24th International Conference
ENGINEERING MECHANICS 2018
Svratka, Czech Republic, May 14 –17, 2018
Paper **#313,** pp. 481–484, doi: 10.21495/91-8-481

DETERMINING THE TRAJECTORY OF FILLING A RECTANGULAR FLAT AREA FOR AN INDUSTRIAL ROBOT USING A MARKER AND A SINGLE-CAMERA VISION SYSTEM

M. Łaganowska[*], J. Zwierzchowski[**]

Abstract: *The paper presents a vision system of an industrial robot, used to identify the location of an object in space by means of a marker. On this basis, the trajectory of the robot is determined, which is to fill the area with a rectangular shape. The ArUco module was used to detect the markers. It allows the recognition of ARTag markers with their position and orientation in relation to the camera. The first stage of the algorithm is calibration of the camera, using the ChArUco table (it is part of the ArUco module) and reading the calibration parameters. Next part is generating a tag that will be used in the algorithm. The next step is to detect the marker and then determine its actual location. In the last stage, a border is drawn for the found tag. Marker location and orientation values resulting from the algorithm's operation are used to control the robot's movement in order to determine the desired trajectory.*

Keywords: vision system, image processing, markers detection, ArUco markers

1. Introduction

Automatic systems play an increasingly important role in controlling the work of industrial robots. The use of vision systems ensures a significant increase in the degree of industry automation.

In this work, the vision system is responsible for calculating the actual position of the object using the marker located on it. Location assessment is of great importance in many computer visualization applications. The process consists in finding the relationship between points in the real environment and the image projection 2D. Marker recognition and determining the position of the object is possible due to the use of the ArUco module from the OpenCV library.

OpenCV is a free open source library. It was built to create a common ground for computer vision systems and to accelerate the use of machine perception. The library has more than 2,500 algorithms that can be used to detect and recognize faces, identify objects, track moving objects, recognize scenery and establish markers in real time.

The use of synthetic markers greatly facilitates the task of recognizing objects. The ArUco module allows the recognition of ARTag markers along with determining the position and orientation of the object relative to the camera. The ArUco module is an addition to the OpenCV library and is used to detect markers. The ArUco marker has a square shape and consists of a black border and an internal binary matrix that determines its identifier. The black border allows quick detection of the marker, and the binary matrix enables its identification. The size of the tag is determined by the size of the internal matrix, e.g. a 4x4 size marker consists of 16 bits.

[*] mgr inż. Małgorzata Łaganowska: Department of Automation and Robotics, Faculty of Mechatronics and Mechanical Engineering, Kielce University of Technology, al. Tysiąclecia Państwa Polskiego 7; 25-314, Kielce; PL, mlaganowska@tu.kielce.pl

[**] dr inż. Jarosław Zwierzchowski: Department of Automation and Robotics, Faculty of Mechatronics and Mechanical Engineering, Kielce University of Technology, al. Tysiąclecia Państwa Polskiego 7; 25-314, Kielce; PL, j.zwierzchowski@tu.kielce.pl

Description of the test stand

The main element of the test stand is the KAWASAKI FS 06N industrial robot with the D+ series controller. The robot communicates with the application installed on the PC. The digital camera implemented in the system works directly with the software and transfers the image in high resolution to the computer. The station for detecting objects and generating trajectories is presented in Fig. 1.

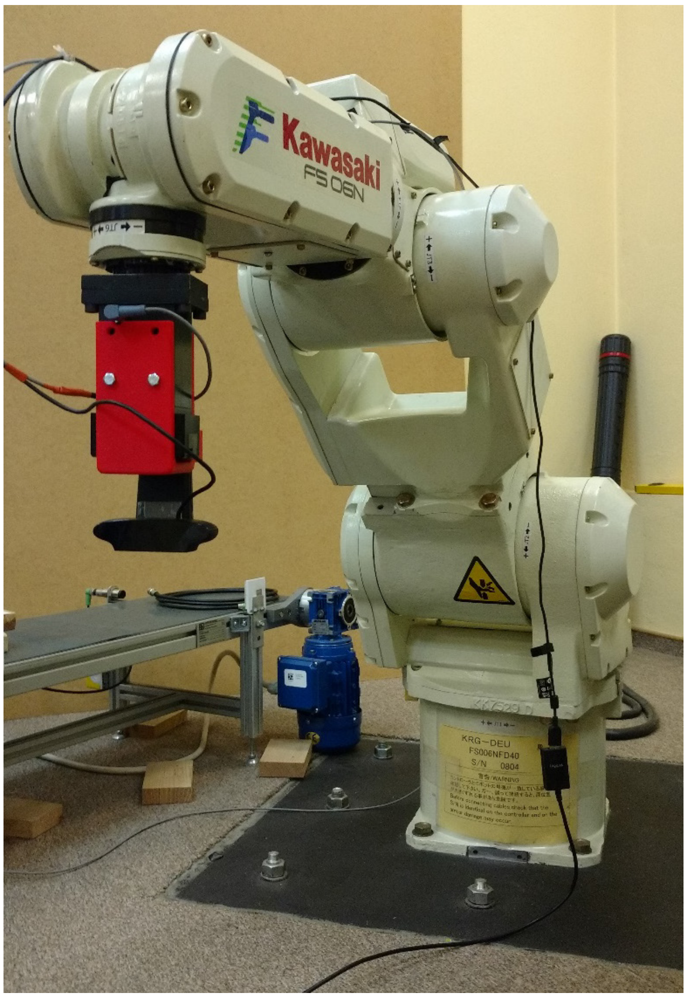

Fig. 1: Test stand.

2. Image processing algorithm

The first stage of the algorithm's operation is calibration of the camera using the ChArUco module and the calibration board. ChArUco is a module implemented in ArUco. It allows user to get a higher calibration quality, because the corners of the ChArUco board are much more accurate compared to the corners of the ArUco board. Calibration of the camera was carried out using a program whose source code is supplied with the ArUco module. The calibration table was generated by a program that was also supplied with the ArUco module. The ChArUco board was created by adding markers on the white chessboard boxes, while the ArUco board is built only from the markers. The parameters obtained from camera calibration were saved in the file *kalibracja*.

The next step is to read the calibration parameters. The *readCameraParameters()* function opens the *kalibracja* file and reads the calibration parameters. In the *camera_matrix* matrix there are camera parameters, and distortion factors in the matrix *distortion_coefficients*. The data from the above matrices have been written appropriately under the variables *camMatrix* and *distCoeffs*.

The next step is to generate the tag used in the algorithm. First, define the range in which the marker is contained. The range indicates the number of bits that make up the marker and the number of tags in this range. The scope statement is made using the following function *Ptr <aruco :: Dictionary> dictionary = aruco :: getPredefinedDictionary (aruco :: DICT_4x4_50)*. This range contains 50 tags that consist of 16 bits (4x4). These markers have numbers from 0 to 49. The following function is used to generate a tag: *drawMarker (dictionary, id, sidePixels, OutputArray, borderBits)*. The *dictionary* parameter specifies the range in which the marker is contained. *Id* is the marker number described in the integer variable. *SidePixels* determines the dimensions of the marker, e.g. 50 means 50x50 pixels. *OutputArray* is the output image of the marker. In the *borderBits* parameter, the width of the marker frame is given.

The vision system algorithm has been implemented in the *getCameraProcessed()* function. First, the image from the camera using the *getCameraMat()* function is downloaded. This image is converted into a grayscale. If necessary, the exact parameters of the marker, e.g. correct identifiers must be specified. These parameters must be saved in a file and then read using the *readDetectorParameter()* function. No additional marker parameters were imposed on this paper. The next stage of the algorithm's operation is to read the camera parameters after calibration. The calibration procedure has been described earlier. The next step in the algorithm is to detect the tag. This is done using the *detectMarkers(image, dictionary, corners, ids, detectorParameters, rejected)* function. The *image* parameter is the name of the variable that stores the input image taken from the camera. *Dictionary* is the range of the generated tag, and *corners* is a vector that stores the coordinates of the tag's corners relative to the tag. The tag identifier is located in the *ids* variable. *DetectorParameters* are tag parameters. Vector *rejected* stores images of markers with an incorrect identifier.

The basic stage of the algorithm is to estimate the actual position of the marker. This is done using the *estimatePoseSingleMarkers(corners, markerLength, camMatrix, distCoeffs, rvecs, tvecs)* function. Parameters of this function are:

- corners – defines the four corners of the marker in the coordinates of the marker, it is passed from the detectMarkers() function.

- markerLength – the length of the side of the marker.

- camMatrix – camera parameters from the calibration file.

- distCoeffs – distortion coefficients from the calibration file.

- rvecs – a 3-element vector that specifies the orientation of the marker relative to the camera.

- tvecs – a 3-element vector defining the position of the marker in the X, Y, Z axis with respect to the camera.

The last step is to draw a border for the found marker. The drawDetectedMarkers(image, corners, ids) function serves this purpose. The image parameter is the output image on which the marker border will be located. Corners are previously recognized marker corners. The ids parameter stores the tag identifier.

Finally, the values of the rvecs and tvecs vectors should be retrieved, using the wektorrot() and the wektrotrans() functions. The position and orientation values of the marker will be used to control the robot.

The steps of the detection algorithm are presented in Fig. 2.

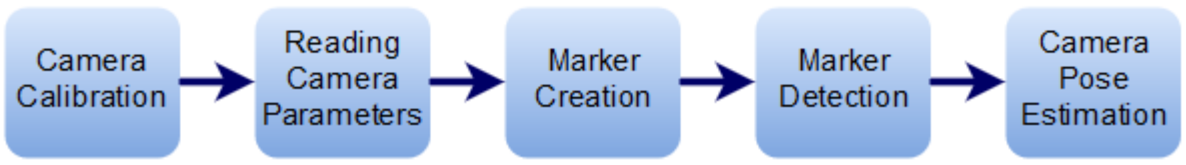

Fig. 2: Steps of the detection algorithm.

The image obtained as a result of the algorithm's operation is shown in Fig. 3. The marker's border is marked in green. In the middle his identifier was displayed.

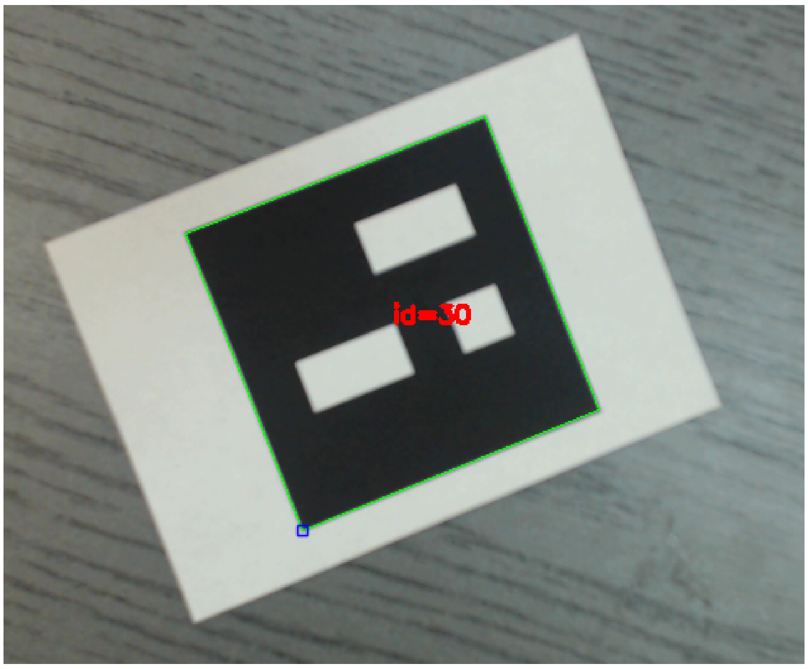

Fig. 3: Identified tag.

3. Implementation of the robot's movement

To support the vision system and connect with the robot, a window application was used as an element of the station's equipment. The application was written in Microsoft Visual Studio using the Qt libraries. The robot's communication with the application is carried out using the TCP/IP protocol. The application allows user to connect to the robot terminal via the KCwinTCP program. Using the application, it is also possible to turn the camera on and off.

The rotation and translation vectors passed in the functions *wektortrans()* and *wektorrot()* are used to control the robot. These are vectors determining the position and orientation of the point lying in the center of the marker, relative to the camera coordinate system. The coordinates of this point are sent to the program that realizes the movement of the robot. Next, the camera offset should be taken in relation to the tool coordinate system. Based on coordinates of the center of the marker, the coordinates of the vertices of the rectangular area are calculated. Thanks to this coordinate, the robot tool can realize the trajectory of filling in a given area.

4. Conclusions

The work presents a vision system for detecting objects using markers and the ArUco module. As a result of the operation of the appropriate image processing algorithm, the values of the object shift and rotation relative to the camera were obtained. These values were used to implement the assumed robot movement. The use of markers makes it easier to detect objects. The main advantage of using them is that a single marker provides sufficient parameters to obtain a real position. In addition, internal binary coding enables them to be reliably identified.

References

Bacik, J. (2017) Autonomous flying with quadrocopter using fuzzy control and ArUco markers, Springer Berlin Heidelberg.

Garrido-Jurado, S., Munoz-Salinas, R., Madrid-Cuevas, F. J. and Marin-Jimenez, M. J. (2014) Automatic generation and detection of highly reliable fiducial markers under occlusion, Pattern Recognition, 47, pp. 2280-2292.

Kawasaki Heavy Industries, Ltd. (2007) Programowanie w języku AS.

Moeslund, T. B. (2012) Introduction to Video and Image Processing, Springer-Verlag London.