# Red Dot Determination with 6 Axis ABB Robot Arm

Güzin Tirkeş [1*], Esra Nur Odabaşı [2], Simge Aytar [2], Gözde Nur Adışanlı [2], İbrahim Turan Bastacıoğlu [3]

[1]*Atılım University, School of Engineering, Computer Engineering, 06830, Ankara, Turkey*
[2]*Atılım University, Faculty of Engineering, Computer Engineering, 06830, Ankara, Turkey*
[3]*Atılım University, Faculty of Engineering, Information Systems Engineering, 06830, Ankara, Turkey*

***Abstract*— The general aim of this study is to determine the positions of the red points on the surface using techniques of real-time image processing via the camera and user interface mounted on the 6-axis robot arm. The robot arm movement is performed in the simulation environment ABB RobotStudio. In this study, 2D Webcam was used in the redpoint detection process. Detection of the red point is done via the interface in MATLAB, depending on three axes. Using the robot's current position and the image it receives from the camera, the interface detects the correct place of the redpoint. This process was made using the prepared mathematical formula according to the camera's position. The X, Y, and Z coordinates of the redpoint can be found on flat surfaces of all shapes with developed software. The goal of this study is to prepare a redpoint detection environment based on six axes.**

***Keywords*— Functional point analysis, Image Processing, MATLAB, Robot programming**

## I. INTRODUCTION

With developing technology, machine power has started to be preferred instead of human resources. Therefore, the robotics industry is growing and improving every day. The robot and robot arms are widely used in every field due to the fast completion of the work and the low error rate. With the widespread use of robots, the need for new features has increased.

In this study, red points on the surface are detected during the surface using a 6-axis robot arm. While the robot arm moves on the surface, the redpoint can be seen with the developed camera system, and its coordinates on the camera can be obtained. The robot arm's coordinates were located on the surface are taken from the simulation environment designed in RobotStudio instead of the physical environment. The camera coordinates taken from the camera, and the robot coordinates taken from the simulation are entered into the developed MATLAB interface to determine the red point coordinates' real location on the surface.

The most significant feature that separates the work done from other work in this field is that the robot's coordinates were taken from the created simulation environment. Thus, this process can be done without the robot arm.

Manuscript received July 23, 2020; accepted December 31, 2020.
*Corresponding author: guzin.tirkes@atilim.edu.tr

## II. LITERATURE REVIEW

Growing and developing world countries for products produced with developing industry technology, looking for high quality, low cost, and low time parameters. Therefore, machine power is preferred over the workforce for many industrial applications. In today's industrial applications, robots and robot arms are widely used due to their features such as high load amount, continuous operation, and low error. This study aims to detect the red points on the flat surface by using a six-axis robot arm and camera. Real-time image processing techniques are used to detect the redpoint. The camera fixed to the 6th axis must always be perpendicular to the plane. In this way, the camera gives the correct coordinate. Studies based on research in this field were examined. The important ones of these studies are given below. The literature review conducted is classified according to various categories. The studies discussed in the literature review are shown in Table 1.

In the study of Yalım, Sinekoğlu, and Kurt, they indicate to make a robot manipulator controlled by Arduino, a family of microprocessors, and to give the robot manipulator the ability to process images using OpenCV library and finding where or at which coordinate the transaction is performed a critical problem. Image processing techniques can solve this problem. The robotic arm finds the objects and puts them in a box that we specify near the platform. The coordinates of the items on the platform were determined by a camera standing on the platform. The photo is saved with a button. The recorded image was then processed with the written codes, and the center coordinates of the objects are located on the Raspberry Pi microcomputer. Raspberry Pi sends these coordinates to the microprocessor Arduino, which provides robot control. Arduino takes the object by controlling the robot arm. With this project, objects can be moved from one place to another. The image processing parts of the project will help detect the locations of the objects [1].

Erdoğan, Şimşek, and Kapusız indicate to separate the colors by processing images through MATLAB using a robot with 4-axis mobility. Their study contains information about the electrical circuits, mechanical structures, and software of a robot arm used in automation systems. The project is an experimental study and was carried out to move the robot arm by adjusting

the place's angular values where the material should be moved according to the color detected by MATLAB and defined by the MATLAB. Arduino Uno with Atmel AVR microcontroller was used as a microcontroller in the project [2].

TABLE I. RESEARCH DETAILS

| Authors | Research Topics | Department | Focus | Approach Details |
|---|---|---|---|---|
| Yalım B., Sinekoğlu Y. E., Kurt A. T. | Image Processing Based 4+1 Axis Robot Arm Design | Electrical Electronics Engineering | Robot Design and Image Processing | 4+1 Axis Robot and OpenCV |
| Erdoğan E., Şimşek K., Kapusız B. | Examine of Image Processing Techniques and Applications in Robots | Mechatronic Engineering | Robot Design and Image Processing | 4 Axis Robot and MATLAB |
| Yağımlı M., Varol H. S. | Real-Time Color Composition Based Moving Target Recognition | Mechatronics Engineering, Electronics and Communication Engineering | Real-Time Detection of Moving Targets | - |
| Çubukçu A., Kuncan M., İmren M., Erol F., Ertunç H.M., Öztürk S., Kaplan K. | Object Discrimination and Sorting With İmage Processing on 3-Axis Robot Mechanism | Mechatronic Engineering, Electronics and Communication Engineering | Robot Systems and Image Processing | 3 Axis Robot and MATLAB |
| Şenel F. A., Çetişli B. | Object Control on Production Line With İmage Processing and Five Axis Robot Arm | Computer Engineering | Robot Systems and Image Processing | 5 Axis Robot Arm and OpenCV |
| Aytan A. E., Öztürk Y., Örgev E. K. | Image Processing | Faculty of Dentistry, Computer Engineering | Image Processing | - |
| İşçimen B., Atasoy H., Kutlu Y., Yıldırım S., Yıldırım E. | Robot Arm Application Using Computer Vision and Gradient Descent Algorithm | Computer Engineering | Computer Vision and Gradient Descent Algorithm | 3 Axis Robot Arm and Computer Vision |
| Vijayalaxmi P. B., Putta R., Shinde G., Lohani P. | Object Detection Using Image Processing for An Industrial Robot | Computer Engineering | Industrial Robot and Image Processing | Pick and Place Robot And MATLAB |
| Özgenel Ç.F. | Image Processing Methods Used in Autonomous Crack Detection and Technology's Potentials | Architecture | Autonomous Crack Detection and Image Processing | - |
| Yılmaz V., Güngör O., Kadıoğulları A.İ. | Determination of Height, Crown Diameter, and Crown Volume of Trees Using Image Processing Techniques | Map Engineering, Forest Engineering | Stereo Image and Image Processing | - |
| Shaha S. K., Mishra R., Prasannaa D., Mohapatroa G. | Fabrication and Deviation Analysis of a 5-Axis Robotic Arm Using Image Processing Technique | Mechanical Engineering | Fabrication and Deviation Analysis and Image Processing | 5-Axis Robot Arm and MATLAB |
| Kalaiarasan G., Thomas A., Kirubha C., Saravanan B. | Development of Edge Processing and Drawing Algorithms for a 6 Axis Industrial Robot | Mechanical Engineering | Industrial Robot and Image Processing | 6-Axis Industrial Robot Arm, ABB RobotStudio, and Computer Vision |
| Cuşkun Y., Duman F., Basık H., Gün F., Kaplan K., Ertunç H. M. | Image Processing Based Multi-Purpose 4-Axis Robot Mechanism | Mechatronic Engineering | Industrial Robot and Image Processing | 4-Axis Robot Arm and MATLAB |
| Soans R. V., Hegde A., Kumar A., Nittul R., Singh C. | Object Tracking Robot Using Adaptive Color Thresholding | - | Image Processing | Robotic Arm and MATLAB |
| Adar N. G., Kozan R. | Vision-Based Control of 5 DOF Robotic Arm | Mechanical Engineering | Robot Systems and Image Processing | 5 DOF Robotic Arm and MATLAB |
| Dragusu M., Mihalache A. N., Solea R. | Practical Applications for Robotic Arms Using Image Processing | - | Computer Vision, Image Processing and Algorithms | Robotic Arm, OpenCV, and MATLAB |

Yağımlı and Varol have conducted a study to develop a software that recognizes moving targets by processing the images from the camera. The target recognition system realized compares pixels between the camera's target images and the target images previously recorded on the computer. While high-resolution cameras provide advantages in image transfer, low-resolution cameras will benefit the system's speed. Using high speed or parallel processor computers will also increase the speed of the system. The target has been determined and recognized, and the process of monitoring this target was considered. This system can be realized by integrating a robot or robot arm without the need for an operator [3].

In another study, Çubukçu et al. indicate to create the text entered through the interface with a camera mounted on a three-axis robot mechanism with letters/numbers on a black background. After the letters/numbers in the word are determined, they compared the word with the data in the database, and the position of the letters/numbers is determined. Positioning was done by using image processing methods using the MATLAB environment.

The image taken by the camera was transferred to the MATLAB Guide interface program. In the MATLAB environment, using the image processing methods, letters/numbers in the word entered on the interface were defined, and the text was re-created with a 3-axis robot. There was a USB camera on the system, and the object images were transferred to the system using this camera. Under normal conditions, the system works successfully, but placing letters irregularly on a black ground and changing the light intensity may cause the robot to go wrong. When the light creates reflections in the work area, the camera perceives the reflection as an object even though there is no object and causes the robot to go to the wrong position. Another reason is that the system does not perceive the small values in the pixel PPS coefficient conversion. All these situations were identified as deficiencies in the study [4].

Şenel and Çetişli, in their study, indicate to detect the products in a production facility through embedded systems and image processing a robot arm. As a result of this study, a real-time image processing application has been developed. The imaging system used in the research was placed in a closed box designed without light. In the closed box, the USB WEB camera was placed with a bird's eye view from above. The images taken from the camera were first passed through a color filter, then only the colors that were the focus of interest remained in the image. According to the point received as a reference in the study, it is specified in advance which engine will move the robot arm to the product pick and drop points. Images were taken from a camera connected to the embedded system, subjected to image processing techniques, and products moving on the production line were classified. When the examined products are faulty, the necessary commands are sent to the robot arm attached to the embedded system. The faulty product is taken from the production line and placed in the faulty product basket. In the studies conducted, a total of 60 products containing two types of objects were classified with a success rate of 100% [5].

In another study, Aytan, Öztürk, and Örgev explain how to perform linear and angular digital images with image processing and image processing methods. Image processing was dependent on and related to the change and analysis of pictorial information in its general position. The primary purpose was to visually reinforce an image that is not in its original shape and statistically evaluate it. Image processing can be examined under three main headings. The first was optical, the second was analog, and the third was digital [6].

In the study of İşçimen, Atasoy, Kutlu, Yıldırım, and Yıldırım, a smart robot arm application that sees, finds, recognizes, and performs the task has been implemented by combining computer vision and robot arm application. For this purpose, a smart robot arm has been designed to recognize the materials used in the foodservice and arrange or collect them in the service order. A new database has been created by collecting pictures of the materials used in the foodservice. Classifying and labeling the materials in the developed system database using image processing techniques sends coordinates to the robot arm. KNN classifier was used for classification of the data, and 90% success was obtained. Then, the robot arm joint angles were calculated by the gradient descent method, and its movement was achieved [7].

Vijayalaxmi, Putta, Shinde, and Lohani analyzed robots for pick-and-place applications using image processing to detect the fully desired object and place it at the selected location. The color detection capability was enhanced by using high-speed processers and low-cost cameras that use little power. MATLAB was used for GUI and for processing the image acquired. A new software RoboCIM has been used, which has an inbuilt program that controls the robot. The coordinates of the object were obtained through the camera and were edited in the program. TTL concept has been used for faster communication of the control signals. Servo motors were used, which play a vital role in many industries and increase the task's accuracy in the logistic and packaging industry [8].

In his study, Özgenel used computer vision methods to detect cracks in buildings using 3D reconstruction methods [9].

Yılmaz, Güngör, and Kadıoğulları studied the heights of the trees by using the point clouds produced with the stereo images provided from the General Command of Mapping. A MATLAB script has also been developed to determine the top diameters and crown of the trees automatically. Hough transformation, which is used for feature extraction, has been used to determine the trees' maximum diameter and height. As a result, the trees in the study area have been detected with more than 90% [10].

In the study of Shah, Mishra, Prasanna, and Mohapatro, few conclusions were drawn after completing the experiment and discussing the results. A robotic model with five joints and an end needle at the end effectors position was developed. The experimental model was fabricated using actuators with the inbuilt position, velocity and torque sensor, and a closed-loop system controller. The robotic arm can travel to any coordinate point specified by the user within its workspace by merely inputting the MATLAB software's coordinate points. Inverse kinematics and forward kinematics were also used to originate the equation for the robotic arm. Testing was done with a robotic arm by taking arbitrary coordinate points on the apple's surface as the target point within the workspace of the robotic arm. The Image processing method was used to find the deviation in reaching the target point to verify the results. The level of variation found in the new model was at the level of 0.1 mm to 0.32 mm. [11].

Kalaiarasan et al. indicated that the edge processing algorithm and drawing algorithm were developed using an industrial robot (ABB IRB 1410). RAPID programming language and RobotStudio simulator were used in the expanded program. The IRC-5 controller controlled all movements and controls of the robot manipulator. This experimental work enables the development of an edge processing algorithm to convert the 3D image to a contrast 2D image using Lab VIEW and the drawing algorithm's development to transform the robot into motion coordinates. The robot can draw every

pixel detected in the edge processing algorithm and reproduce it in more than one copy simultaneously. The image was captured in real-time from the camera using the Direct Acquisition method. The edges of this image from the camera are detected and displayed. In the edge detection process, the image file path was taken from the directory. According to the developed algorithm, only images in jpeg and BMP formats can be read. After the images are defined, they were converted to 32-bit and 8-bit gray images, and the threshold value is changed. Depending on the predefined starting point, the system develops the image's x and y edge coordinates. The drawing was made by transferring the coordinate value to the manipulator to allow the manipulator to move from point to point. The study was necessary to set the light well to perceive the camera [12] entirely.

Coşkun et al. indicated that the word written in the MATLAB interface program was created with scattered letters. The captured image was first converted to gray tone and then converted to black and white by applying a particular threshold value for matching. Besides, geometric shapes and objects with different colors were distinguished with the options available in the MATLAB program and grouped and ordered according to the user's choice. During this process, the coordinates of the geometric shape in the selected color were determined in pixels. These pixel coordinates were converted into motor rotation information with a specific transformation before being transferred to the engines. The plates' position and angle with different features were also calculated in a MATLAB environment with image processing methods, and a 4-axis robot transported the plates. The determined plates were then taken with the vacuum holder and placed on the ground at zero degrees and controlled by the Arduino UNO card. The robot moves to the specified location by PLC - MATLAB communication. In the study, the images were provided by the camera connected to the computer, and the location of the camera was located at the height of 120 cm from the setup, just in the middle of the work area. In practice, the process of printing the words created from numbers and letters on the ground with the robot was achieved with a success of nearly 90% in straight letters [13].

Soans et al. designed a prototype that detects an object with different properties, monitors, and generates a control signal using image processing techniques. The image was segmented from its original resolution using MATLAB. In the next step, the object of interest was detected using color thresholding. Due to the extreme light conditions, noise occurs in the image, and morphological processes were applied to remove this noise. Then, the threshold mask was created to detect the required color in MATLAB. Once the color is detected, the center of gravity is drawn and then a bounding box is drawn around the object [14].

In Adar and Kozan's study, they indicate that a robot arm with a structure similar to a human arm has been designed. The designed robot arm has a total of five degrees of freedom. Real-time image processing techniques were used with the Matlab-Simulink program. "Video device" block has been used to get data from the web camera connected to the computer with Matlab-Simulink "Image Processing Toolbox". In this block, the RGB video format and 640x480 size are selected. The image taken from the video camera was passed through the appropriate threshold value, and a black and white image was obtained. Then, an opening morphological process is applied to the image to get a better-quality image. The center and area of the object were calculated from the black and white image obtained. With the computed equation, the distance of the object to the camera is calculated mathematically. The object is kept at a fixed point in the camera field of view. The x-y-z coordinates of the object were calculated with the image processing block. These coordinates were sent to the block where the inverse kinematics of the robot arm was calculated, and the angular displacement of each corresponding joint was calculated. These calculated values are sent to the motors by making appropriate conversions, and the arm is allowed to move to the point where the object is located [15].

The study of Dragusu, Mihalache, and Solea indicates that image processing applications are carried out with a robot arm. These applications are made with a video camera that captures the images to be processed for recognition, sequencing, and extraction of contour coordinates. The designed system processes the images captured by the webcam, extracts essential information, and sends it to the robotic arm. Image processing techniques are applied to the image captured with the software developed, using the OpenCV library. The canny method, one of the edge detection techniques, is applied to the image. This method is preferred because it is successful in noisy images. The coordinates extracted from this image are rescaled in the polar system. Then these coordinates are shown in a MATLAB simulation. For the applied drawing algorithm, the contour extraction algorithm's coordinates are converted from the Cartesian system to the Polar coordinate system and sent to the robotic arm [16].

As a result of our literature reviews, the studies on 3, 4, 5, and 6 axis robot arms were examined. In the studies examined, the researchers carried out their tasks using the 4 and 4 + 1 axis robots they designed and industrial 3, 4, 5, and 6 axis robots. In these studies, MATLAB and OpenCV were mostly used for image processing. The image processing techniques generally applied in the studies are similar, and the distance between the camera and the ground has been kept constant. One of the common problems experienced in the studies was insufficient and variable light level. The light level of the environment to be processed should be stable and sufficient. Studies have shown that better results are obtained with stereo images in image processing, and better results have been obtained with stereo cameras in multidimensional studies. However, as stereo cameras' price was much more expensive than webcams, webcams in image processing are more preferred. Since the webcams provide a 2-dimensional image, pixel conversion has used when working in 3 or more axes. As a result of the literature review, the images taken from the camera using the camera and MATLAB are processed

with image processing techniques, and the coordinates of the redpoint are on three axes. The difference in this study is that the movement of the robot arm was done in the simulation environment.

## III. IMPLEMENTATION

In this project, image processing was done by using an ABB robot and camera. The camera is fixed on the end-point of the ABB robot. In this way, the camera would detect the redpoints on the grounds of different sizes using the robot arm's motion. The RobotStudio program developed by ABB for its robot users was used to create the environment where the robot will travel and ensure the robot's movement on the ground.

The robot moves 200 mm above the ground with a camera that has real-time image processing features. Since the camera perceives the image in real-time, it continuously takes the image and transfers it to the developed system. Therefore, as soon as the camera detects the redpoint, the user must first stop the robot and then the image. The user needs the three-dimensional X, Y, and Z position when the robot is stopped, and the X and Y coordinate on the camera image to access the coordinate of the redpoint on the ground. When the user enters the robot's coordinates and the redpoint's coordinates taken from the camera into the interface that works according to the developed algorithm, the real X, Y, and Z coordinates of the redpoint on the ground are determined.

Using the RobotStudio simulation program, environments similar to the physical environment were created, and the robot coordinates to be taken from the robot in the physical environment were taken from the simulation environment. Thus, the coordinates of the redpoint on the real ground were found. The camera and interface system were developed in the MATLAB environment as initially planned.

### *A. Simulation Environment*

With the designed system, the camera can detect redpoints on all 3-dimensional grounds that the robot arm can reach. The system can operate at different ground measures. Therefore, two simulation environments have been designed in RobotStudio. In both environments designed in the simulation environment, the robot closest to the real robot named IRB1600_6_145__02 was used. In both simulation environments, the symbolic camera is placed on the robot to symbolize the camera moving with the robot. The simulation environments designed are shown in Figure 1 and Figure 2.

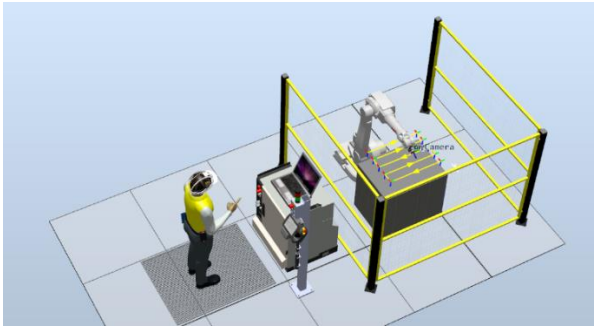

Fig. 1. Simulation Environment 1

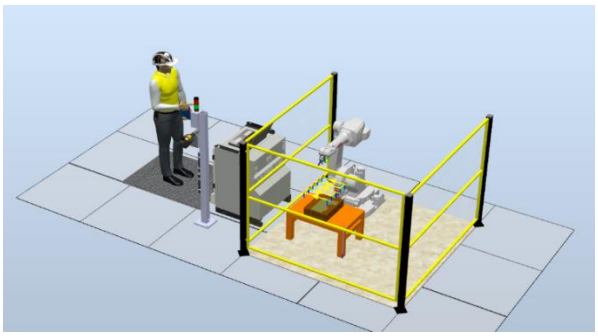

Fig. 2. Simulation Environment 2

The user must determine a path for ABB robots to act in a planned way. Users can either do this operation with the hand terminator next to the robot or design their path in the RobotStudio and transfer it to the robot. Therefore, in both simulation environments, a path was drawn according to the ground measures to enable the robot to navigate the ground in a planned manner. Since the project's camera is programmed assuming that it will look at the base 200 mm from above at a right and fixed angle, the paths were drawn 200 mm above the ground.

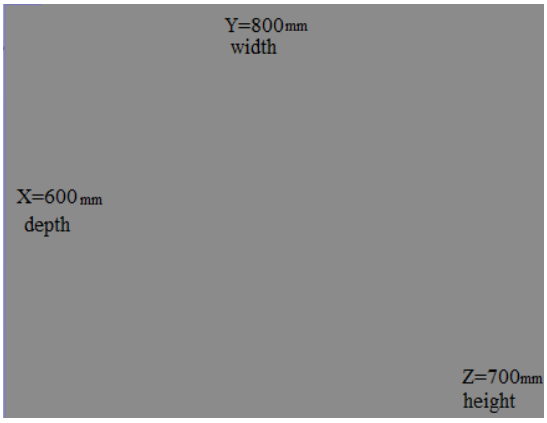


Fig. 3. 1st Ground Measures

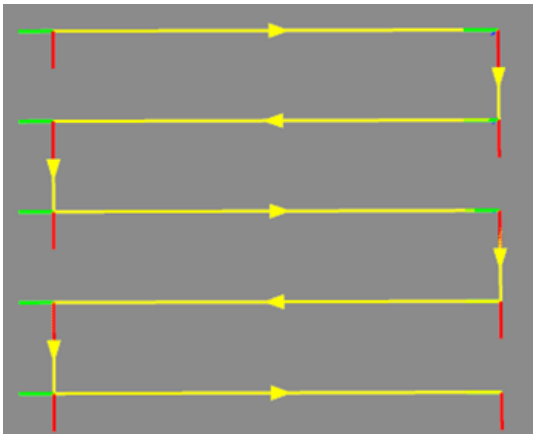

Fig. 4. 1st Ground Path

Figure 3 shows the 1st ground measures, and Figure 4 shows the path drawn for the 1st ground.

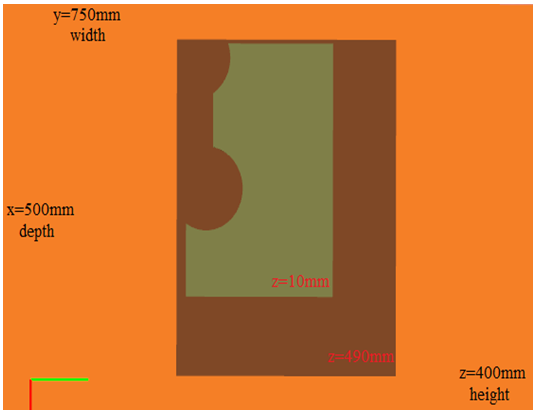


Fig. 5. 2nd Ground Measures

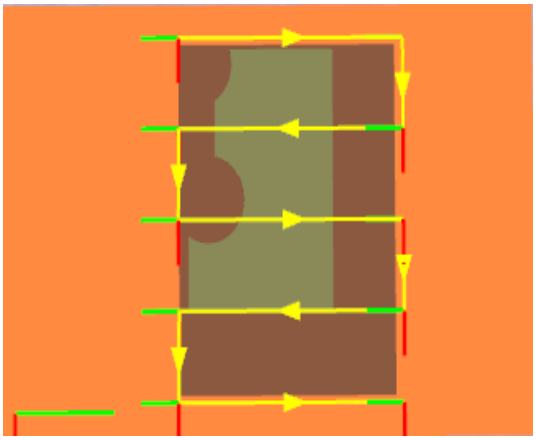

Fig. 6. 2nd Ground Path

Figure 5 shows the 2nd ground measures, and Figure 6 shows the path drawn for the 2nd ground.

After the paths in Figures 4 and 6 are drawn in the simulation environment, they are converted to RAPID code in the simulation environment. This converting process is done with the synchronization button in RobotStudio, and the robot travels the ground in the direction of the path drawn. With these paths marked, the camera fixed on the end-point of the robot arm follows these paths and travels the ground. The user can stop the robot at its desired position during the robot arm's ground travel and access the robot's current X, Y, and Z coordinates shown in Figure 7.

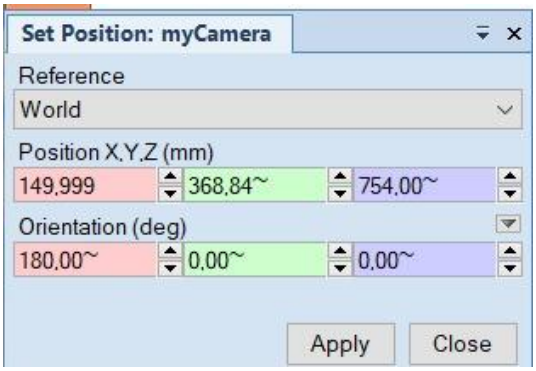


Fig. 7. Robot Position

*B. Camera and MATLAB System*

The MATLAB program has been used to do real-time image processing with the camera. Besides, Image Acquisition Toolbox and Computer Vision Toolbox were used in MATLAB.

First, getting the necessary information about the camera hardware, extracting the camera's features from the hardware information, getting the memory information of the computer, making all the available memory available for image processing operations, and creating the video object according to the camera features are provided respectively.

It is then provided to get the camera's name, information, and resolution values according to the hardware information with a function to identify the camera in MATLAB. The resolution value of the camera was set to 480x640 pixels. Since the camera does real-time image processing, it instantly detects the redpoint's position on the video image. Therefore, the video loop of the camera code consists of a single and infinite while loop. As soon as the user stops the image, the loop is exited.

The necessary image processing techniques are used for redpoint detection in the while loop. Respectively, the screenshot is taken from the video, the grey layer is extracting from the red image layer, and a 9x9 Median filter is applied. Then, the colors are extracted according to the threshold value, and the red objects found are displayed on the screen, as painted in Figure 8.


Fig. 8. Showing Redpoint

The image where the location and area of the redpoint are detected is given in Figure 8. The long edge of the camera represents the Y-axis, and the short edge represents the X-axis. To ensure that the camera image's axes are in the same direction as the ground in the simulation environment. The Centroid function used for locating the redpoint determines the redpoint's center coordinates according to the pixel values. The Y-axis is 640 pixels, and the X-axis is 480 pixels in length. However, since the ground measurements are in mm in the simulation environment, these pixel values should be converted to mm. The most crucial rule to achieve this transformation is that the camera always faces a fixed height and right angle. When the camera looks at the ground at a fixed height and right angles, although it only

detects in two dimensions, it allows the Z-axis to work with the correct values.

The image in Figure 8 is the view of the camera 200mm above the ground. The camera is looking at the ground at the right angle. Based on these conditions, the Y-axis was measured as 160 mm and the X-axis as 120 mm. As a result, the ratio between the X and Y axis's pixel values and the mm values was 1/4, the ratio between the pixel values of the X and Y axis and the mm square values was 1/16 found. The ratios found were used as multipliers in printing the coordinates and the area in pixels on the screen where the coordinates in mm and the area in mm square were displayed on the screen. Throughout the project, it was studied based on the camera traveling 200mm above ground with the right angle; however, in line with the new measurements made for different camera heights, transformation can be provided with new rates.

### *C. Red Point Detection System*

Red Point Detection System is the user interface developed in MATLAB. This system calculates the redpoint's real coordinates on the ground according to the algorithm developed by taking the robot coordinates and camera coordinates from the user. It shows the calculated coordinates to the user through the user interface.

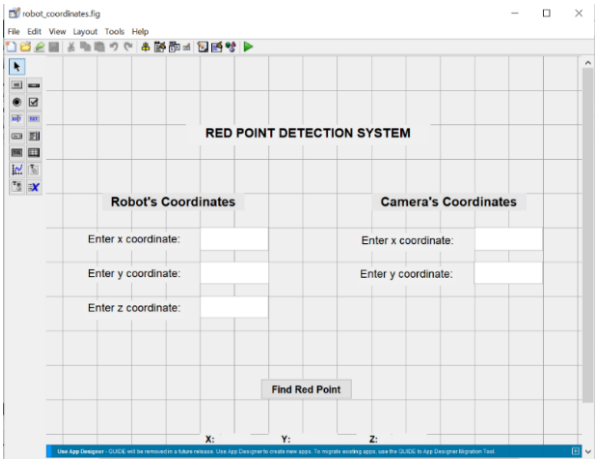


Fig. 9. MATLAB Guide

The user interface is created by designing through MATLAB GUIDE, shown in Figure 9. The interface is connected with codes on the Inspector screen in MATLAB GUIDE. The codes of the interface are also created. At the interface designed, the user gets the coordinates through TextBoxes. The coordinates received by the TextBoxes are developed by the algorithm developed in the findRedPointCoordinate_Callback() function and displayed on the user interface screen.

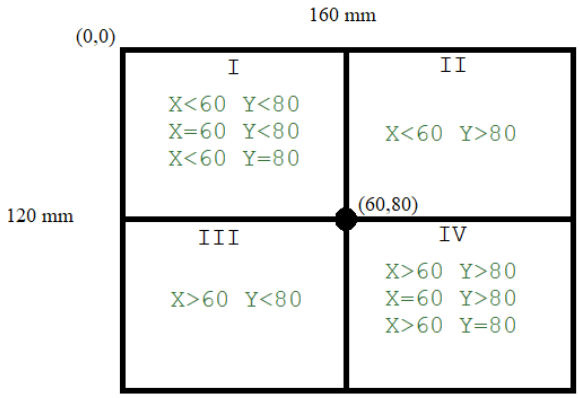


Fig. 10. Camera View and Regions

In the algorithm developed for the redpoint location, the location is determined according to the image of the camera. The measures and center point of the camera image are given in Figure 10. The X coordinate of the center point on the camera image is 60 mm, and the Y coordinate is 80 mm. When the user stops the robot arm and camera image, the robot arm's location is the same point as the center point of the camera image. In this way, the center point of the camera image on the ground is found. The real coordinate of the detected redpoint on the ground is seen from this point.

To find the redpoint's real location, it is necessary to know exactly which side of the center point lies. Therefore, it is assumed that the screen image is divided into four equal regions according to the intervals given in Figure 10. According to the camera image coordinates, the absolute distances between the X and Y coordinates of the redpoint and the center point are found. These distance values are added to or subtracted from the robot's coordinates, depending on the region where the redpoint is located. In this way, the real coordinates of the redpoint on the ground are obtained.

The reallocation of the redpoint is found by making the following calculations.

$$
\begin{aligned}
&f(camera\ coordinate\ X, camera\ coordinate\ Y) = \\
&\{ if\ camera\ coordinate\ Y \leq \\
&medpoint\ Y\ \&\&camera\ coordinate\ X \leq \\
&medpoint\ X, \\
&red\ point\ Y = robot\ coordinate\ Y - |medpoint\ Y \\
&\qquad - camera\ coordinate\ Y| \\
&red\ point\ X = robot\ coordinate\ X - |medpoint\ X \\
&\qquad - camera\ coordinate\ X| \} \\
&\{ if\ camera\ coordinate\ Y > \\
&medpoint\ Y\ \&\&camera\ coordinate\ X < \\
&medpoint\ X, \\
&red\ point\ Y = robot\ coordinate\ Y + |medpoint\ Y \\
&\qquad - camera\ coordinate\ Y| \\
&red\ point\ X = robot\ coordinate\ X - |medpoint\ X \\
&\qquad - camera\ coordinate\ X| \} \qquad (1) \\
&\{ if\ camera\ coordinate\ Y \\
&< medpoint\ Y\ \&\&camera\ coordinate\ X \\
&> medpoint\ X, \\
&red\ point\ Y = robot\ coordinate\ Y - |medpoint\ Y \\
&\qquad - camera\ coordinate\ Y| \\
&red\ point\ X = robot\ coordinate\ X + |medpoint\ X \\
&\qquad - camera\ coordinate\ X| \} \\
&\{ if\ camera\ coordinate\ Y \\
&\geq medpoint\ Y\ \&\&camera\ coordinate\ X \\
&\geq medpoint\ X, \\
&red\ point\ Y = robot\ coordinate\ Y + |medpoint\ Y \\
&\qquad - camera\ coordinate\ Y| \\
&red\ point\ X = robot\ coordinate\ X + |medpoint\ X \\
&\qquad - camera\ coordinate\ X| \}
\end{aligned}
$$

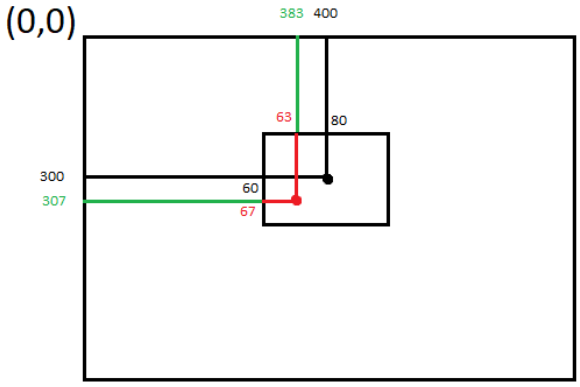


Fig. 11. Redpoint Example

According to these calculations, an example of a redpoint is given in Figure 11. The large rectangle represents the ground; the small rectangle represents a camera image. In this point example, the redpoint is located in the 3rd region on the screen display, and the X coordinate is 67 mm, the Y coordinate is 63 mm. The center point is the X coordinate 300 mm and the Y coordinate 400 mm on the robot arm. These coordinates taken from the robot corresponds to the midpoint of the camera. According to the calculations for the 3rd region, the real X coordinate of the redpoint was 307 mm, and the real Y coordinate was 383 mm.

The distance between the camera screen and the ground is 200mm, and the length of the symbolic camera connected to the end-point of the robot arm in the simulation environment is 54mm. Since calculations are made by keeping the distance between the robot and the ground constant, the Z-axis coordinate of the redpoint is calculated separately from other coordinates. The Z-axis coordinate of the redpoint is calculated by subtracting 254mm from the Z-axis coordinate of the robot arm.

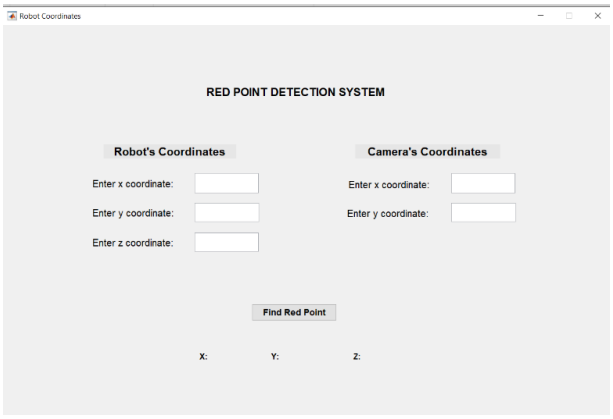


Fig. 12. Redpoint Detection System

The user enters the coordinates taken from the robot's simulation environment into the Redpoint Detection System shown in Figure 12 and pressing the Find Redpoint button, and reaches the X, Y, and Z coordinates of the redpoint.

## IV. Conclusions

The user can detect the redpoint on the surface through the redpoint detection system and move the robot in the simulation environment. The user can view the redpoint's area through the camera and the distance between the redpoint's new location and the old position in the moving camera. The user can also take a photo of the redpoint displayed on the screen and save it. This developed software has been observed to work correctly on different surface sizes due to the tests carried out. 3-axis redpoint detection is a preliminary study. The study's future goals are to integrate the camera into the robot arm and detect the red point by detecting the image depending on 6-axis. Future studies will be continued projects by making crack detection in metal forming.

## Acknowledgment

We offer our most profound respect and gratitude to our valuable Dr. Süha TİRKEŞ, who is always with us in all problems we encounter during the research and development process for all their help and contributions. Part of this study was presented orally in the Third International Conference on Data Science and Applications 2020 (ICONDATA'20).

## References

[1] Yalım B., Sinekoğlu Y. E., Kurt A. T. "Görüntü İşleme Tabanlı 4+1 Eksen Robot Kol Tasarımı". Undergraduate Thesis, Sakarya University. Sakarya, Turkey, 2016.

[2] Erdoğan E., Şimşek K., Kapusız B. "Robotlarda Görüntü İşleme Teknikleri Ve Uygulanmalarının İncelenmesi".

Undergraduate Thesis, Karabük University. Karabük, Turkey, 2017.

[3] Varol H. S., Yağımlı M. "Renk Bileşenleri Yardımıyla Hareketli Hedeflerin Gerçek Zamanlı Tespiti". Journal of Naval Science and Engineering, 89-97, 2009.

[4] Çubukçu A., Kuncan M., İmren M., Erol F., Ertunç H. M., Öztürk S., Kaplan K. "Görüntü İşleme İle 3 Eksenli Robot Mekanizması Üzerinde Nesne Ayırt Edilmesi ve Sıralanması". Undergraduate Thesis, Kocaeli University. Kocaeli, Turkey, 2015.

[5] Şenel F. A., Çetişli B. "Görüntü İşleme Ve Beş Eksenli Robot Kol İle Üretim Bandında Nesne Denetimi". Research Article, Süleyman Demirel University. Isparta, Turkey, 2014.

[6] Aytan A. E., Öztürk Y., Örgev E. K. (1993). "Görüntü İşleme". İ. U. Journal of the Faculty of Dentistry, 1993.

[7] İşçimen B., Atasoy H., Kutlu Y., Yıldırım S., Yıldırım E., "Bilgisayar Görmesi Ve Gradyan İniş Algoritması Kullanılarak Robot Kol Uygulaması", Akıllı Sistemlerde Yenilikler Ve Uygulamaları, Mustafa Kemal University, İskenderun, Hatay, Turkey,2014.

[8] Vijayalaxmi P. B., Putta R., Shinde G., Lohani P.," Object Detection Using Image Processing for An Industrial Robot"., International Journal of Advanced Computational Engineering and Networking, ISSN: 2320-2106, Fr. C. Rodrigues Institute of Technology, Vashi, India,2013.

[9] Özgenel Ç.F.," Otonom Çatlak Tespitinde Kullanılan Görüntü İşleme Yöntemleri Ve Teknolojinin Potansiyelleri". 3. Ulusal Yapı Kongresi Ve Sergisi Teknik Tasarım, Güvenlik Ve Erişilebilirlik, TMMOB Mimarlar Odası, Ankara, Turkey,2016.

[10] Yılmaz V., Güngör O., Kadıoğulları A.İ.," Görüntü İşleme Teknikleri İle Ağaçların Boy, Tepe Çapı Ve Tepe Hacimlerinin Belirlenmesi ", TUFUAB VIII. Teknik Sempozyumu, Konya, Turkey,2015.

[11] Shaha S. K., Mishra R., Prasannaa D., Mohapatroa G.," Fabrication and Deviation Analysis of a 5-Axis Robotic Arm Using Image Processing Technique"., ScienceDirect, 4622–4629,2019.

[12] Kalaiarasan G., Thomas A., Kirubha C., Saravanan B. "Development of Edge Processing and Drawing Algorithms for a 6 Axis Industrial Robot". CAD/CAM, Robotics and Factories of the Future: Proceedings of the 28th International Conference on CARs & FoF. India, 2016.

[13] Cuşkun Y., Duman F., Basık H., Gün F., Kaplan K., Ertunç H. M. "Görüntü İşleme Tabanlı 4 Eksenli Çok Amaçlı Robot Mekanizması". Conference Paper, Mechatronics Engineering Department, Kocaeli University. Kocaeli, Turkey, 2016.

[14] Soans R. V., Hegde A., Kumar A., Nittul R., Singh C. "Object Tracking Robot Using Adaptive Color Thresholding". Proceedings of the 2nd International Conference on Communication and Electronics Systems (ICCES 2017), 2017.

[15] Adar N. G., Kozan R. "5 Serbestlik Dereceli Robot Kolunun Görüntü Esaslı Kontrolü". 16th National Machine Theory Symposium, Atatürk University, Engineering Faculty. Erzurum, Turkey, 2013.

[16] Dragusu M., Mihalache A. N., Solea R. "Practical Applications for Robotic Arms Using Image Processing". System Theory, Control and Computing (ICSTCC), 16th International Conference on IEEE. 2012.