Robotic Arm Control Algorithm Based on Stereo Vision Using RoboRealm Vision

Roland SZABÓ, Aurel GONTEAN
Applied Electronics Department, Faculty of Electronics and Telecommunications
Politehnica University of Timişoara, 300223, Romania
roland.szabo@etc.upt.ro

***Abstract*—The goal of this paper is to present a stereo computer vision algorithm intended to control a robotic arm. Specific points on the robot joints are marked and recognized in the software. Using a dedicated set of mathematic equations, the movement of the robot is continuously computed and monitored with webcams. Positioning error is finally analyzed.**

***Index Terms*—decision making, image color analysis, machine vision, manipulators, stereo vision, video equipment.**

I. INTRODUCTION

The design and control a robotic arm is not an easy task as it is difficult for the robotic arm to follow the assigned geometry path in high precision and accuracy manner [1].

Most humans or animals move and walk easily without explicitly controlling their movements [2]. The robots need to use fuzzy logic in order to make more human-like movements. Artificial neural networks have been traditionally employed to learn and compute the inverse kinematics of a robotic arm [3].

The robotic arm can have many attributions, but are used mainly in the industry or in special applications like serving a glass of water to an immobile patient, using a remote pressing operation [4-7].

This paper presents the robotic arm control with stereo cameras. In the industry most of the robots have no vision system, they just move following predefined paths, which they have learned previously, but no decision is made by them, one can say that almost no artificial intelligence is implemented in their control software [8-11]. The industry needs robots with one, two or even more cameras, thus minimizing the work of the operator almost to zero.

The calibration of the arm is very important and time consuming, because the production line needs to be stopped to do it, and it needs to be done quite often to increase or maintain precision [9]. With the computer vision method, the calibration of the robotic arm can be performed continuously, because the robotic arm will always check its position with the cameras and will make small adjustment to reach the exact point which is needed.

II. RELATED WORKS

There have been quite many attempts to give sight to the robotic arms. Some of them are presented below.

One of the tests for a robotic arm was to apply a coating of plasma for certain objects. The goal was to reduce the time and cost of learning for a robotic arm. With vision system, a robotic arm with six degrees of freedom can be manipulated [12]. The system is similar to the one presented in this paper, the goal is to create a system which moves the robotic arm without teaching it all the points in its path. It is also desired to avoid calibration of the robotic arm; this should be made automatically while the robotic arm executes its necessary movements to manipulate the targeted object.

Another way to add vision system to a robotic arm is to mount the camera exactly on the end effector of the robotic arm, this method is also called camera in hand. This can be advantageous when the camera is filming objects which are approached by the camera. The system does not need to calculate the Jacobian matrix or to compute direct or inverse kinematics. All you need is a camera mounted on the robot's end effector, which can estimate distances [13]. The method seems interesting, but it could be considered rather a complementary method to this presented in this paper. When there are fixed cameras mounted next to the robot system, this can have an influence of both the robotic arm and the object which the arm wishes to manipulate.

Another approach is using a robotic arm with multiple fingers, with a video camera on its end effector which is filming the surrounding world. The implementation is quite interesting because there is a parallel system that reconstructs the object in 3D from the captured images. The system computes how robotic arm can grasp the object [14]. The system requires quite high computing power and the porting of such a system on a dedicated platform would be quite a big challenge. On the other hand the speed and integration area of dedicated processors and FPGAs are growing every year, so placing a complex system on a dedicated platform is only a matter of time. However the belief is that this system is more complex even than direct and inverse kinematics. The solution presented in this paper is much simpler.

A similar method to those presented before is a system with a handheld camera. The camera is mounted on the end effector of the robotic arm, but it also takes into consideration the dynamics of the camcorder. The system continuously makes a big number of iterations and calculates all the possible trajectories and places them into a tree structure. After doing some computations, it selects the optimal path for movement [15]. The system is quite interesting, but it needs high resources to operate, because much iteration is needed. The placement of the camera to the robotic arm's end effector gives much narrower vision range than many fixed cameras placed in the room where the robotic arm is.

Digital Object Identifier 10.4316/AECE.2015.02009

Another actuators control system has a robotic arm with six degrees of freedom and the system has, of course, video cameras. The system has a three-point spherical projection of Cartesian distances. The second part consists in representing the angle-axis rotation matrix from two measured points from the image [16]. The system is very interesting and is close to what will be presented in this paper, but mathematics used is more complex than the one used in this paper.

III. Problem Formulation

In order to accomplish the objectives of this study, the authors started of a small scale robot available in the lab: the Lynxmotion AL5 type robotic arm, which ships with almost no control software.

The task was clear, to create a system which can control the robotic arm, move it in any physically reachable place just with color recognition algorithms. The robot calculates and decides the movement direction and distance. The robotic arm starts to move only when the targeted object is in its field of view. The robotic arm tries to move the gripper as close as possible to the targeted object, after it tries to grab the object and bring it in the desired position. The robotic arm has colored bottle stoppers placed on joints in order to be visible for the cameras which use a color recognition algorithm. The base joint has a blue colored bottle stopper, the elbow has a yellow colored bottle stopper, the gripper has a red colored bottle stopper and the targeted object has a green colored bottle stopper. When the targeted object (the green bottle stopper) is moved, the robotic arm follows it, grabs it and brings it in a desired position. The robotic arm can be used for object sorting, gathering objects or other object manipulation tasks. Basically the mechanism needs to make multiple decisions and computations. The robotic arm needs to decide to reach or not the object and needs to compute the distance and the direction of the targeted object.

After the initial implementation on the small robotic arm, the algorithm was validated on an industrial robotic arm (SCORBOT-ER III). Because of the nature of the algorithm (video recognition of the robotic arm), it can work on robotic arms of every size and can be extended on robots with more joints. To extend the method on more joints is to repeat the algorithm on every joint of the robotic arm.

IV. Material and Method

A. Theoretical Background

The idea was to recognize each joint of the robot using just computer vision. In order to do so, a well-known method (mostly used on athletes when it's needed to introduce the data of their movement in the PC) was used. The luminous spots placed at subject's joints are united with lines to create a computer skeleton of the movement. This method is also used when the computer game creators want to make human-like movements to their characters and they record the movement data of actors. The next step is to assign the data to the character which is moving in the computer game [17-20].

In our experiments colored bottle stoppers were attached to each robotic arm joint. By acquiring the image, performing color recognition and finally uniting the recognized spots with straight lines the skeleton and the movement in real time of the robotic arm was obtained [21-25].

Afterwards, at the gripper level, a 2D coordinate system (similar to a parallelogram) was drawn by the software. This way the movement on 0Z and 0Y axes could be calculated. The length of each side of the parallelogram is one side of an orthogonal triangle and the segment connecting the colored spots is the other side. The arctangent of the movement angle can be computed for the specific motor which needs to move. For the movement around the base (0X axis), stereo triangulation had to be used to compute the distances between the cameras and the gripper and between the cameras and the target point (the green bottle stopper in this case). The difference between these two distances is the distance between the gripper and the target point. The robotic arm could be moved this way on three axes in 3D space.

In order to link the robotic with the angles, (1) is used. These values were sent as SCPI commands (Standard Commands for Programmable Instruments) to the robotic arm on the RS-232 interface.

$$robotic\_constant = \frac{\Delta\omega}{180° - 0°} = \frac{2500 - 500}{180° - 0°} = \\ = 11.(1)\_robotic\_values \quad (1)$$

In (1), 2500 is the maximum and 500 is the minimum robotic value and they correspond to a 180º maximum angle, leading to the result presented in (2).

$$1° \cong 11.(1)\_robotic\_values \quad (2)$$

Fig. 1 introduces the block diagram of the setup.

The image recognition algorithm is presented next. An RGB software filter is involved for each color, using a blob size threshold to get rid of the noise and the false positives. The center of gravity was computed next to determine the center of the colored spot to finally get the coordinate where the joint number will be drawn (this is also the start point of the segment which unites the joint with other joints). After some specific mathematical computations, the final values which enable the robot's motors to move as much as it is needed are concatenated in the SCPI commands sent to the robot's SSC-32 controller using the VISA driver for RS-232.

B. Algorithm Details

Fig. 2 includes the overlay drawing on the image acquired with a webcam. P_0 is the point from the base of the robotic arm (blue bottle stopper), P_1 is the point from the elbow of the robotic arm (yellow bottle stopper), P_2 is the point from the gripper of the robotic arm (red bottle stopper) and P_T is the point from the targeted object (green bottle stopper). P_2P_4 is an orthogonal vector to P_2P_0 and has the same length. P_2P_5 is an orthogonal vector to P_2P_1 and has the same length. P6 is the intersection between the P_TP_6 and P_5P_2 vectors, which is parallel to the P_4P_2 vector. P7 is the intersection between the P_TP_7 and P_4P_2 vectors, which is parallel to the P_5P_2 vector. Angle α is actually $P_TP_2P_6<$ angle and angle β is actually $P_TP_2P_7<$ angle. These two angles are used for computing slopes used in the equation of the straight line.

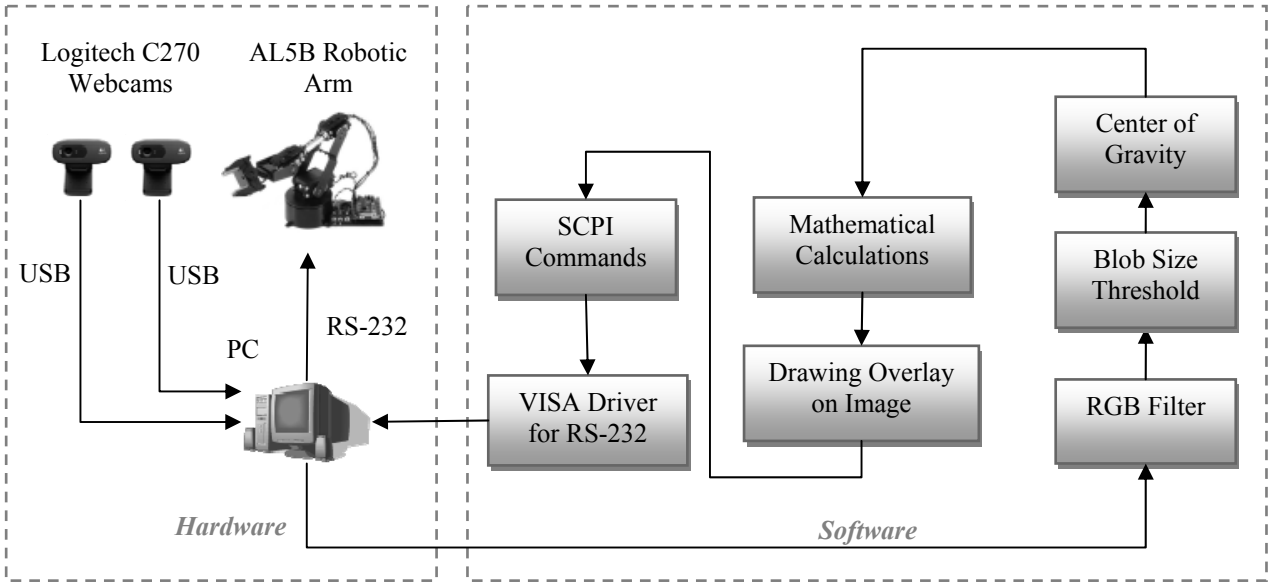

Figure 1. Block diagram of the experiment

The difference between two vectors can be computed as in (3).

$$\begin{cases} \Delta_x = x_2 - x_0 \\ \Delta_y = y_2 - y_1 \end{cases} \tag{3}$$

The vector length could be computed as shown in (4). This is the Euclidian norm.

$$\|l\| = \sqrt{x^2 + y^2} \tag{4}$$

The orthogonal vector is presented in (5).

$$\begin{cases} \tilde{x}_{ort} = y \\ \tilde{y}_{ort} = -x \end{cases} \tag{5}$$

Combining (3) with (5), the coordinates of points P_4 and P_5 can be obtained in (6) and (7).

$$\begin{cases} x_4 = \Delta_{x_{0x}} - 2 \cdot \Delta_{x_{0x}} + x_2 \\ y_4 = \Delta_{x_{0y}} - 2 \cdot \Delta_{x_{0y}} + y_2 \end{cases} \tag{6}$$

$$\begin{cases} x_5 = \Delta_{y_{0x}} - 2 \cdot \Delta_{y_{0x}} + x_2 \\ y_5 = \Delta_{y_{0y}} - 2 \cdot \Delta_{y_{0y}} + y_2 \end{cases} \tag{7}$$

The parallelogram $P_7P_TP_6P_2$ is calculated in the following way. First the slope (m) of the two tangents of the circles is computed as shown in (8).

$$\begin{cases} m_\alpha = \dfrac{y_5 - y_2}{x_5 - x_2} \\ m_\beta = \dfrac{y_4 - y_2}{x_4 - x_2} \end{cases} \tag{8}$$

The segment which goes through the points P_T, P_2 and used for the robotic arm's gripper is obtained in (9).

$$y_2 = m_\alpha x_2 + b \tag{9}$$

The Y intercept (b) is introduced in (9) and the next relation (10) is generated.

$$y = m_\alpha x + y_2 - m_\alpha x_2 \tag{10}$$

Using (10) twice for the x points and twice for the y points for both slopes (m_α and m_β), the expressions (11)-(14) are derived.

$$y_6 = m_\alpha x_6 + y_2 - m_\alpha x_2 \tag{11}$$

$$y_7 = m_\beta x_7 + y_2 - m_\beta x_2 \tag{12}$$

$$y_6 = m_\beta x_6 + y_T - m_\beta x_T \tag{13}$$

$$y_7 = m_\alpha x_7 + y_T - m_\alpha x_T \tag{14}$$

Next, equating y_6 in (11) and (13) yields (15).

$$m_\alpha x_6 - m_\beta x_6 = y_T - m_\beta x_T - y_2 + m_\alpha x_2 \tag{15}$$

Finally P_6 coordinates are obtained in (16).

$$\begin{cases} x_6 = \dfrac{(m_\alpha x_2 - m_\beta x_T + y_T - y_2)}{m_\alpha - m_\beta} \\ y_6 = m_\alpha (x_6 - x_2) + y_2 \end{cases} \tag{16}$$

Performing the same calculations for y_7, the coordinates for P_7 can be expressed in (17) and (18).

$$m_\beta x_7 - m_\alpha x_7 = y_T - m_\alpha x_T - y_2 + m_\beta x_2 \tag{17}$$

$$\begin{cases} x_7 = \dfrac{(m_\beta x_2 - m_\alpha x_T + y_T - y_2)}{m_\beta - m_\alpha} \\ y_7 = m_\beta (x_7 - x_2) + y_2 \end{cases} \tag{18}$$

The stereo distance calculation is explained in Fig. 3. The tangent of the *a* and *b* angles are shown in (19).

$$\begin{cases} tg(a) = \dfrac{distance}{camera\_separation} \\ tg(b) = \dfrac{offset}{conversion\_factor} \end{cases} \tag{19}$$

The offset distance can be computed as the difference from the initial right and left points (20).

$$offset = \left| x_{0R} - x_{0L} \right| \tag{20}$$

The conversion factor needs to be computed for calibration as shown on equation (21).

$$conversion\_factor = \\ = \frac{offset \cdot initial\_distance}{camera\_separation} \tag{21}$$

The offset distance can be expressed by the difference from the new right and left points, as shown in (22).

$$offset = \left| x_{2R} - x_{2L} \right| \tag{22}$$

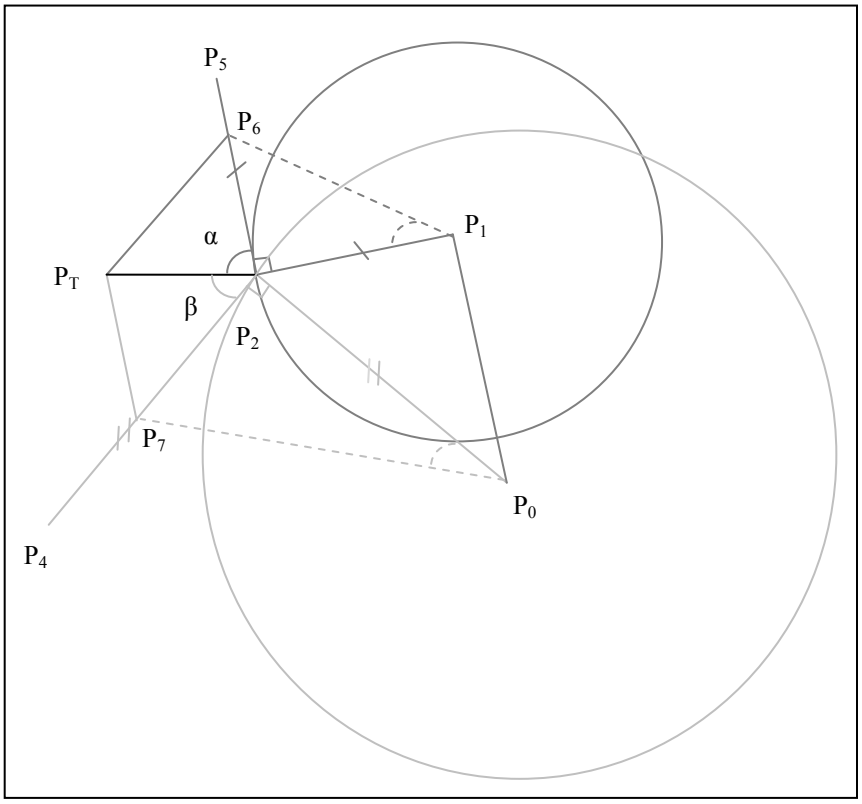

Figure 2. The overlay drawings on the robotic arm for the mathematical computations

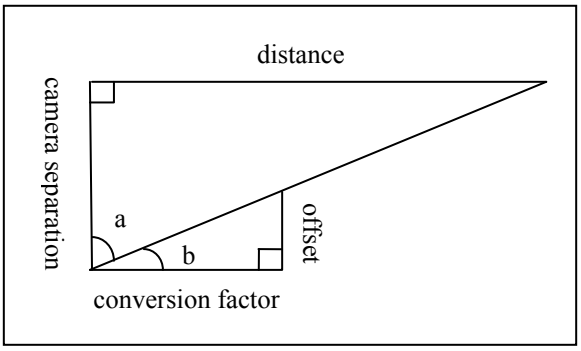

Figure 3. Stereo distance calculation algorithm

The final distance is (23).

$$final\_distance = \frac{conversion\_factor \cdot camera\_separation}{offset} \quad (23)$$

The angle in degrees is computed as shown in (24). The two segments (tangent and radius) form a right triangle in a circle which is useful to compute the movement angle of a motor.

$$angle = \frac{180°}{\pi} \cdot arctg\left(\frac{tangent\_length}{radius\_length}\right) \quad (24)$$

The robotic values are computed as shown in (25).

$$final\_robotic\_values = \\ = angle \cdot 11.(1)\_robotic\_values \quad (25)$$

The real world coordinates were converted to pixels as follows. The monitor size used in the setup is 19" and the resolution of the captured video image is 320x240 pixels as shown in (26).

$$\begin{cases} horizontal\_resolution = 320 \\ vertical\_resolution = 240 \\ diagonal = 19'' \end{cases} \quad (26)$$

To compute pixels from real world coordinates, the pixel density is needed, which can be obtained from (27). From (26) the data is 21.05 PPI.

$$display\_size = 15.2'' \cdot 11.4'' = 173.28in^2 \\ at 21.05 PPI \quad (27)$$

To compute the distance in pixels it's needed to know the pixel density from (27) and it's needed to convert the length from inch to cm, because the measurement of the distance was done in cm as shown on (28).

$$scale[pixels] = \\ = real\_world\_distance[cm] \cdot \frac{21.05[PPI]}{2.54[cm]} \quad (28)$$

The execution time of the algorithm is quite fast. The robotic arm can go to the target position in less than 1 second and can go back to the home position again in less than 1 second. The code memory usage was 2552 KB. The same algorithm was also ported on a Raspberry PI development board and displayed a similar performance (512 MB of RAM stored both the operation system and the code).

The whole code compiled executable has 964 KB size on PC.

The initial image (left and right) of the Lynxmotion AL5B robotic arm created with two cameras placed one near another (stereo camera configuration) is presented in Fig. 4. These images are processed in MATLAB (Fig. 5) to generate a 3D model of the actual robot. This 3D image

generated from the two initial images (left and right) is placed on a 3D graph, which can be moved with mouse.

On Fig. 6 these is a 3D model of the Lynxmotion AL5 type robotic arm with the overlay drawings generated by the image recognition algorithm.

Figure 4. Initial images (left and right) of the robotic arm used for a 3D image generation in MATLAB

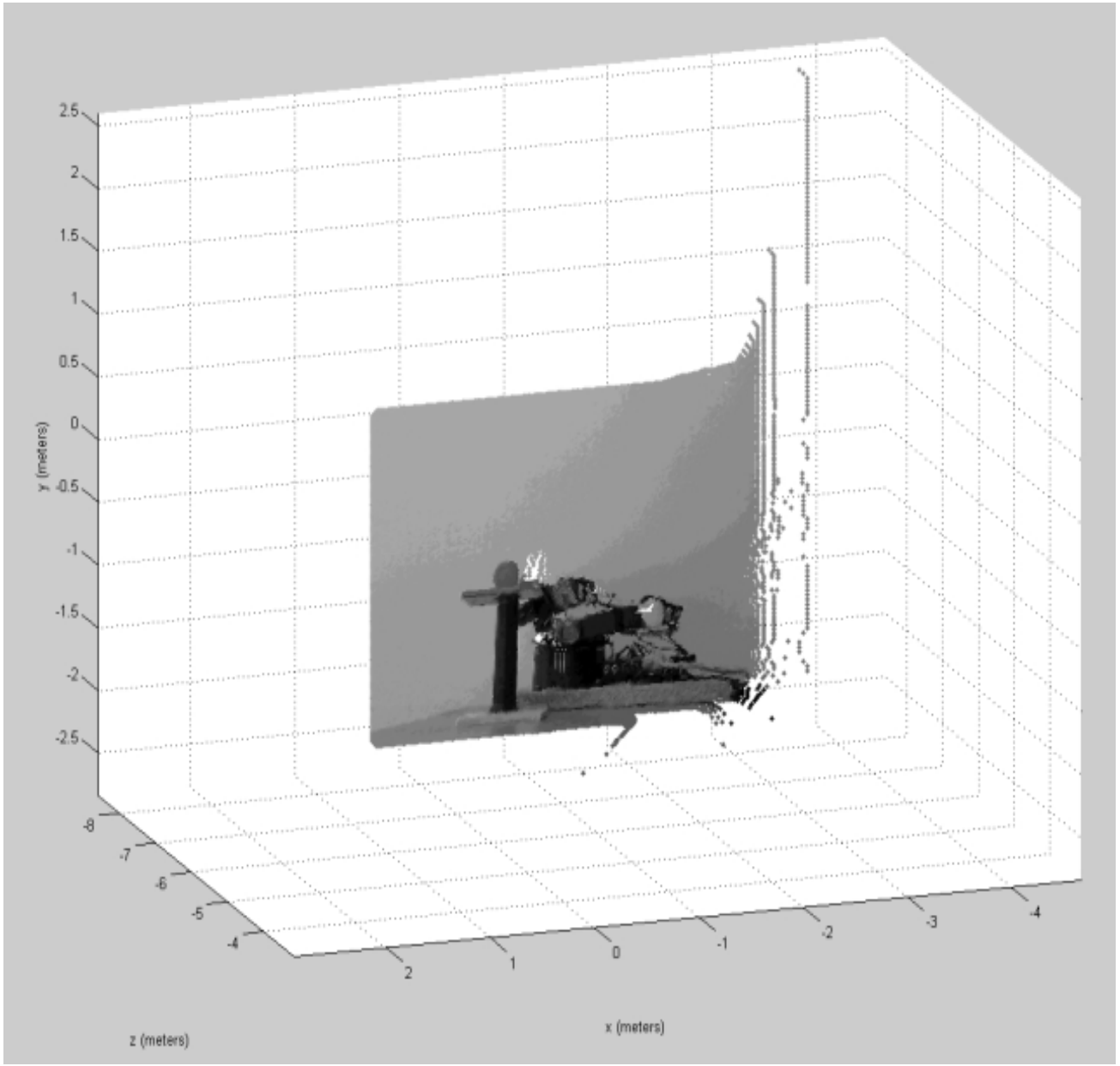

Figure 5. The 3D image created in MATLAB

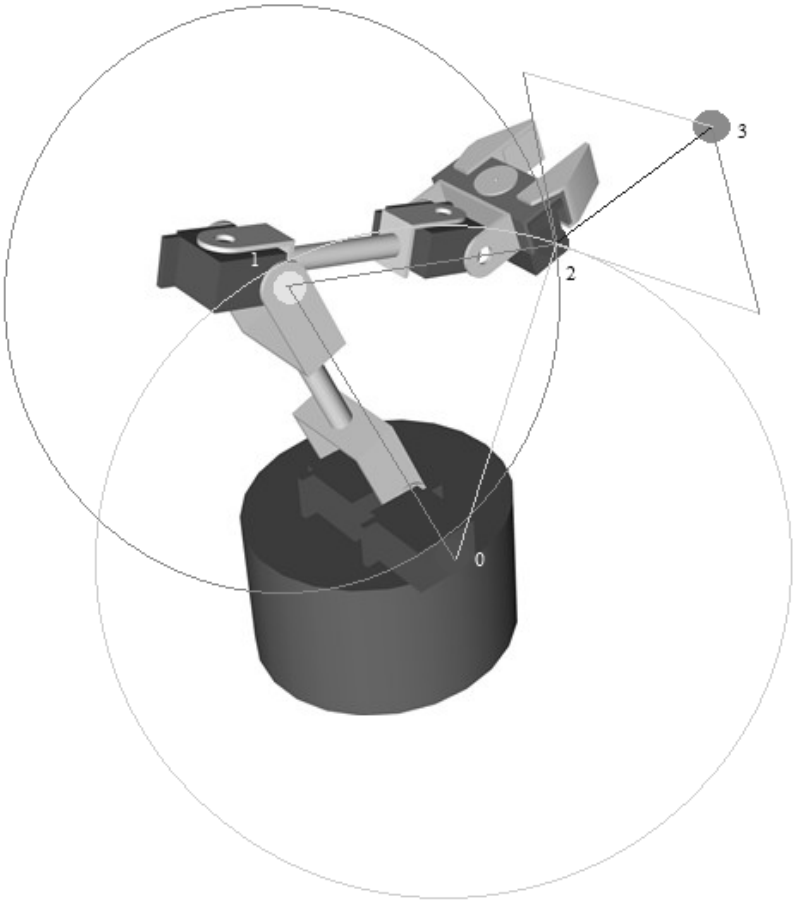

Figure 6. The 3D model of the Lynxmotion AL5 type robotic arm with the overlays generated by the presented image recognition algorithm

C. Software Implementation

The software was implemented in LabWindows/CVI. This programming language was chosen, because it has an excellent GUI (Graphical User Interface) and has the possibility to control many computer ports without using external libraries. In the experiments, the communication protocol is ensured by the RS-232 interface.

The main window of the application is presented in Fig. 7.

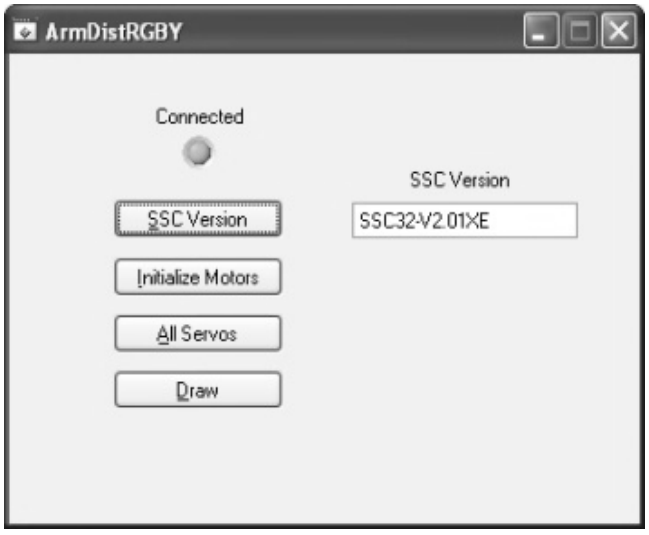

Figure 7. Robotic arm control GUI

The connected LED indicator informs the operator that the webcams are operational. The SSC Version button reads the version of the SSC-32 servo control board (being responsible for the robotic arm's motor control). The current version of the board is 2.01XE. The Initialize Motors button tests the digital ports of the ATmega168 microcontroller from the SSC-32 board and also the servomotors. The All Servos button "wakes up" the robotic arm. Basically it puts all servos in the middle position, which has the value of 1500. The robot is assembled mechanically in a way that after "waking up" it has the shape of the Greek capital gamma letter (Γ). This is the robots initial position.

The Draw button draws all the overlays over the robotic arm's webcam image and moves the robotic arm to reach the desired object (in our case, the green bottle stopper).

The algorithm in pseudo-code is shown next.

function DiffPoint (pos_1, pos_2)

$diff.x \leftarrow pos_1.x - pos_2.x$

$diff.y \leftarrow pos_1.y - pos_2.y$

function VectLen (vect)

$$\sqrt{vect.x^2 + vecy.y^2}$$

function Orthogonalize (vect)

$vect.x \leftrightarrow vect.y$

main ()

$xAxis \leftarrow DiffPoint\ (point[2], point[0])$

$yAxis \leftarrow DiffPoint\ (point[2], point[1])$

$xAxis \leftarrow Orthogonalize\ (xAxis)$

$yAxis \leftarrow Orthogonalize\ (yAxis)$

$$m_1 \leftarrow \frac{point[5].y - point[2].y}{point[5].x - point[2].x}$$

$$m_2 \leftarrow \frac{point[4].y - point[2].y}{point[4].x - point[2].x}$$

$$x_1 \leftarrow \frac{m_1 \cdot point[2].x - m_2 \cdot target.x}{m_1 - m_2} + \frac{target.y - point[2].y}{m_1 - m_2}$$

$$y_1 \leftarrow m_1 \cdot (x_1 - point[2].x) + point[2].y$$

$$x_2 \leftarrow \frac{m_2 \cdot point[2].x - m_1 \cdot target.x}{m_2 - m_1} + \frac{target.y - point[2].y}{m_2 - m_1}$$

$$y_2 \leftarrow m_2 \cdot (x_2 - point[2].x) + point[2].y$$

$xAxis \leftarrow DiffPoint\ (point[2], point[0])$
$gradlen \leftarrow VectLen\ (xAxis)$
$yAxis \leftarrow DiffPoint\ (point[2], point[1])$
$bradlen \leftarrow VectLen\ (yAxis)$
$diffg \leftarrow DiffPoint\ (point[7], point[2])$
$gtanlen \leftarrow VectLen\ (diffg)$
$diffb \leftarrow DiffPoint\ (point[6], point[2])$
$btanlen \leftarrow VectLen\ (diffb)$

$$degb \leftarrow \frac{180}{\pi} \cdot arctg\left(\frac{btanlen}{bradlen}\right)$$

$$degg \leftarrow \frac{180}{\pi} \cdot arctg\left(\frac{gtanlen}{gradlen}\right)$$

$$degd \leftarrow \frac{180}{\pi} \cdot arctg\left(\frac{dtanlen}{gradlen}\right)$$

$$robodegb \leftarrow degb \cdot \frac{2000}{180}$$

$$robodegg \leftarrow degg \cdot \frac{2000}{180}$$

$$robodegd \leftarrow degd \cdot \frac{2000}{180}$$

V. Experimental Results

The initial stereo image of the Lynxmotion AL5B robotic arm image acquired with the two webcams is presented in Fig. 8.

The result of the overlay on the Lynxmotion AL5B robotic arm initial image, after the Draw button is pressed is introduced in Fig. 9. All the colors are recognized and after the required mathematic calculations, the lines and circles are drawn. Finally after calculating the length of the sides of the parallelogram, the final angles were computed, which were needed to for each motor to move in the desired position.

The overlay on the SCORBOT-ER III robotic arm image is shown in Fig. 10, case used for final validation of the proposed method.

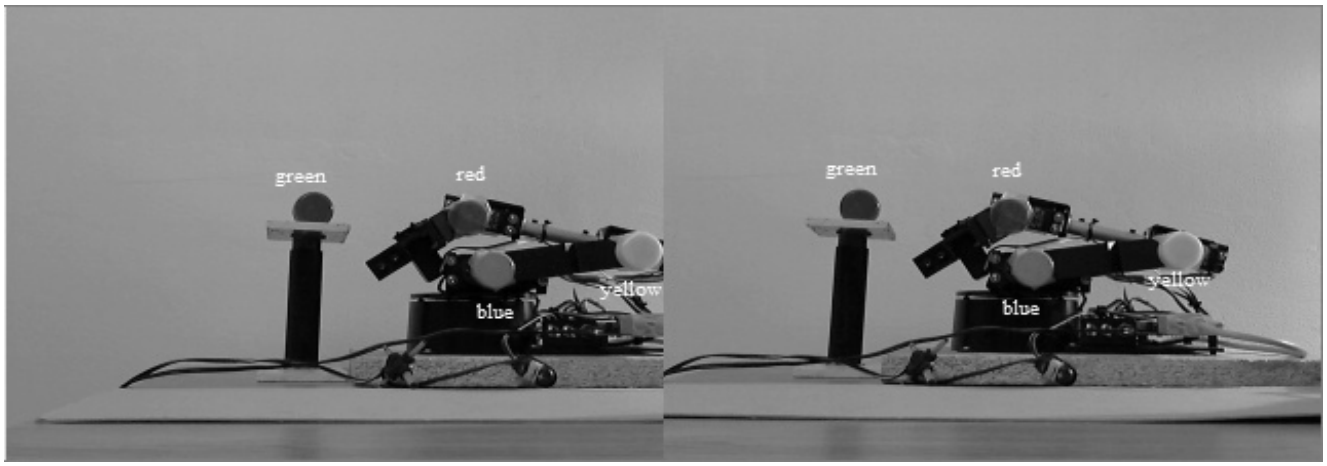

Figure 8. The initial stereo image of the Lynxmotion AL5B robotic arm acquired by the two webcams (blue bottle stopper at base, yellow bottle stopper at elbow, red bottle stopper at gripper of robotic arm and green bottle stopper at target)

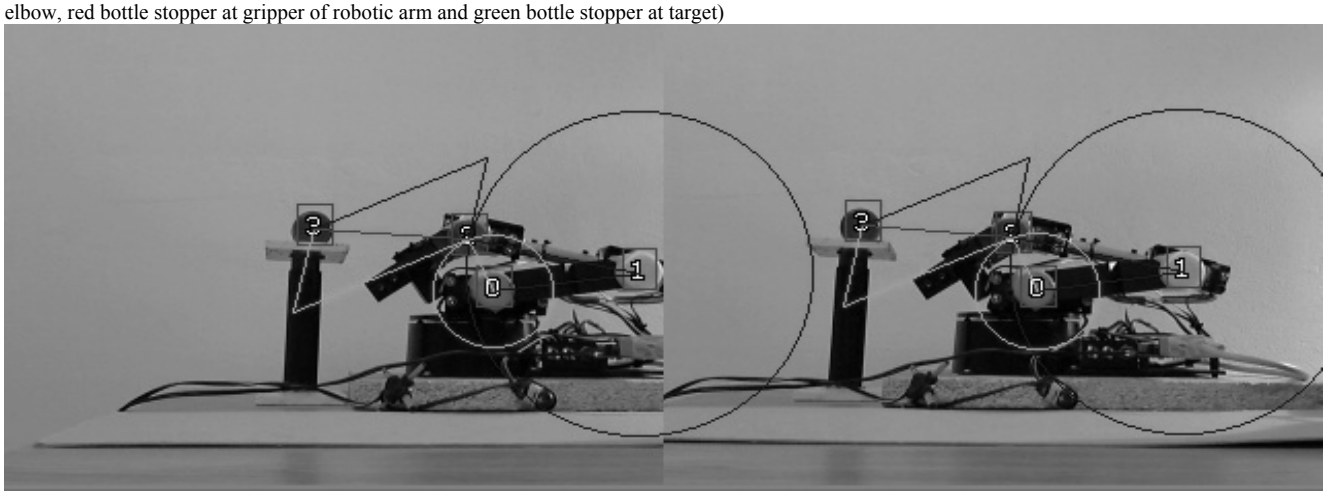

Figure 9. Overlay of the lines and circles on the Lynxmotion AL5B robotic arm after the correct color detection on initial stereo image from Fig. 8 (green small circle with green tangent, blue big circle with blue tangent, green parallel line to blue tangent, blue parallel line to green tangent, red line between points 2 and 3, red square around points 0, 1, 2 and 3)

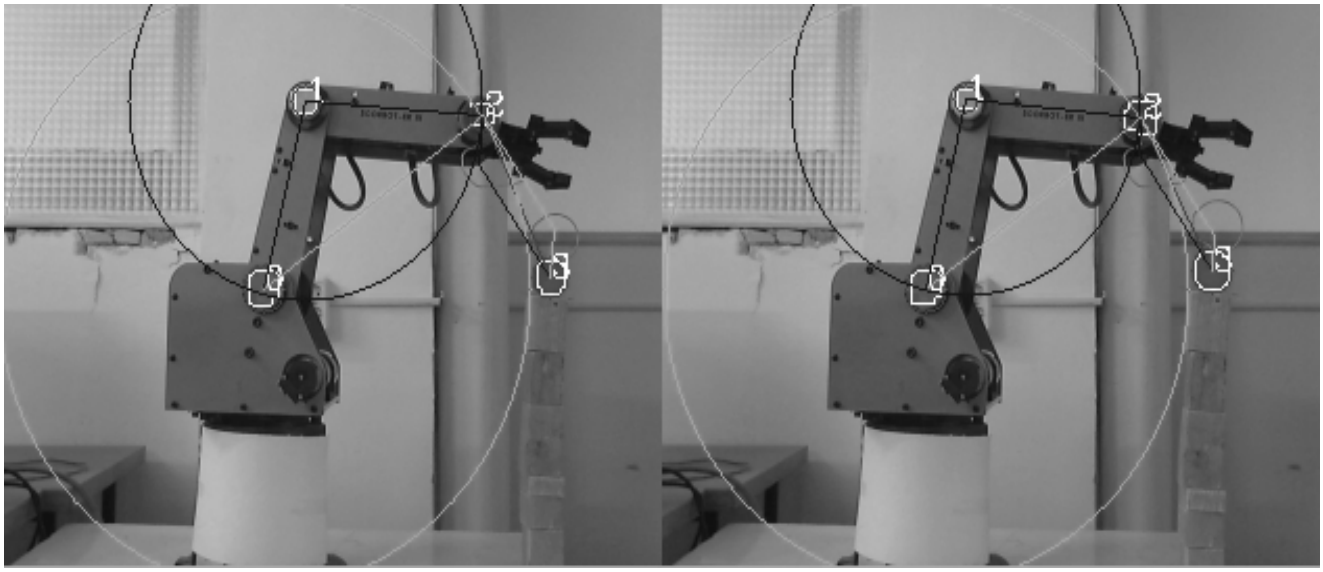

Figure 10. Overlay of the lines and circles on the SCORBOT-ER III robotic arm after the correct color detection on initial stereo image

The movement on the 0X axis is ensured by the stereo triangulation calculations, which were presented in the previous chapter.

The measurement results for the stereo distance computation are introduced in Table I. The robotic arm was placed in different positions with respect to the cameras. The real distance was measured with laser and compared to the projected distance which was computed in the software.

TABLE I. POSITIONING ERROR

Real Distance [m]	Projected Distance [m]	Delta [mm]
0,5	0,495	5
0,6	0,594	6
0,7	0,696	4
0,8	0,795	5
0,9	0,897	3
1	0,996	4
1,1	1,094	6
1,2	1,195	5
1,3	1,293	7
1,4	1,398	2

The relation between the error and the distance is represented in Fig. 11. It can be concluded that the stereo distance computation algorithm worked well in 99% of the tested time.

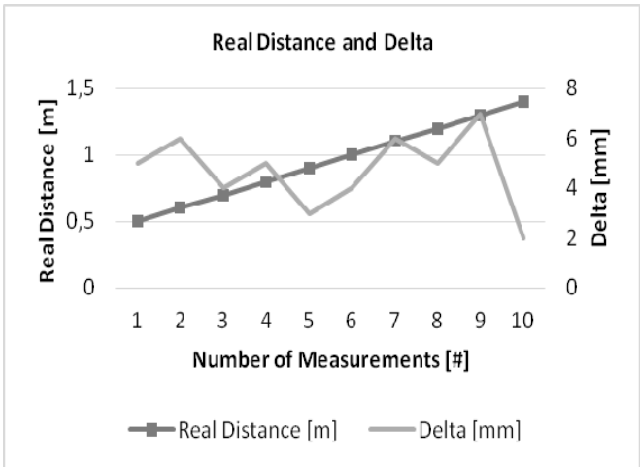

Figure 11. Real distance and delta

In Fig. 12 the error distribution is introduced, the highest positioning error is 5 mm at over 10 m distance, which in our case can turn to even smaller error because the practical distances are under 1 m. The error can be further reduced using higher resolution cameras.

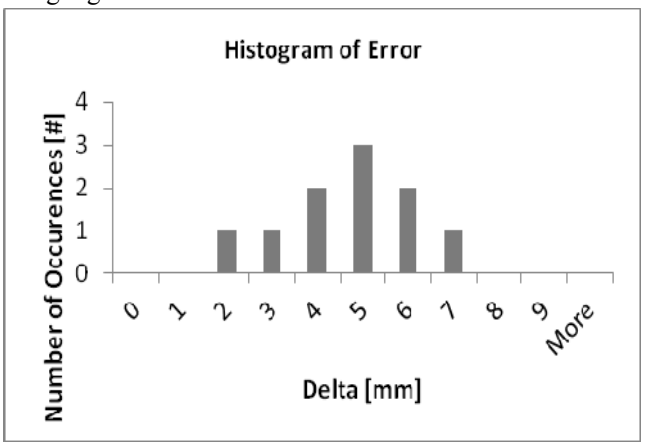

Figure 12. Histogram of positioning error (real distance and projected distance)

In Fig. 13 these is the process capability sixpack made for the positioning error in Minitab.

The upper and lower limits were set between -2 and 12 mm, because this would be acceptable. From the I Chart and Moving Range Chart is clear that everything is in these limits. The histogram and normal probability plot shows that the distributions of the values have a normal distribution. For this small amount of measured values the best graph is a normal probability plot, which creates a better graph than the histogram, which is generated just by the default by Minitab in the process capability analysis sixpack.

The σ (standard deviation) is 1,675 and the C_{pk} (process capability index) is 1,33; which is equal to 4σ quality, which is a really good result for an existing process ,the 6σ process is the ideal process, not existent in the real world.

The C_{pk} formula is presented in (29), where USL is the upper specification limit, LSL is the lower specification limit, μ is the mean and σ is the standard deviation.

$$C_{pk} = min\left[\frac{USL-\mu}{3\sigma}, \frac{\mu-LSL}{3\sigma}\right] \quad (29)$$

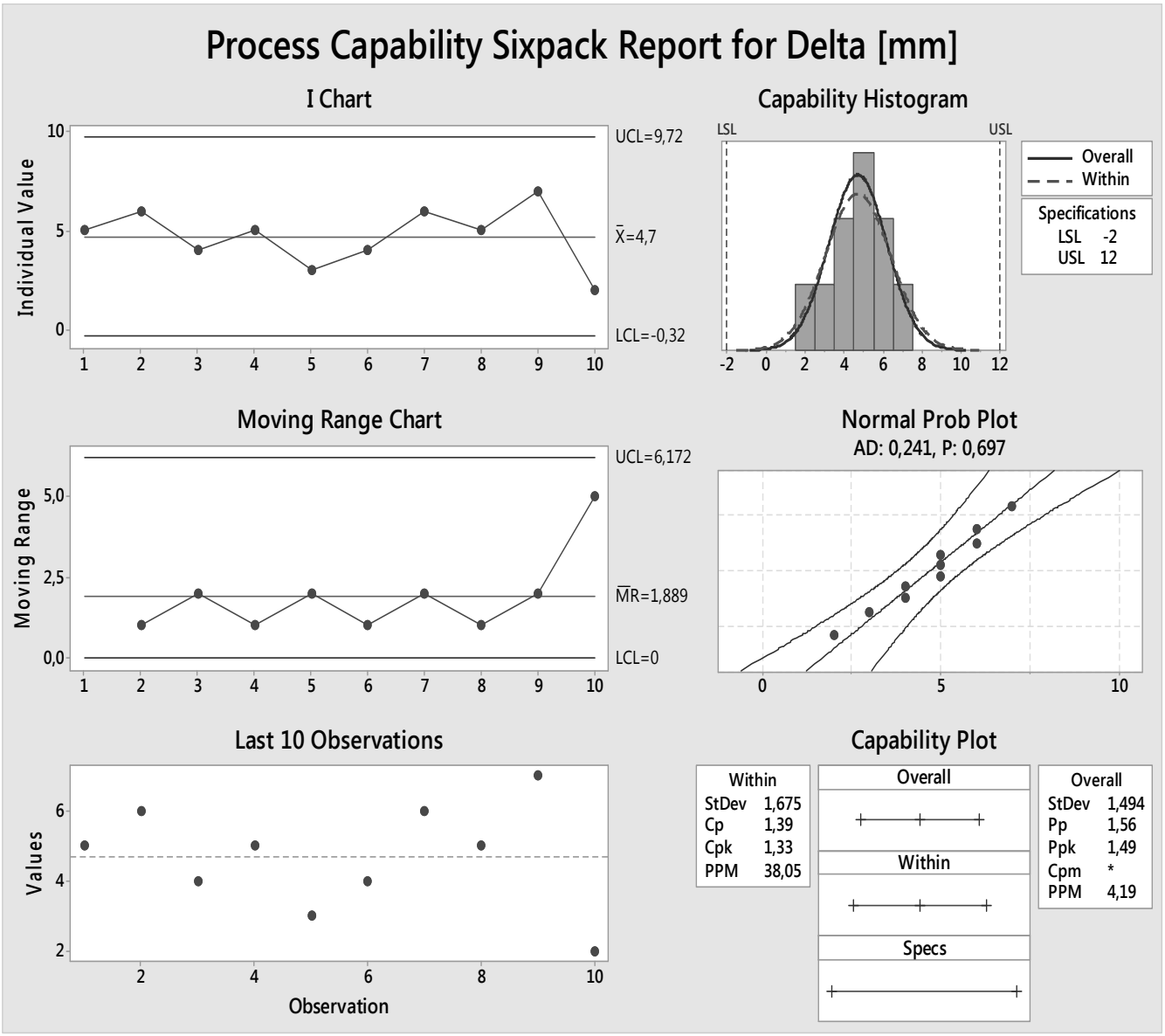

Figure 13. Process capability sixpack of positioning error

VI. Conclusion

The paper presented all the steps involved in an application development which eventually can control the robotic arm in the 3D space, using stereo vision. In the example, the robotic arm had three motors, one for each axis in the XYZ coordinate system.

The algorithm is based on two distance calculation methods. The first computes the movement position of the 0Y and 0Z axes (in 2D). This will make possible find the needed movement angle for robot base ($P_2P_0P_7<$) on 0Y axis and robot elbow ($P_2P_1P_6<$) on 0Z axis. The second is the stereo distance calculation where the movement of the base on the 0X axis is performed. Based on these values, the movement of the robotic arm in 3D is derived. For distance calculation two cameras were used. First, the distance to the base of the robotic arm and afterwards the distance to the target were evaluated. The difference of these two distances is the distance for the base motor to move on the 0X axis. This distance is one of the sides of a right triangle; the other side can be the radius (P_0P_2).

On Table II. there is a comparison of the presented method versus the methods of other authors.

TABLE II. Comparison of Algorithm to Similar Methods

Characteristics	Presented Method	[12] M. Seelinger	[13] R. Kelly	[14] V. Lippiello	[15] M. Kazemi	[16] R. T. Fomena
Number of Joints	3	6	2	multi-finger	6	6
Cost	low	low	low	high	high	high
Precision	high	high	high	high	high	high
Execution Time	< 1s	< 1s	< 1s	< 1s	< 1s	< 1s
Complexity	low	medium	low	high	high	high
Memory Usage	low	medium	low	high	high	high

In order to further evaluate the algorithm, the authors will also experiment it using different programming languages and operating systems to compare the resources involved.

The final target is to have all the system running on an FPGA and if possible, on an ASIC.

This algorithm would scale up to bigger robotic arms too; it can be also extended to be used for robotic arms with more joints just by repeating the algorithm. The robotic arm can used to manipulate almost any object. One of the advantages of the method is the low to medium complexity, being easy to implement. It does not need complex matrix computations or forward and inverse kinematics

calculations. The algorithm can be also combined with the current method used in the industry. The goal of the algorithm is not to replace the already well tested methods from industry but to work along them. The most useful filed of usage could be to repeatedly optically calibrate the robotic arm during execution, this way the production line would not need to be stopped and this way a lot of calibration cost could be reduced.

REFERENCES

[1] W. G. Hao, Y. Y. Leck, L. C. Hun, "6-DOF PC-Based Robotic Arm (PC-ROBOARM) with efficient trajectory planning and speed control," 4th International Conference On Mechatronics, Kuala Lumpur, pp. 1-7, 2011. [Online]. Available: http://dx.doi.org/10.1109/ICOM.2011.5937171

[2] W. Yang, J. H. Bae, Y. Oh, N. Y. Chong, B. J. You, S. R. Oh, "CPG based self-adapting multi-DOF robotic arm control," International Conference on Intelligent Robots and Systems, Taipei, pp. 4236-4243, 2010. [Online]. Available: http://dx.doi.org/10.1109/IROS.2010.5651377

[3] E. Oyama, T. Maeda, J. Q. Gan, E. M. Rosales, K. F. MacDorman, S. Tachi, A. Agah, "Inverse kinematics learning for robotic arms with fewer degrees of freedom by modular neural network systems," International Conference on Intelligent Robots and Systems, pp. 1791-1798, 2005. [Online]. Available: http://dx.doi.org/10.1109/IROS.2005.1545084

[4] N. Ahuja, U. S. Banerjee, V. A. Darbhe, T. N. Mapara, A. D. Matkar, R.K. Nirmal, S. Balagopalan, "Computer controlled robotic arm," 16th IEEE Symposium on Computer-Based Medical Systems, New York, pp. 361-366, 2003. [Online]. Available: http://dx.doi.org/10.1109/CBMS.2003.1212815

[5] M. H. Liyanage, N. Krouglicof, R. Gosine, "Design and control of a high performance SCARA type robotic arm with rotary hydraulic actuators," Canadian Conference on Electrical and Computer Engineering, St. John's, CA, pp. 827-832, 2009. [Online]. Available: http://dx.doi.org/10.1109/CCECE.2009.5090244

[6] M. Mariappan, T. Ganesan, M. Iftikhar, V. Ramu, B. Khoo, "A design methodology of a flexible robotic arm vision system for OTOROB," International Conference on Mechanical and Electrical Technology, Singapore, pp. 161-164, 2010. [Online]. Available: http://dx.doi.org/10.1109/ICMET.2010.5598341

[7] H. Guo-Shing, C. Xi-Sheng, C. Chung-Liang, "Development of dual robotic arm system based on binocular vision," International Automatic Control Conference, Nantou, pp. 97-102, 2013. [Online]. Available: http://dx.doi.org/10.1109/CACS.2013.6734114

[8] R. Szabó, A. Gontean, "Controlling a Robotic Arm in the 3D Space with Stereo Vision," 21th Telecommunications Forum, Belgrade, pp. 916-919, 2013. [Online]. Available: http://dx.doi.org/10.1109/TELFOR.2013.6716380

[9] R. Szabó, A. Gontean, "Robotic arm control in 3D space using stereo distance calculation," International Conference on Development and Application Systems, Suceava, pp. 50-56, 2014. [Online]. Available: http://dx.doi.org/10.1109/DAAS.2014.6842426

[10] R. Szabó, A. Gontean, "Remotely Commanding the Lynxmotion AL5 Type Robotic Arms," 21th Telecommunications Forum, Belgrade, pp. 889-892, 2013. [Online]. Available: http://dx.doi.org/10.1109/TELFOR.2013.6716373

[11] R. Szabó, A. Gontean, "Creating a Programming Language for the AL5 Type Robotic Arms," 36th International Conference on Telecommunications and Signal Processing, Rome, pp. 62-65, 2013. [Online]. Available: http://dx.doi.org/10.1109/TSP.2013.6613892

[12] M. Seelinger, E. Gonzalez-Galvan, M. Robinson, S. Skaar, "Towards a robotic plasma spraying operation using vision," IEEE Robotics & Automation Magazine, vol. 5, issue 4, pp. 33-38, 49, 1998. [Online]. Available: http://dx.doi.org/10.1109/100.740463

[13] R. Kelly, R. Carelli, O. Nasisi, B. Kuchen, F. Reyes, "Stable visual servoing of camera-in-hand robotic systems," IEEE/ASME Transactions on Mechatronics, vol. 5, issue 1, pp. 39-48, 2000. [Online]. Available: http://dx.doi.org/10.1109/3516.828588

[14] V. Lippiello, F. Ruggiero, B. Siciliano, L. Villani, "Visual Grasp Planning for Unknown Objects Using a Multifingered Robotic Hand", IEEE/ASME Transactions on Mechatronics, vol. 18, issue 3, pp. 1050-1059, 2013. [Online]. Available: http://dx.doi.org/10.1109/TMECH.2012.2195500

[15] M. Kazemi, K. K. Gupta, M. Mehrandezh, "Randomized Kinodynamic Planning for Robust Visual Servoing", IEEE Transactions on Robotics, vol. 29, issue 5, pp. 1197-1211, 2013. [Online]. Available: http://dx.doi.org/10.1109/TRO.2013.2264865

[16] R. T. Fomena, O. Tahri, F. Chaumette, "Distance-Based and Orientation-Based Visual Servoing From Three Points", IEEE Transactions on Robotics, vol. 27, issue 2, pp. 256-267, 2011. [Online]. Available: http://dx.doi.org/10.1109/TRO.2011.2104431

[17] N. C. Orger, T. B. Karyot, "A symmetrical robotic arm design approach with stereo-vision ability for CubeSats," 6th International Conference on Recent Advances in Space Technologies, Istanbul, pp. 961-965, 2013. [Online]. Available: http://dx.doi.org/10.1109/RAST.2013.6581353

[18] F. Medina, B. Nono, H. Banda, A. Rosales, "Classification of Solid Objects with Defined Shapes Using Stereoscopic Vision and a Robotic Arm," Andean Region International Conference, Cuenca, pp. 226, 2012. [Online]. Available: http://dx.doi.org/10.1109/Andescon.2012.71

[19] M. Puheim, M. Bundzel, L. Madarasz, "Forward control of robotic arm using the information from stereo-vision tracking system," 14th International Symposium on Computational Intelligence and Informatics, Budapest, pp. 57-62, 2013. [Online]. Available: http://dx.doi.org/10.1109/CINTI.2013.6705259

[20] T. P. Cabre, M. T. Cairol, D. F. Calafell, M. T. Ribes, J. P. Roca, "Project-Based Learning Example: Controlling an Educational Robotic Arm With Computer Vision," IEEE Revista Iberoamericana de Tecnologias del Aprendizaje, vol. 8, issue 3, pp. 135-142, 2013. [Online]. Available: http://dx.doi.org/10.1109/RITA.2013.2273114

[21] G. S. Gupta, S. C. Mukhopadhyay, M. Finnie, "WiFi-based control of a robotic arm with remote vision," Instrumentation and Measurement Technology Conference, Singapore, pp. 557-562, 2009. [Online]. Available: http://dx.doi.org/10.1109/IMTC.2009.5168512

[22] L. Haoting, W. Wei, G. Feng, L. Zhaoyang, S. Yuan, L. Zhenlin, "Development of Space Photographic Robotic Arm based on binocular vision servo," Sixth International Conference on Advanced Computational Intelligence, Hangzhou, pp. 345-349, 2013. [Online]. Available: http://dx.doi.org/10.1109/ICACI.2013.6748528

[23] C. Wen-Chung, C. Chih-Wei, "Automatic Mobile Robotic Manipulation with Active Eye-to-Hand Binocular Vision," 33rd Annual Conference of the IEEE Industrial Electronics Society, Taipei, pp. 2944-2949, 2007. [Online]. Available: http://dx.doi.org/10.1109/IECON.2007.4460000

[24] P. C. Nunnally, J. M. Weiss, "An inexpensive robot arm for computer vision applications," Energy and Information Technologies in the Southeast, Columbia, vol. 1, pp. 1-6, 1989. [Online]. Available: http://dx.doi.org/10.1109/SECON.1989.132303

[25] T. Kizaki, A. Namiki, "Two ball juggling with high-speed hand-arm and high-speed vision system," IEEE International Conference on Robotics and Automation, Saint Paul, MN, pp. 1372-1377, 2012. [Online]. Available: http://dx.doi.org/10.1109/ICRA.2012.6225090