Real-Time Color-Based Sorting Robotic Arm System

Yonghui Jia, Guojun Yang and Jafar Saniie

Department of Electrical and Computer Engneering
Illinois Institute of Technology, Chicago, Illinois, USA

***Abstract*— Sorting is a labor intense process. With machines that can recognize objects, it is possible to automate the sorting process. In this paper, we present a robotic sorting arm based on color recognition technique. In this system, when a new frame is captured by the camera, the object will be detected using color-base image processing technique. The position of the object in real-world will be calculated by its mass center in image. Using Inverse Kinematics algorithms, the control input for the robotic arm will be calculated and then sent to Arduino microcontroller. Then, the microcontroller will drive the motors on the robotic arm to sort and position the objects according to their color. Using the proposed technique, the sorting robotic arm system can distinguish and sort different objects successfully with properly tuned parameters for both machine vision and 3D mobility of the robotic arm.**

I. INTRODUCTION

In recent years, robotic automation is a process that is extremely important for industrial environments, since it improves the quality while reduce the time spent to accomplish a given task, all with a minimal human intervention. People always been seeking for means of developing intelligent machines for their further employment to a variety of useful applications which has already achieved tremendous advances in the field of robotic[1]. Workers usually operate machines so that they can perform certain task. Operating process requires robust functional algorithms to deliver a desirable outcome efficiently. The main objective of the sorting machine is to control the robotic arm to perform exact movements for picking up arbitrary color objects from the panel and put it into an assigned sorting boxes.

In this paper, we present a new method to sort objects automatically using single camera image processing technique and Inverse Kinematics algorithm [2] [3]. Color-based detection, segmentation and locating objects captured by the camera in 2D plane are used to generate commands through a serial communication to a robotic arm controller for 3D picking and repositioning of the objects.

II. HARDWARE SYSTEM ARCHITECTURE

The hardware system layout including the robotic arm are shown Figure 1.

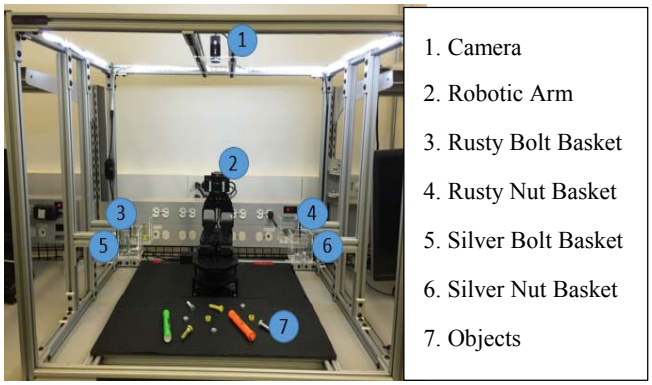


Figure 1. Sorting robotic arm system layout

Figure 2. Phantom X Reactor robotic arm (left) and Dynamixels AX-12A (right)

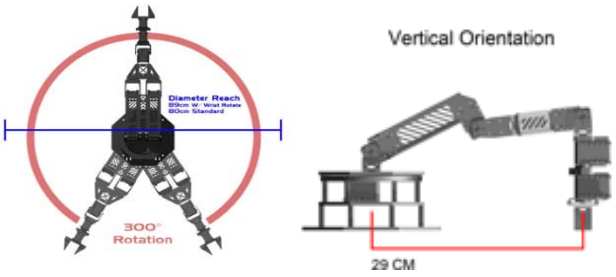


Figure 3. Base rotation (left) and vertical reach (right)

The robot controller is an ArbotiX-M microcontroller as shown in Figure 4. The controller has two Serial Ports with 28 Digital I/O, and it is programmable by the Arduino IDE software.

978-1-5090-4767-3/17/$31.00 ©2017 IEEE

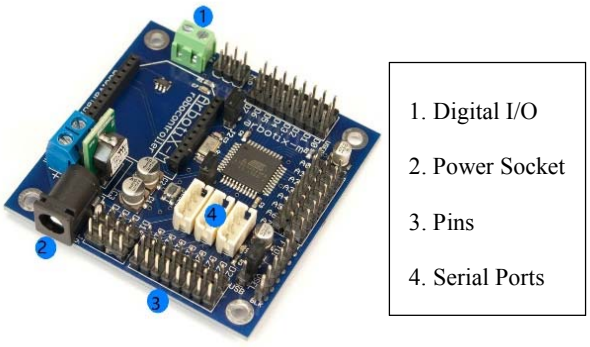


Figure 4. ArbotiX-M microcontroller

To build the system, following steps were taken:

- Object detection and recognition using input from camera.
- Estimate location of objects in 3D coordinate using their location in images.
- Solve the inverse kinematics problem using Denavit-Hartenberg matrix.
- Developing the software for Arduino microcontroller.

The work flow of the entire system is as follows:

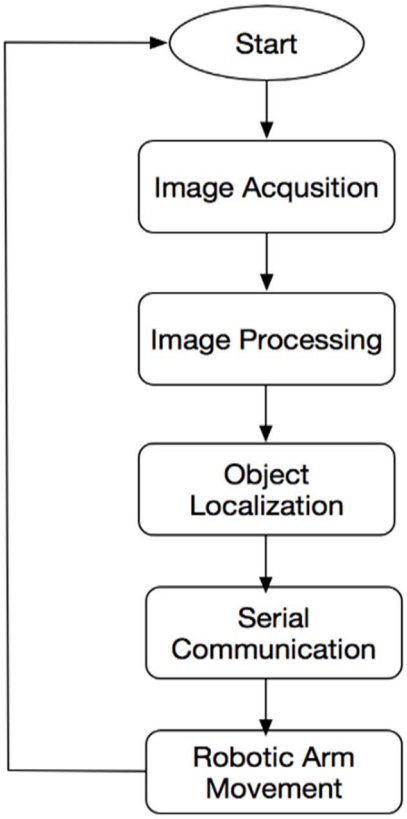


Figure 5. Sorting Robotic Arm Work Flow

III. Image Processing and Object Detection

Machine vision is achieved by using images captured via an external webcam mounted on the frame holding manipulator. These images are sent to the computer for image processing. We are using only one RGB camera. This camera is calibrated for distortion of images created by the lens using a camera calibration function implemented in OpenCV(Open Source Computer Vision library). Then the position of objects in image plane can be converted into position in 3D coordinate more accurately. This process is very important since it allows the system to calculate the position of objects using the image obtained by camera.

The process of camera calibration requires capturing several pictures of a printed black-white chessboard pattern on random positions while recording the size of the squares. The program, converts the pixel coordinates of the camera to millimeters through comparison between the real size of the pattern and its pixel size. The algorithm then outputs a calibration matrix and a matrix with the camera's intrinsic and extrinsic parameters. These data are used to convert pixel coordinates to 3D coordinates using equation 1, shown as below.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (1)$$

In this equation, the first vector contains the coordinates in pixel coordinates, where "z" represents, the "z" axis in homograph coordinate. In the 3 by 3 matrix, we have the intrinsic camera parameters, where "f_x" and "f_y" are the camera focal lengths while "c_x" and "c_y" denotes the optical center in pixel coordinates. The second vector denotes corresponding position in real world coordinates. Using images from calibrated camera, the objects can be captured without distortion.

All images captured with RGB camera will be converted to grayscale. These grayscale images will be further converted to binary images. Once been converted into binary images, these images will be processed via morphological operations which are Dilation and Erosion as shown in Figure 6 and Figure 7.

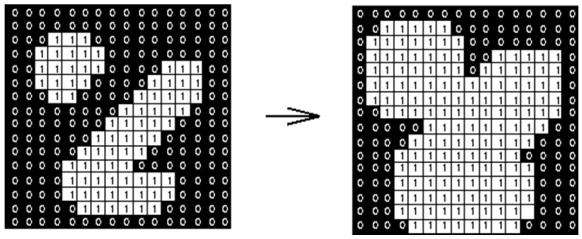


Figure 6. Dilation on a binary image

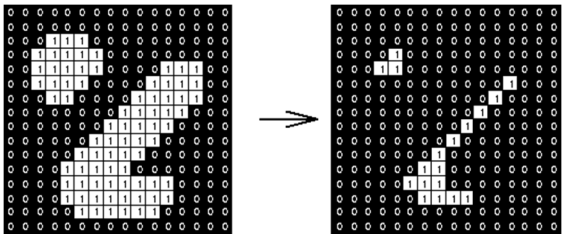
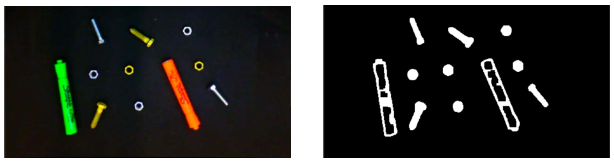


Figure 7. Erosion on a binary image

Dilation can gradually enlarge the boundaries of regions of foreground pixels (typically white pixels) making the foreground pixels grow while holes within those regions become smaller. Erosion is eroding away the boundaries of the foreground (white) pixels, shrink objects and enlarging holes within those areas. The combination of these operations allows us to remove most unwanted noise in the background, as well as isolating objects from each other. The result of these morphological operations can be seen in Figure 8.



(a) (b)

Figure 8. Morphological Operation Results (a) Original image; (b) after morphological operations

The next step is assessing circularities of the object to differentiate different objects which in this case are bolts and nuts, then differentiate the silver objects from the yellow objects. To perform color detection, the image is converted into HSV color space. HSV stands for hue, saturation, and value respectively. Hue represents the shade of the color; saturation describes the intensity of the color; and value (or brightness) describes how dark the color is, often referenced as luminance. A representation of these parameters can be seen in Figure 9. HSV separates the image intensity from the color information. When using color base method, HSV allows detecting objects with less error. The result of this color detection method can be seen in Figure 9 [4].

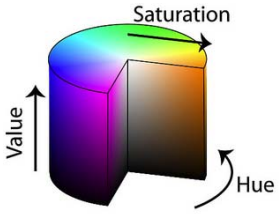
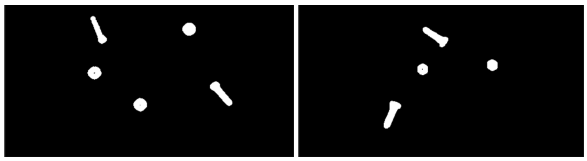


Figure 9. HSV color space representation

After combining of these image processing operations, two sets of images can be obtained: one with the information of whether it is a bolt, nut or other object, the other set of images are images of objects with color data. All this information is combined to successfully separate all the objects [5]. The results are shown in the Figure 10 as shown below.



(a) (b)

Figure 10. Image after color detection (a) Silver objects; (b) yellow objects

IV. Control of Robotic Arm

All information obtained through the image processing is sent to the Arduino via a serial port. To be more specific, sending the information of objects (centroid coordinates, object's orientation, and object type) through the serial port, which will be received by Arduino and used to control the robot.

The angle of each joint is estimated by Inverse Kinematics, where the information given is the object's centroids, and this is converted to robot joint angles. With angles of joints provided, robotic arm can reach and pick target object successfully. In our system, the Arduino receives the centroids and handles all the kinematics operation. After picking up an object, the coordinates of proper boxes are sent to the robotic arm. Then robotic arm will drop the object in the box. Finally, the robotic arm will return to initial gesture. The detecting and sorting processes are performed repeatedly until there is no more object within the reach of the robotic arm [6].

The robot arm takes control inputs from an Arduino. Arduino is responsible for the commands that directly control the motors and algorithms of the Inverse Kinematics [7]. Inverse Kinematic algorithm is based on angles of joints on the robotic arm need to be turned while Robot Arm is reaching a particular 3D position. Using the Denavit-Hartenberg [8] matrix to solve this task was possible, and it is an ideal approach when working with the kinematics of a robotic arm (see Figure 11).

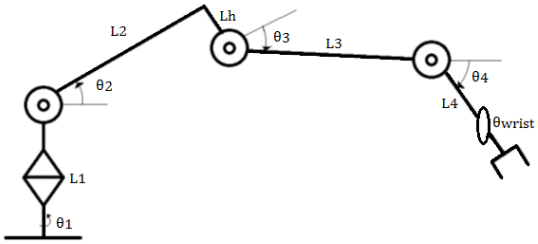


Figure 11. Robotic arm simplified [6] [9] [10]

The Denavit-Hartenberg parameters are obtained by calculating following equations using parameters shown in Table I [8]:

TABLE I. Denavit-Hartenberg Parameters

I	a	A	d	Θ
1	90	0	L1	θ1
2	0	Lh	0	θ2
3	0	L3	0	θ3
4	0	L4	0	θ4

$$T1 = \begin{bmatrix} \cos\theta 1 & 0 & \sin\theta 1 & 0 \\ \sin\theta 1 & 0 & -\cos\theta 1 & 0 \\ 0 & 1 & 0 & L1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

$$T2 = \begin{bmatrix} \cos\theta 2 & -\sin\theta 2 & 0 & Lh * \cos\theta 2 \\ \sin\theta 1 & \cos\theta 2 & 0 & Lh * \sin\theta 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

$$T3 = \begin{bmatrix} \cos\theta 3 & -\sin\theta 3 & 0 & L3 * \cos\theta 3 \\ \sin\theta 3 & \cos\theta 3 & 0 & L3 * \sin\theta 3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4)$$

$$T4 = \begin{bmatrix} \cos\theta 4 & -\sin\theta 4 & 0 & L4 * \cos\theta 4 \\ \sin\theta 4 & \cos\theta 4 & 0 & L4 * \sin\theta 4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

Using T1 – T4, it is possible to find the following three equation that are the physical Cartesian coordinates (x,y,z) to calculate the position that the robotic arm will reach based on the inputs given to the joint angles.

$$x = \cos(\theta 1) * (L3 * \cos(\theta 2 + \theta 3) + Lh * \cos(\theta 2) + L4 * \cos(\theta 2 + \theta 3 + \theta 4)) \quad (6)$$

$$y = \sin(\theta 1) * (L3 * \cos(\theta 2 + \theta 3) + Lh * \cos(\theta 2) + L4 * \cos(\cos(\theta 2 + \theta 3 + \theta 4)) \quad (7)$$

$$z = L1 + L3 * \sin(\theta 2 + \theta 3) + Lh * \sin(\theta 2) + L4 * \sin(\theta 2 + \theta 3 + \theta 4) \quad (8)$$

To capture objects in any given orientation (see Figure 12), the wrist rotation angle based on the wrist output values was provided by the software. Based on the geometrical analysis, the following equation was developed.

$$\alpha(wrist) = \alpha(base) + \alpha(object) - 90^{\circ} \quad (9)$$

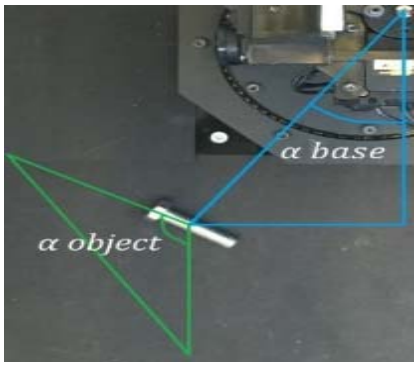


Figure 12. Angles for wrist calculation

Serial communication is setup between the computer which performs image processing algorithm and the Arduino which controls the robotic arm. The image processing part sends the coordinate, orientation, and the type of the object to the robot controller which transforms this information into joint movements for picking up the desired objects and placing it into the appropriate basket.

V. Software Implementation

To conduct the image processing in real time, the algorithms need to be implemented on the host computer. OpenCV is a library that includes a great number of the computer vision, image processing and general-purpose numerical algorithms for a large range of computer vision problems. The majority of the algorithms implemented in OpenCV are more efficient when compare with Matlab. OpenCV also has C/C++, Python, Java and other interfaces for the most popular programming language. To interface with Arduino, C++ is used to interface with image processing and serial communication. Given the nature, that C++ is a general-purpose, high-level programming language widely used for dealing with computer vision problems.

The control input of the arm is generated by the Inverse Kinematics algorithm implemented on Arduino controller. Arduino is an open-source single board microcontroller including software which comprises a standard programming language compiler. IDE and a bootloader loads the compiled code into the chip by serial port through USB port. With PCA9685 board, the Arduino controller can serve as an instrument for controlling all servo motors on the arm simultaneously.

After receiving the control inputs form Arduino controller, PWM (Pulse-width modulation) signals will be generated by ArbotiX-M microcontroller. These signals will be used to control each motor. Servo motor provides precise positioning of any joint to which it is stuck on the specific angle. Servo motors do not rotate continually and their rotation is restricted within fixed angles. Usually they are capable of rotating from 0 to 180 degree, but some of them manage to turn up to 360 degrees. Servo motor usually has three inputs: + 5V power, ground and PWM. By changing the PWM signal servo motor and consequently, the arm can perform different movements to sort the objects.

VI. Result Analysis

Some of the results for the robot pickup and delivering process are seen in Figure 13 and 14. We can see that every object was picked up and delivered properly. The system can distinguish the undesired objects and remove them from the table and then separate the bolts and nuts correctly and place them in the corresponding bins. Finally, the system can also distinguish overlapped bolts with different colors, and separate them by shifting or pushing them, so they can be picked in the next iterations. In the future, we will transfer the whole system from PC to Raspberry Pi.

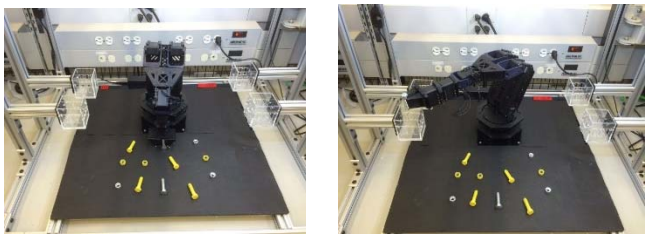


Figure 13. Picking up a silver bolt and delivering it to its correct box

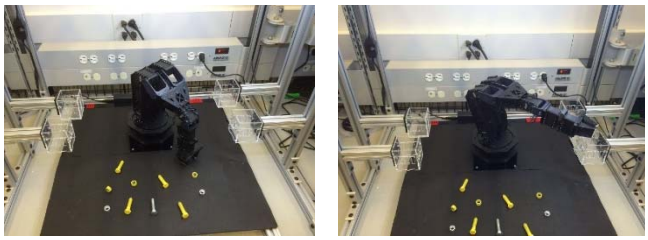


Figure 14. Picking up a silver nut and delivering it to its correct box

The performance of the system can be affected by a series of factors, such as illumination of the environment, the reflection on the surface of objects. With parameters in the system tuned, the system can adapt to different environments. Through an iterative process all object can be delivered properly, regardless of their positions, color and orientations.

VII. CONCLUSION

By combining image processing algorithms, Inverse Kinematics algorithm, we developed a robotic arm that can sort objects according to their shape and color. For future works, the image processing will be implemented on ARM based Raspberry Pi. Performing image processing on Raspberry Pi will reduce the size of the system while increase the efficiency in terms of power consumption. Installing light sensors can reduce the interference from the environment. Using more sophisticated neural network can help the system differentiate a larger variety of the objects.

REFERENCES

[1] N. Rai, B. Rai and P. Rai, "Computer vision approach for controlling educational robotic arm based on object properties," in *Emerging Technology Trends in Electronics, Communication and Networking (ET2ECN), 2014 2nd International Conference on*, 2014.

[2] R. Mussabayev, "Colour-based object detection, inverse kinematics algorithms and pinhole camera model for controlling robotic arm movement system," in *Electronics Computer and Computation (ICECCO), 2015 Twelve International Conference on*, 2015.

[3] P. S. Lengare and M. E. Rane, "Human hand tracking using MATLAB to control Arduino based robotic arm," in *Pervasive Computing (ICPC), 2015 International Conference on*, 2015.

[4] S. D. Gajbhiye and P. P. Gundewar, "A real-time color-based object tracking and occlusion handling using ARM cortex-A7," in *India Conference (INDICON), 2015 Annual IEEE*, 2015.

[5] Q. Ji and W. Qi, "A color management system for textile based on HSV model and bayesian classifier," in *Control, Automation, Robotics and Vision, 2008. ICARCV 2008. 10th International Conference on*, 2008.

[6] R. Szabó and A. Gontean, "Controlling a robotic arm in the 3D space with stereo vision," in *Telecommunications Forum (TELFOR), 2013 21st*, 2013.

[7] C.-L. Hwang and J.-Y. Huang, "Neural-network-based 3-D localization and inverse kinematics for target grasping of a humanoid robot by an active stereo vision system," in *Neural Networks (IJCNN), The 2012 International Joint Conference on*, 2012.

[8] R. Alqasemi and R. Dubey, "Kinematics, control and redundancy resolution of a 9-DoF wheelchair-mounted robotic arm system for ADL tasks," in *Mechatronics and its Applications, 2009. ISMA'09. 6th International Symposium on*, 2009.

[9] R. Szabó and G. Gontean, "Robotic arm control with stereo vision made in LabWindows/CVI," in *Telecommunications and Signal Processing (TSP), 2015 38th International Conference on*, 2015.

[10] R. Szabó, A. Gontean and A. Sfiraţ, "Robotic arm control in space with color recognition using a Raspberry PI," in *Telecommunications and Signal Processing (TSP), 2016 39th International Conference on* , 2016.