Controlling A Robotic Arm Using Hand Recognition Software

Ali Cetinkaya*‡, Onur Ozturk**, Ali Okatan***

*Technology Transfer Office, Istanbul Gelisim University, Avcılar, Istanbul, Turkey.

**School of Management, Faculty of Engineering, University College London (UCL), Euston, London, UK.

***Department of Computer Engineering, Faculty of Engineering, Istanbul Gelisim University, Avcılar, Istanbul, Turkey.

(alcetinkaya@gelisim.edu.tr, onur.ozturk.16@ucl.ac.uk, aokatan@gelisim.edu.tr)

Corresponding Author: Ali Cetinkaya, Technology Transfer Office, Istanbul Gelisim University, Avcılar, Istanbul, Turkey.
Tel: +90 212 422 70 00 / 7187. alcetinkaya@gelisim.edu.tr

Received: 21.09.2018 Accepted: 27.04.2019

Abstract- With the increasing need of repetitive tasks in the manufacturing industry, robotic automation is becoming a necessity. In the steel industry, workers become less efficient over time, causing interruptions during assembly. Robotic automation is capable of operating at highest efficiency therefore increasing productivity in the steel industry. The robot will be able to pick up and drop metallic object with the help of the electromagnet present on the robotic arm. The handling of the objects will be triggered by the hand gestures from the user. The image to be processed will be captured by an external camera. This robot is built as a prototype for the steel industry.

Keywords Gesture Recognition, OpenCV, Embedded System Robotic Arm Control, Embedded C.

1. Introduction

The aim of this system was to operate the robot with embedded tasks via finger recognition software coming from a video stream from the PC video output due to the reasons listen in the abstract [1]. The image recognition software was designed in OpenCV3, whereas the embedded system was designed in Arduino.

Finger recognition works by counting the empty spaces between the fingers. The finger recognition software works using convex hull algorithms and contour detection. In simpler terms, convex hull approach aims to confine a set of given points in a plane by the smallest polygon [2, 3].

OpenCV is a library designed for use in open source software and real-time image processing applications. This library, which can calculate all the moments of a polygonal and random shape, finds convexity lines through hand recognition [4, 5].

In sectors such as production, repetitive jobs are performed in most areas. There are many manpower and cost increases. An interactive design has been done by creating a robotic system to perform object recognition and operations [6].

Autonomous systems aim to control the movement of robots in robotic studies and researches. The algorithms used in such systems can make improvements by analyzing the errors that occur during the movements of the robots. These errors are calculated during the experiments on the robotic arms and in the next trials they provide the system with operational response [7].

A new algorithm for real-time and recognition is based on hand movement recognition. This algorithm is based on three main steps: hand partitioning, hand tracking and gesture recognition. Skin color, hand segmentation and monitoring based algorithm have been proposed for hand recognition. It has been investigated that it leads to good performances on the system [8].

The Kalman filter was used for the steady position of the hand movement by using convective neural networks for real-time interaction with hand movements [9]. For control of the robotic arm, control of the robotic arm is provided via the analog signals from the sensors. Wearable robotic arm is designed to simulate the natural movements of the human arm [10]. People who work in explosive and security operations are at risk and life-threatening. Therefore, we can keep the danger away from people by the help of a robotic hand on an autonomous device [11].

Motion control can be used on heavy machines. These movements are used to control the movement of robotic arms with image processing techniques [12]. The natural hand movement recognition system is developed upon hand recognition and recognition of hand movements. Hand tracking, background cleaning, and lighting status create difficulties in motion recognition systems [13]. It is controlled by sensors and accelerometers in a system that simulates the movement of the user's arm by manual tracking. Kinect simulates the skeleton of the human body and provides its three-dimensional coordinates to its users [14].

Systems designed to work on more than one person have been designed with a finger-tipped approach to recognizing hand movements [15].

With the removal of the hand zone from the background, the data sets of the hand movements are created by detecting the palms and the fingers. With the data sets created, the accuracy of the method is controlled [16].

A robot kinematics can be examined geometrically and the position, velocity and acceleration information of all of its movement from force and torque components can be extracted [17, 18].

The robotic arm is controlled via pre-defined actions in Arduino. The four actions defined by the robotic arm are as follows: forwards arm motion, turning on the electromagnet, backwards arm motion and turning off the electromagnet. The finger count for these actions are respectively 2, 3, 4 and 5. The fingers do not have to be showed in order. For example, showing index and pinky finger would still trigger action 2.

2. System and Software Algorithm’s

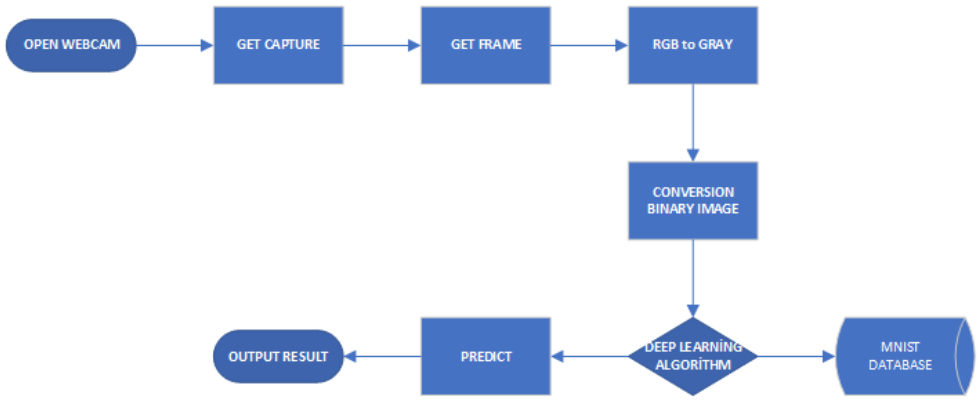

Fig. 1. Flow diagram of algorithm step.

Fig. 1 shows summaries the necessary steps in this study. In this study using the deep learning algorithm and Mnist data set consists of 3 main stages. These are: preprocessing on the image, deep learning algorithm and mnist data set.

The Arduino Mega code was produced using the Arduino IDE. Arduino is responsible for controlling the servo engines using the data received from the hand recognition software. For this purpose, the libraries used were standard to Arduino, therefore any user can duplicate the results.

Firstly, in the code, library and variable definitions were created. There are three servo definitions and three servo angle definitions.

Secondly, in the setup() section, the servos were attached to their respective pins and the initiation angles to the servos are written. The servo engines initiate at 90 degrees. There is no user input that can keep the servos at 90 degrees, therefore the once the arm stands at a 90-degree angle, the user can understand that the system has initiated. Furthermore, a three-note music is played by the system once the serial opens, to make sure that the communication pathway between the Python hand recognition software and Arduino Mega is open.

Lastly, in the loop() section, a switch case was created. This switch case operates given the user input from the Python program. According to the number of fingers received from the Python program, different cases in switch are activated, which operates the robotic arm.

The Python hand recognition software works using OpenCV and Serial Port [5]. OpenCV proves rather useful when working with computer vision and image recognition due to wide variety of supported libraries and conducted experiments. OpenCV has more than 47.000 users and more than 14 million estimated downloads. Use extends from interactive art to mine inspection, to stitch maps on the web, or to advanced robots. [5].

When the software is initiated, a video stream from the camera of the PC (or webcam, in this system) is recorded and presented on the screen. Even though the entire video stream is presented on the screen, the part of the screen which processes the information is marked with a red square. Instead of detecting the entire video stream, the red square was chosen to be area of interest due to efficiency purposes. The program works more efficiently in a smaller area. The user must present his/her hand in this square for the software to process the information. Once the hand is presented in this square, the software is responsible for counting the empty spaces between the fingers. For example, if the software is counting 3 defects, there would be 4 fingers present in the square, which would trigger an action to the servos [6, 7].

The Python code can be seen alongside the closed system can be seen in Fig. 2.

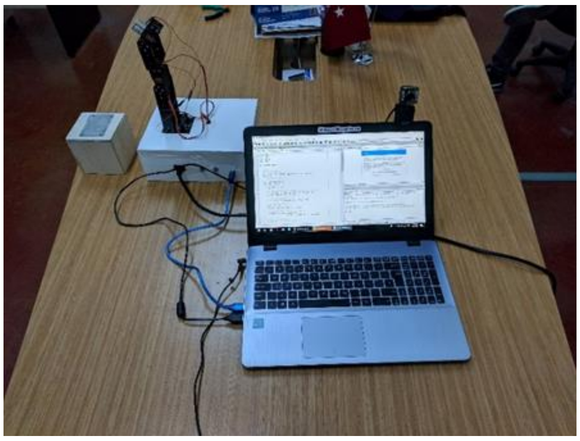

Fig. 2. Closed system along with the software.

3. Analysis Process On Image

In image processing, the thresholding method of the Otsu is used for the decision of the automatic thresholding level based on the shape of the histogram [19, 20]. The method of Otsu selects the threshold by minimizing the intra-class variation of two pixel groups separated by the thresholding operator [19, 20]. The aim is to select a point as the threshold value between two peaks representing the foreground and background pixel values [19, 20].

With the calculation below, this calculation depends on the histogram counting set [20]. Taking into account the probability $\omega 1$ on equation 1 and $\omega 1$ on equation 2;

$$\omega_0 = \sum_{i=0}^{k} Pi = \omega(k) \tag{1}$$

$$\omega_1 = \sum_{i=k+1}^{L} Pi = 1 - \omega_0(k) \tag{2}$$

Equation 3 above μ0, μ1 on Equation 4 and Equation 5 above ω (k) is calculated.

$$\mu_0 = \sum_{i=1}^{k} \frac{iPi}{\omega 0} = \frac{\mu(k)}{\omega(k)} \tag{3}$$

$$\mu_1 = \sum_{i=k+1}^{k} \frac{iPi}{\omega 1} = \frac{\mu T - \mu(k)}{1 - \omega(k)} \tag{4}$$

where

$$\omega(k) = \sum_{i-1}^{k} iPi \tag{5}$$

The average pixel value of the total image is calculated using μt equation 6.

$$\mu T = \sum_{i=1}^{L} iPi \tag{6}$$

Class variables to be used in calculations σ_0^2 ve σ_1^2 equation 7 and 8 is calculated by.

$$\sigma_0^2 = \sum_{i=1}^{k} (i - \mu 0)^2 \, pi / \omega_0 \tag{7}$$

$$\sigma_1^2 = \sum_{i=k+1}^{L} (i - \mu_1)^2 \, pi / \omega_1 \tag{8}$$

The Otsu algorithm mentions three classes significantly. These changes to the class λ, change between classes κ and total variance Ƞ with these variables are defined on equations 9, 10 and 11.

$$\lambda = \sigma_B^2 \tag{9}$$

$$\kappa = \sigma_T^2 / \sigma_W^2 \tag{10}$$

$$\eta = \sigma_B^2 / \sigma_T^2 \tag{11}$$

where

$$\sigma_W^2 = \omega_0 \, \sigma_0^2 + \omega_1 \, \sigma_1^2 \tag{12}$$

$$\sigma_B^2 = \omega_0 (\mu_0 - \mu_T)^2 + \omega_1 (\mu_1 - \mu_T)^2 = \omega_0 \, \omega_1 (\mu_1 - \mu_0)^2 \tag{13}$$

Equation 12 and 13 on σ^2 ve σ_B^2 formulas where the variables are defined are shown. Otsu states that when any of these criteria are maximized, the others are equivalent to maximizing. Equation 11 on definition Ƞ larger threshold value according to the selected equation can be re-displayed on the articles 13 σ_B^2 's is the same as maximizing.

$$\sigma_B^2(k) = \frac{[\mu T \omega(k) - \mu(k)]^2}{\omega(k)[1 - \omega(k)]} \tag{14}$$

The formula shown on Equation 14 is the most important calculation process on the Otsu algorithm. For all possible threshold values with this formula σ_B^2 calculated. And the maximum value that makes it the threshold value is selected.

The contours are described as a curve that combines all the continuous points of the same color or density along the boundary [18]. The contours are used for shape analysis and object detection and recognition [18]. The contour is the curve of two variable functions in which the function has a fixed value. A contour combines points above a certain level and an equal height [21].

Convex hull, concave polygons are traded on a defined region. The purpose is to find the smallest convex shape that surrounds the concave shape. This algorithm has been used to scan all points in the set of contour points within the system. A drawing process is performed around the contour of the hand to create a convex body by scanning all contour points [5, 21, 22].

4. Experiments

Upon completing the system, experiments were conducted in order to see the movement of the robotic arm. The system was successful when recognizing the number of fingers shown to the camera. The recognized number of fingers seen by the camera were written on the Python screen. The different results acquired from the camera image can be seen in the pictures below. Fig. 3, 4, 5, 6 and 7 show 1, 2, 3, 4 and 5 fingers respectively.

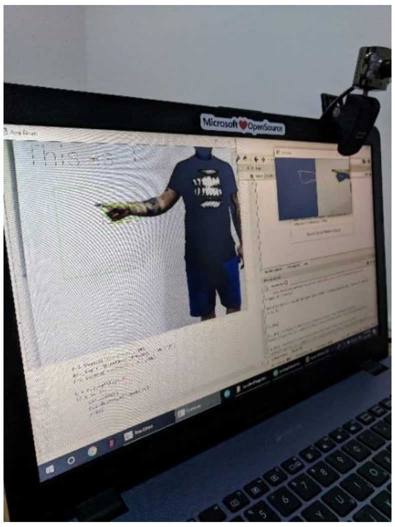

Fig. 3. Results showing 1 finger.

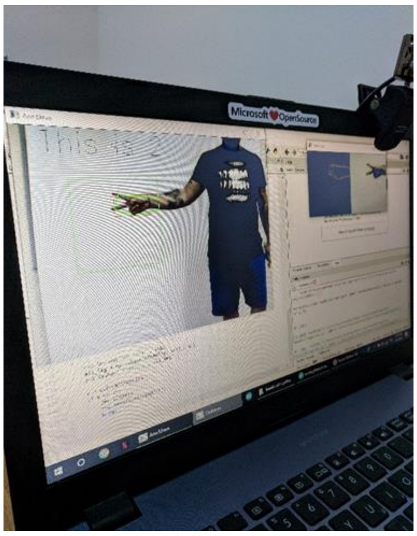

Fig. 4. Results showing 2 fingers.

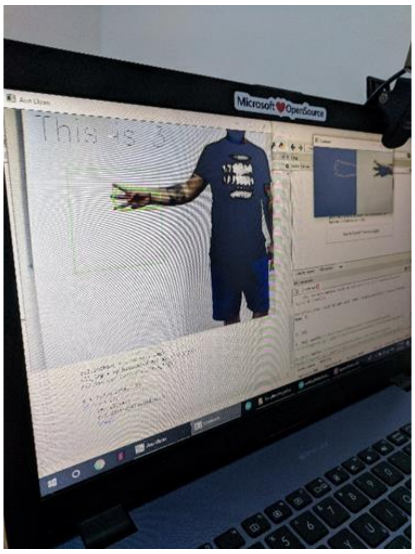

Fig. 5. Results showing 3 fingers.

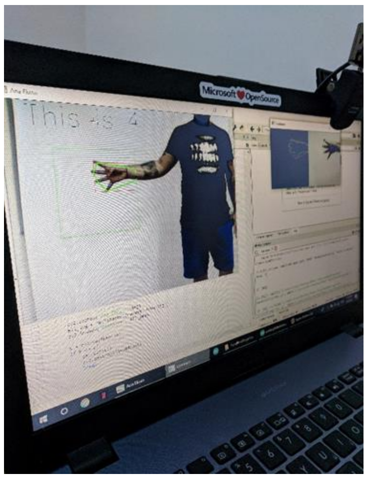

Fig. 6. Results showing 4 fingers.

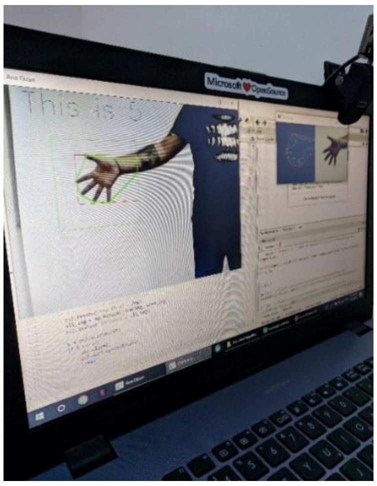

Fig. 7. Results showing 5 fingers.

During the experiments there were two potential problems that could diminish the results produced by the robot. If the background behind the red square in the software of polluted, the results were incorrect. Secondly, if the user does not leave adequate space between their fingers, the software often does not recognize the spaces between fingers. If these circumstances are satisfied, the results are successful, and the robotic arm movement is established.

5. Conclusions

As a result, as the distance between the webcam and the image increased, the hand gestures on the background could not read the software clearly, and incorrect operations were encountered. In addition, if the angle between the fingers is small, it has been seen that it defines the movement wrongly. The results are shown in Figures 3, 4, 5, 6 and 7.

6. Discussion

The aim of this study is to be able to detect finger movements of a person with instant camera images and perform defined operations on embedded systems. In the experiments, errors were detected in capturing the image depending on the distance between the background and the camera. Camera resolution is required to correct this error. The light level of the environment can be changed according to the experiment.

Acknowledgements

Many thanks to Istanbul Gelisim University and Technology Transfer Office for their support.

References

[1] E. B. Mathew, D. Khanduja, B. Sapra, B. Bhushan, Robotic arm control through human arm movement detection using potentiometers, International Conference on Recent Developments in Control, Automation and Power Engineering, 2015.

[2] B. İşçimen, H. Atasoy, Y. Kutlu, S. Yıldırım, E. Yıldırım, Smart robot arm motion using computer vision, Elektronika Ir Elektrotechnika, ISSN 1392-1215.

[3] M. A. Jayaram, H. Fleyeh, Convex hulls in image processing: a scoping review, American Journal of Intelligent Systems, 2016.

[4] OpenCV library document, https://opencv.org/

[5] Structural Analysis and Shape Descriptors, OpenCV "2.4.13.7 documentation". https://docs.opencv.org

[6] A. Dhawan, A. Bhat, S. Sharma, H. K. Kaura, Automated robot with object recognition and handling features, International Journal of Electronics and Computer Science Engineering, ISSN 2277-1956/V2N3-861-873.

[7] Abhishek Chavan, Abhishek Bhuskute, Anmol Jain, Dynamics of robotic arm, International Journal of Computer Applications (0975 – 8887), 2014.

[8] C. Manresa, J. Varona, R. Mas, F. J. Perales, ‘‘Hand tracking and gesture recognition for human-computer interaction’’, Electronic Letters on Computer Vision and Image Analysis 5(3):96-104, 2005.

[9] P. Xu, A real-time hand gesture recognition and human-computer interaction system, arXiv:1704.07296v1 [cs.CV] 24 Apr 2017.

[10] A. Soetedjo, I.K. Somawirata, A. Irawan, ‘‘Human arm movement detection using low-cost sensors for controlling robotic arm’’, Journal of Telecommunication, Electronic and Computer Engineering, e-ISSN: 2289-8131 Vol. 10 No. 2-3.

[11] A. Alam, T. Rana, M. Hashemy, An autonomous detective robotic arm, International Conference on Mechanical, Industrial and Materials Engineering 2017.