasing it.

C. Tasks of RoboTalk

Figure 12 shows the main tasks of the robot. Generally, RoboTalk is designed to handle 2 tasks:

i) Read the coordinate values and pick up the object.
ii) Place the object to the target position. (On the left of the robot arm).

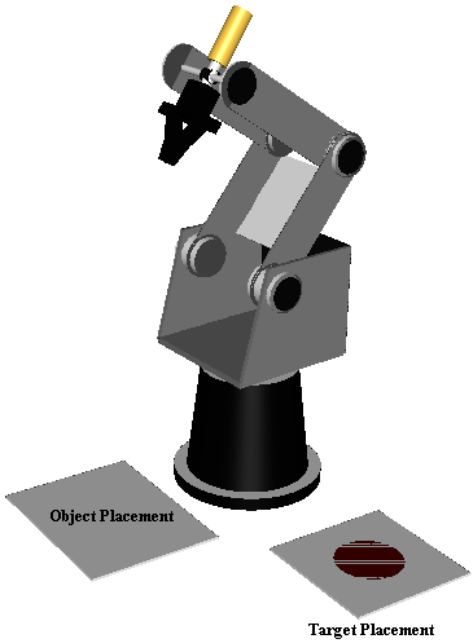

Fig. 12. Robot's tasks

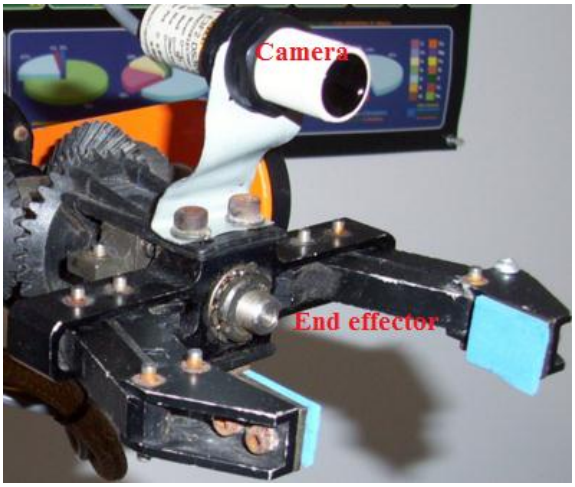

Fig. 13. Camera integrated to the end effector

Figure 13 shows an example of the end effector of Scorbot with the integrated camera [14].

D. System Integration

The main program integrates every function. This starts with image processing function to obtain the x and y centroid coordinates of the biggest object in unit pixel. The x and y centroid coordinate is then sent to ACL interface function as well as RoboWorks function and convert from pixel into millimeter. Each of this function is designed with different mapping method so that different value of x and y coordinate will be sent out. The function named 'inverse kinematics' obtained the x and y coordinate from RoboWorks function and calculates the joint angles for RoboWorks modeling. The ACL interface first sends the x and y coordinate value to the parallel port and next to the ACL (Advanced Control Language) controller. **Figure 14** and **15** explain control mechanisms of the robot end effector through computer and ACL controller.

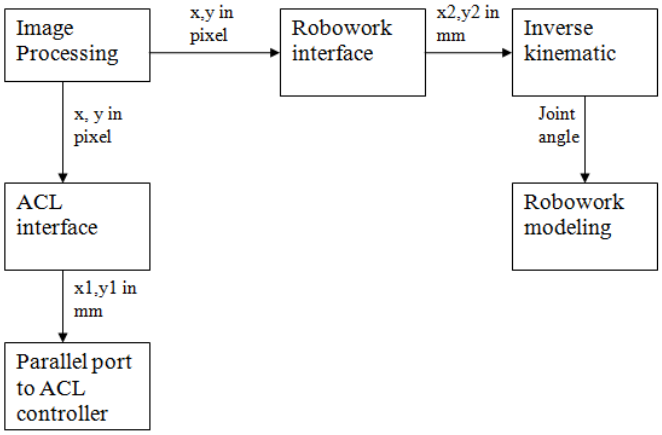

Fig. 14. Program interface

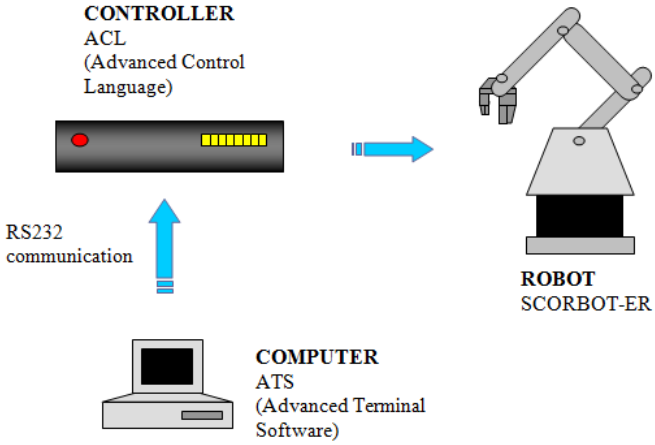

Fig. 15. Robot control through ACL

V. Conclusions

It is noted that the image processing function is able to capture the images of the 3 blocks and able to identify the target block with the largest area. In addition, the direct and inverse kinematics of the robot model is done in order to simulate the RoboWorks using RoboTalk. The computer is able to send the data to ACL controller through computer parallel Port. By integrating higher resolution camera, it is possible to perform other task such as color sorting, etc. Apart from that, sensor can be added to detect height or Z-axis position so that it is possible to send the coordinate data of the object in 3 dimensional work spaces.

References

[1] Alejandra C. B and Gilberto M. A, " Vision System via USB for Object Recognition and Manipulation with Scorbot-ER 4U. International Journal of Computer Applications (0975 – 8887), Volume 56– No.18, October 2012.

[2] Intelitek ®, "Scorbot-Er 4u, User Manual", Copyright ©2001 Intelitek ® Inc.Catalog #100343 Rev.B. September 2001.

[3] Galnares, J., http://www.prototipando.es/proyectos/73-cortadora-laser scorbot?showall=&start=2.

[4] Almanza, O. D., "Implementación de la estrategia de juego Tic-Tac Toe para la interacción con un brazo robótico", Avances en Inteligencia Artificial, ISSN: 1870-4069, IPN.

[5] Soto, M. C.E, "sistema de guía dinámico para brazo robot Scorbot Er-IX mediante vision artificial", http://cybertesis.ubiobio.cl/tesis/2006/soto_c/html/inde-frames.html.

[6] Verma, A., Vivek, A.D., "End-effector Position Analysis of SCORBOT-ER V plus Robot", International Journal of Smart Home, Vol.5,No.1, January 2011.

[7] Suthakon, J., Chirikjan G.S., "A new inverse kinematics algorithm for binarymanipulator with a many actuators", Advance Robotics, Vol. 15, No. 2. Pp.225-244 (2001).

[8] Bertram, D., Kuffner, J., Dillmann. R., and Asfour, T., "A integrated approach to kinematic inverse and path planning for redundant manipulators", Proceedings of the 2006 International Conference in Robotics and Automation, Orlando, Florida, (2006), pp. 1874-1878.

[9] Hartley, R.I.: Self-calibration from Multiple Views with a RotatingCamera. In: 3rd European Conference on Computer Vision, Stockl-holm, pp. 471-478 (1994)

[10] Quan, L.: Self-calibration of an Affine Camera from Multiple Views. International Journal of Computer Vision 19(1), 95-105 (1996)

[11] Vincze M., Prenninger J.P., Gander H.: A laser tracking system tomeasure position and orientation of robot end effectors under motion. Int J Robotics Res 13(4), 305-14 (1994).

[12] http://www.digital-circuitry.com/MyLAB_Scorbots.htm