symmetry

MDPI

Article

Programming of Industrial Robots Using the Recognition of Geometric Signs in Flexible Welding Process

Olaf Ciszak, Jakub Juszkiewicz and Marcin Suszyński *

Faculty of Mechanical Engineering, Poznan University of Technology, 60-965 Poznań, Poland; olaf.ciszak@put.poznan.pl (O.C.); jakub.juszkiewicz@gmail.com (J.J.)

* Correspondence: marcin.suszynski@put.poznan.pl

Received: 20 July 2020; Accepted: 26 August 2020; Published: 28 August 2020

Abstract: The purpose of the article was to build a low-cost system for identifying shapes in order to program industrial robots (on the base of the six-axis "ABB IRB 140" robot) for a welding process in 2D. The whole system consisted of several elements developed in individual stages. The first step was to identify the existing robot control systems, which analysed images from an attached low-cost digital camera. Then, a computer program, which handles communication with the digital camera capturing and processing, was written. In addition, the program's task was to detect geometric shapes (contours) drawn by humans and to approximate them. This study also presents research on a binarization and contour recognition method for this application. Based on this, the robot is able to weld the same contours on a 2D plane.

Keywords: robotic welding; identifying shapes; vision system

1. Introduction

Symmetry as a spatial relation through geometrical transformations is of great importance in programming of industrial robots particularly in the flexible welding process. Moving the TCP (tool centre point) using machine vision techniques is related to the control of the robot, which is able to move with high repeatability, but with low accuracy [1]. The accuracy of the robot positioning in the case of machine vision is much more important than its repeatability, though there is often a complex industrial environment with numerous moving objects, and in the case of cooperating robots, also people. Such modern technology, also in the case of industrial processes with high flexibility, opens up completely new opportunities, but also creates new and quite complex challenges in the design of security [2]. Surfaces without texture, lighting conditions, occlusions, unspecified or moving objects are critical issues that the vision system must deal with. Obtaining a 3D point cloud to assess the position of the robot, running a robot or any other purpose, can be achieved through the use of many different sensors and techniques, depending on the needs and requirements of the particular application to decide which is best suited [3]. Spatial coordinates and a large number of points can be obtained almost immediately or within seconds, but they require further processing. Point clouds must be processed using specific algorithms to identify the geometry of the elements, the parts to be detected, measured or set. Then, filtering operations, a structure or interactive segmentation of the point clouds should be carried out [4–6]. In industrial processes, two-dimensional surfaces are often enough, which do not have to be so advanced, and can be a relatively economical and fast-acting solution, which in the further part of the research, will be developed into 3D. This work deals with fast image processing and generating code that is understandable for the robot in RAPID language. The main priority of this work was to build a low-cost and novel robotic system with a digital camera, which detects drawn or

simulated contours on a 2D surface and on this basis, generates the robot motion instructions on the desired trajectories. The system is based on high-level computer language (C++/CLI), performing the function of data acquisition, processing, generating results and is also able to communicate with people and other devices. Such a system can be particularly useful for the fast programming of the welding trajectory, especially in small-lot production.

2. Vision-Guided Automatic Robotic Path-Teaching Method

Applications such as welding, painting, or gluing are some of the common jobs carried out nowadays by industrial robots. However, welding is one of the most important processes, and the tasks involved are usually programmed using conventional methods, like using a teach pendant, in an online programming manner on highly constrained environments. The following sections of this article present a low-cost system supporting the fast welding-path programming of industrial robot tracks based on a vision system. The simplified method algorithm is presented in Figure 1.

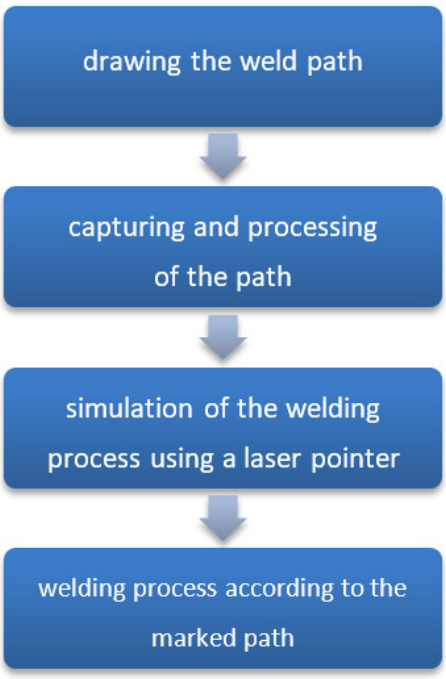


Figure 1. Simplified method algorithm.

The Scheme of a Robotised Workstation

The general scheme of operation of the proposed system is shown in Figure 2. The camera was placed on the workstation, connected via a USB interface, and transmits images to the processing system, which takes the form of a PC, in which, after they are intercepted and processed, a RAPID code is generated. Then, this code, via a dedicated RobotStudio program, is sent entirely by the Ethernet standard to the robot. In its tool there is a laser pointer, which is controlled by one of its signal outputs, simulating a welding tool (classic or laser welding).

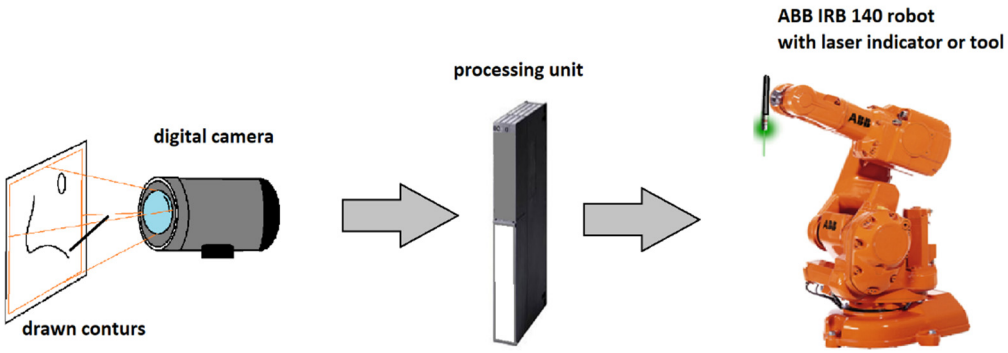


Figure 2. The scheme of the robotised workstation.

Controlling the robot from the computer is achieved with the help of the created operating software supporting the whole workstation. It allows the entire process to be automated, from the data acquisition, through image analysis and processing, to code generation and robot control (Figure 3). First, the camera captures the shapes to which it is directed, drawn by hand with a black marker, illustrating the path of the welding location. Then, the program with the appropriate conversion threshold selection parameters, performs the binarization of the image to give one bit per pixel. Due to the presence of a large number of points with which the robot may have a problem while moving (minor movements—vibrations), points close to each other are allocated to individual contours, and then according to the given accuracy, reduced. Reducing the number of points will allow the faster generation of the trajectory of movement and a more complete use of the robot's movements (smoother movement).

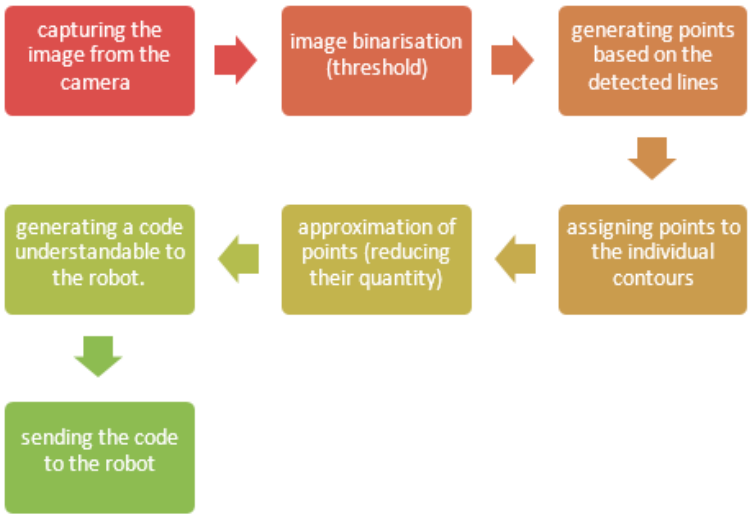


Figure 3. Simplified scheme of the program.

3. Development of the Algorithm and the Robot Control Program

By choosing a programming language (C++), it was also decided to use the Open Source library—OpenCV. This library has a huge number of functions that are constantly being developed and used during image processing, mainly in real time. This work uses the currently available version (30 March 2017) described as "OpenCV-3.2.0" [7], and for graphic development, a Windows Visual CLR environment was used, which in its database has all the necessary elements to build a functional program, additionally interacting to a large extent with the classic language C++/CLI. [8].

3.1. Development of Graphic Layout

The interface of the program focused on its intuitiveness and ease of use. In the tab views, two ways of approximation were placed, allowing the simultaneous editing of various process parameters. The most important functions of the program are presented below on Figure 4.

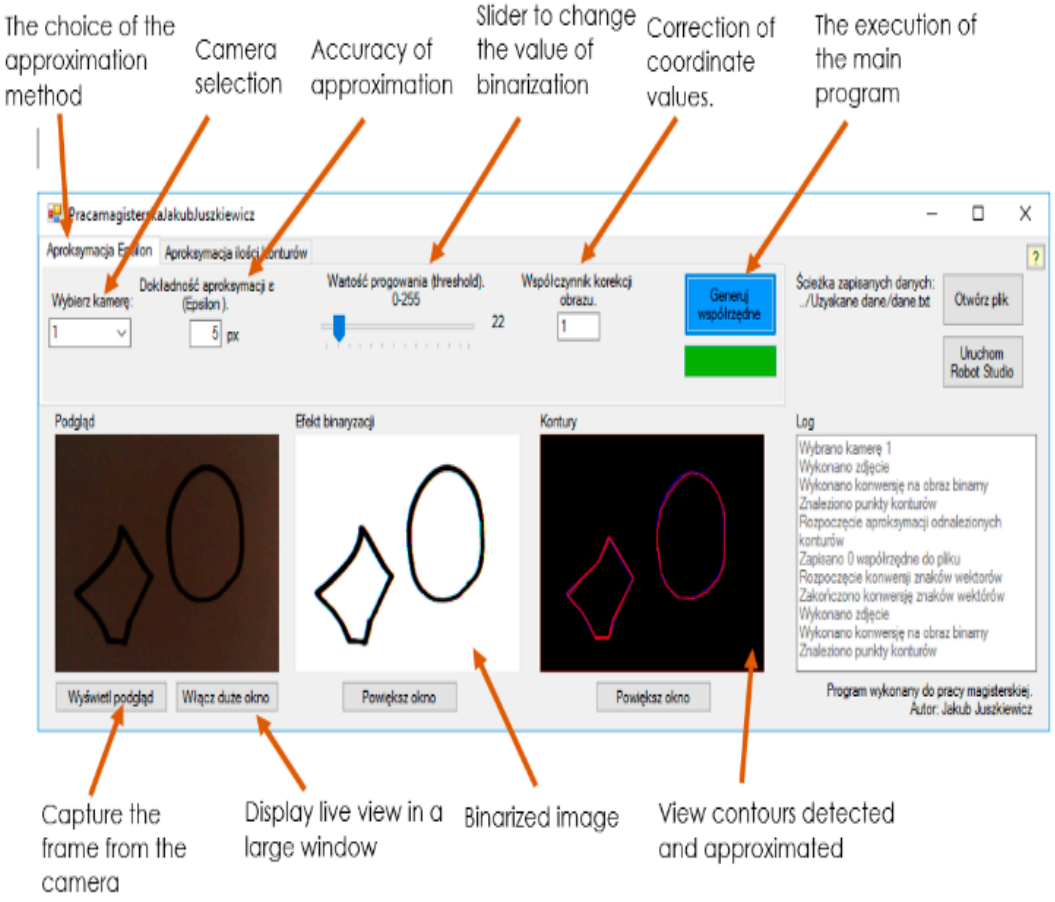


Figure 4. The program interface.

3.2. Program Code

The main assumptions of the system operation implemented in the program are presented in further points.

3.2.1. Binarization

The image processing in this case consisted of saving each element with one bit (black or white). There are several methods for binarizing the image. In order to best fit the binarization model to the given task, the following three binarization methods were tested [9].

Method 1—Binarisation with a given threshold

Determination of the threshold—Threshold level. All elements below a given threshold are converted to 0 (black) and those above to 255 (white). Before this happens, however, the colour image is converted to grayscale [10] using the CV_BGR2GRAY or CV_LOAD_IMAGE_GRAYSCALE function.

threshold(m, mThresh, 30, 255, THRESH_BINARY);

Method 2—Otsu binarization

This method, named after Nobuyuki Otsu and also called the optimal threshold method, assumes that the histogram consists of two normal distributions (foreground and background pixels). The Otsu algorithm checks all the options and selects the optimal threshold [11]. With complex images, this is often not the case, which results in incorrect results.

threshold(img, Otsu, 0, 255, THRESH_BINARY | THRESH_OTSU);

Method 3—Adaptive Thresholding

In some situations, performance can be significantly improved by using multiple thresholds. Adaptive Thresholding is such a method. It consists of three steps:

- Division of the image into separate areas of n × n pixels (the value is selected manually);
- Determination of the binarization threshold in each area (e.g., with Otsu);
- For each pixel, use a local threshold that is approximated by the four nearest thresholds [12,13].

Mat binary_img, Otsu, prog;

int block_size = 11;//The size of the adjacent area, to calculate the threshold value for the pixel

double offset = 15;//Constant subtracted from the average or weighted average on which the binarization result depends

adaptiveThreshold(img, binary_img, 255, ADAPTIVE_THRESH_MEAN_C, THRESH_BINARY, block_size, i);

3.2.2. Comparison of the Results Obtained

Aggregate results for the methods presented for the task under consideration were summarised in Table 1. Despite the expectation for the Otsu method to be best suited, it failed to produce the expected results. For the whole image, the algorithm selected a threshold value of 82. In order to confirm the operation of the Otsu algorithm, a fragment of the image with a smaller tonal range was analysed. In this case, a well selected threshold was closer to the expected value i.e., 37. For this method, the result correctness level can be increased by using additional noise filters (e.g., Gauss filter) [10]. Investigating them methods of binarization and setting various image conversion parameters, the best results were obtained by the simplest possible algorithm, setting a pre-determined binarization threshold. Despite a poorly illuminated input image (uneven light from one side), the selection of the right threshold turned out to be very simple. Therefore, we decided to use this method in our programme.

Table 1. Comparison of the end results of the binarization methods.

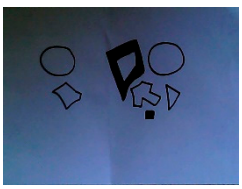
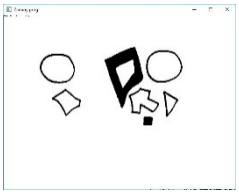
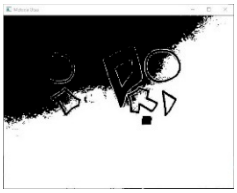
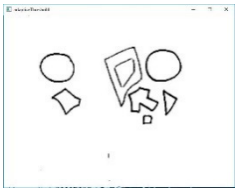
Comparison of Binarization Results	
The source image	The threshold method (value = 30)
	

Table 1. *Cont.*

Comparison of Binarization Results	
Otsu method	Adaptive threshold method
	

3.2.3. Contour Recognition

There are many methods for contour recognition. The following methods from the library have been tested for the usefulness of the data obtained for the task (Table 2):

- CV_RETR_EXTERNAL—only searches for the most external contours. In this case, it only found one contour touching the lower edge of the image;
- CV_RETR_LIST—searches for all contours, both internal and external, as well as the contours of solid figures. It does not create a data storage hierarchy;
- CV_RETR_CCOMP—searches for all the contours and segregates them in a two-level hierarchy. On the top level, external contour borders appear and on the second level, internal borders do. If there is another contour in the opening of any of the elements, it will be assigned according to the appropriate levels;
- CV_RETR_TREE—searches for all the contours and creates a full hierarchy of nested contours, starting with coordinates [0.0], and ending with the maximum coordinates [14]. The results of their operation are presented in graphical form in the table below:

Table 2. Comparison of the contour recognition methods.

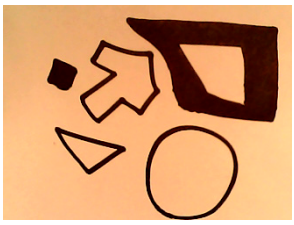
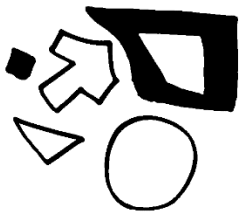
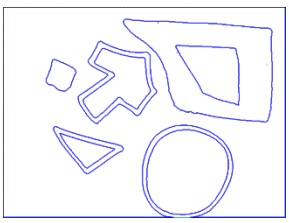
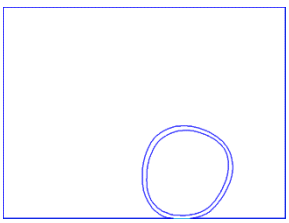
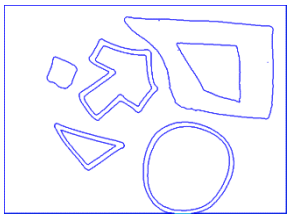
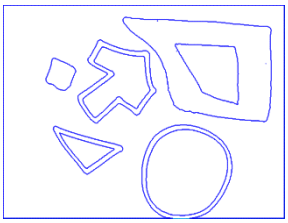
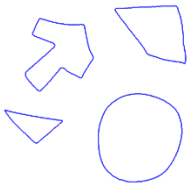
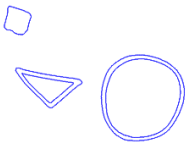
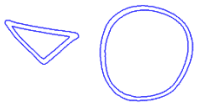
Comparison of Contour Recognition Methods	
The source image from the camera	Image after binarization
	
Methods of recognition	
CV_RETR_CCOMP	CV_RETR_EXTERNAL
	

Table 2. *Cont.*

CV_RETR_LIST	CV_RETR_TREE
	

When checking the hierarchy for each method, a patch was added to the algorithm to exclude half of the selected contours (Table 3).

Table 3. Attempt to obtain internal contours.

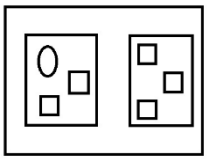
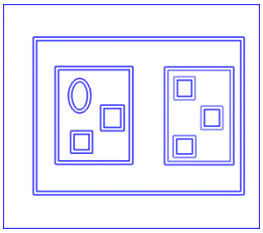
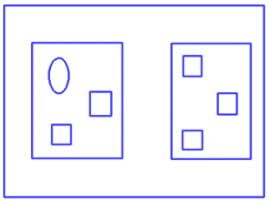
Attempt to Obtain Internal Contours	
CV_RETR_CCOMP	CV_RETR_EXTERNAL
	
CV_RETR_LIST	CV_RETR_TREE
	

The lack of systematisation in the contour saving makes the CV_RETR_CCOMP method the most suitable for implementation in this robotised system. In this case, the external and internal contours are ordered on two levels (the first one includes external contours, the second one, internal contours), which allows them to be extracted quite simply. In order to handle the nested contours, we decided to use a modified drawing version and modify the parameters of the loop. The I = hierarchy[i][0] function was substituted for the loop counter.

```
for (i = 0; i < contours.size()-2; I = hierarchy[i][0])
{
        drawContours(mc, contours, i, Scalar(255, 0, 0), 1);
}
```

Its task is to present the detected internal contours only. Additionally, in order to exclude the outline of the whole image from further processing (Table 4), the last hierarchy level was excluded from the parameter ending the loop's operation—it is always the outline.

Table 4. Detection of nested contours.

Detection of Nested Contours	
The source image	All contours detected
	
Drawn contours after modification of the loop operation	
	

3.2.4. Approximation of Contours—Ramer–Douglas–Peucker Algorithm

The purpose of this algorithm is to analyse a curve consisting of line segments (commonly called polyline) for the purpose of finding a similar complex curve that contains fewer points. The algorithm creates it based on the maximum distance between the original curve and the new, simplified curve (i.e., the Hausdorff distance between the curves). The simplified curve consists of a subset of points that defined the original one [15–17].

Hausdorff distance—also called the Hausdorff metric or Pompeiu–Hausdorff distance, it is the distance between subsets in the same complete metric space [18]. Two sets of points are close in Hausdorff distance if each point of each set is located close to some point of the second set. It is thus the greatest among the distances between a point in the first set to the nearest point in the second set. In Figure 5, examples of the greatest Hausdorff distances for set X (1) and set Y (2) [18–20] are marked.

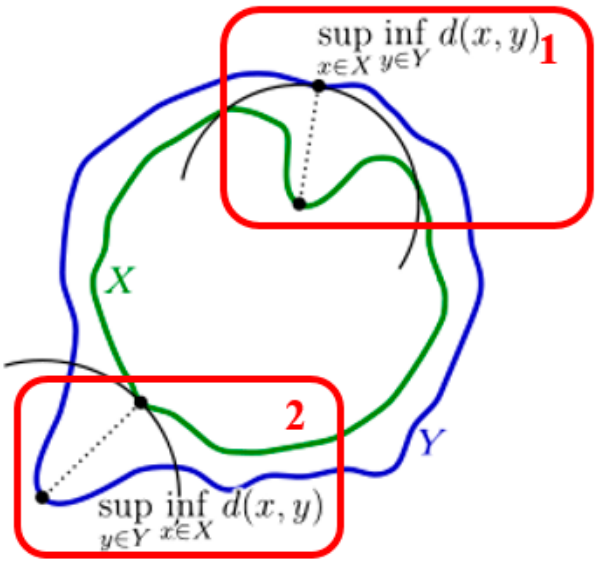


Figure 5. Hausdorff distance [16].

The steps of the Ramer–Douglas–Peucker algorithm are as follows:

- Start and end point retrieval—finding the farthest point from the line between the extreme points;
- Checking whether the distance between the farthest point and the line is bigger or smaller than the set epsilon value;
- If the distance is greater, the program skips this point and calls the function again. At this moment, the program checks all the points from the first to the most distant point;
- The procedure is repeated until the program finds a point at a distance smaller than epsilon. If it is the case, it enters into a new table the first point together with the one that was farthest away at the very beginning.

In the new table, the new points are added as the algorithm progresses, whose distance from the current extreme points exceeds epsilon. In this way, a new set of points is created with a smaller number of points, on the basis of which the program is able to build a new, less complex curve [2,5,16]. Methods of analysing a curve consisting of line segments include:

Method I

Approximation consists of finding the maximum values of the distance between the points of the new curve and those of the original curve. Approximation accuracy(ε) value in pixels is responsible for the precision of the contour determination.

```
for (k = 0; k < j; k++)//k- number of closed contours
{
approxPolyDP(contours[k], approx,
contours.push_back(approx);
drawContours(mc, contours, contours.size() – 1,
Scalar(0, 0, 255), 1);//draw approximatedcontours in red color
}
```

Method II

The loop is executed until the approxPolyDP function brings the new curve closer to a specified number of vertices. This time, the value of the approximation accuracy does not mean the distance measured in pixels, but the number of vertices to be reached.

```
for (k = 0; k < j; k++)//k- number of closed contours
{
   double d = 0;
   do
   {
    d = d + 1;
    approxPolyDP(contours[k], approx, d, true);
   }
   while (approx.size() > dokladnoscAproksymacji);//end approximating when the required
   number of vertices is reached
```

3.2.5. Saving Coordinates to the File—Preparation of the Code for the RobotStudio Program

The obtained data are subjected to several operations in order to prepare them to be sent as instructions to the robot. The first operation includes checking and deleting the old file with the previous commands. A new file is then created, in which the current detected and approximated coordinates are saved. Between the subsequent contours, information on turning the laser on or off is added to for the simulation of the welding process, or for the actual welding process. The next step includes the conversion of the coordinates so that they conform with the structure of the RAPID language.

3.2.6. Preview of Detected Contours

To correctly display all the generated contours in the preview window, vector information is converted into a bitmap image.

```
System::Drawing::Graphics^ graphics2 = zdjKontury->CreateGraphics();//in the
        zdjBinarne field, create a graphic called graphics1
System::IntPtr ptr2(mc.ptr());
//convert data line by line to pixels in a bitmap image
System::Drawing::Bitmap^ b2 = gcnew System::Drawing::Bitmap(mc.cols, mc.rows,
        mc.step, System::Drawing::Imaging::PixelFormat::Format24bppRgb, ptr2);
System::Drawing::RectangleF rect2(0, 0, zdjKontury->Width, zdjKontury->Height);
graphics2->DrawImage(b2, rect2);//display the bitmap by assigning it to graphics1
```

4. Development and Construction of a Test Stand

An indispensable structural element of a working stand includes the design of its individual parts, as well as their assembly. In order to carry out tests of the developed system, a laser electronic system and its casing were designed and made using rapid prototyping technology (Figure 6). This allows for a much simpler verification of the solution than in the case of the classic welding process.

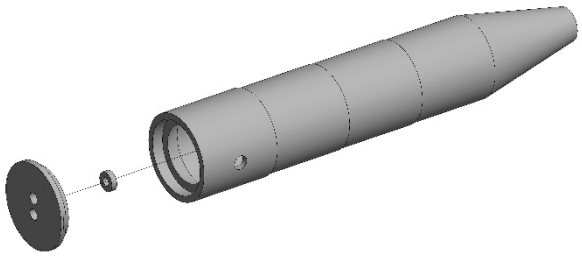


Figure 6. Casting—exploded view.

5. Test and Verification of the System Solution

In order to move the laser module correctly, it was necessary to define the tool coordinates in order to test the positioning accuracy of the welding head or laser. The tip of the laser casing had to be positioned at the same point in different robot axis configurations. A four-point method was used to determine the TCP (tool centre point) and tool layout. This results in the coordinates generated in the program did not have to be additionally converted to the basic coordinate system of the robot. Three points are sufficient to determine a new one. Two from the X axis and one from the Y axis (Figure 7).

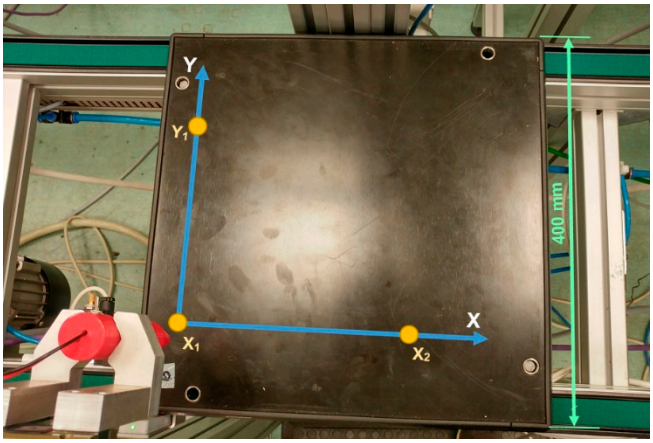


Figure 7. Defining the local coordinate system.

Execution of the Generated Program Code

During the tests, we noted that the robot does not move smoothly. The robot stopped once the set trajectory points were reached, which was caused by the definition of the way the given coordinates were reached. All of them were marked as fine. Due to this, in order to make the next move, the robot first had to reach the previous value as accurately as possible. After changing the fine command to z1, the robot started to send motion data in parallel with the start and stop commands from the digital output (Figure 8) [21,22]. This caused the laser to start too early, when it has not yet reached the point at which it should switch on. The advantage of this definition was that the movements became smooth. After approaching the target point at 1 mm, the robot performed a small arc to reach the next movement trajectory.

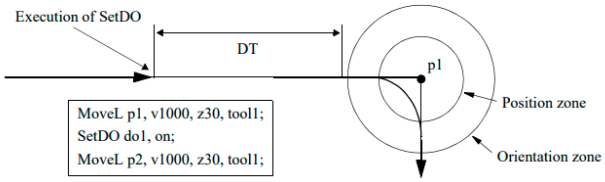


Figure 8. Switching on the output before reaching the point [3].

The solution to this problem was to use fine and z1 commands. Before each command to enable (SET) or disable (RESET) the output, the motion command had to be marked as fine. In such a case, the robot reached the target in a precise manner and then turned the laser on/off. The remaining motion commands describing the approximated contour were executed with z1 approximation [22]. The work of the running robot station with the implemented and described system is shown in Figure 9.

Figure 9. Test and research stand.

6. Conclusions

In this article, we developed assumptions and implemented a low-cost system based on an industrial vision for the simple programming of the movement path, which is particularly important

due to the high variability of welded products. It is based on a fully functional program for analysing camera images and a test effector in the form of a laser.

Such solutions are more and more often needed to prepare quick tests or implement the first test products. The system for converting a handwritten drawing into a welded path was designed due to the lack of similar fast solutions. The requirements analysed allowed for a comprehensive research of available possibilities of mathematical image processing. For this purpose, a specific fragment of the OpenCV library was used, which produced a satisfactory result in reproducing the handwritten image.

Even though the vision system achieved good functionality levels, several additional improvements could be proposed. The first of them may be the addition of a function automatically calculating the correction factor of the image, based on the data entered about the camera resolution, distance and angle of view. The next stage in the development of this functionality may include the automatic collection of the previously mentioned information on the basis of a template placed within the camera vision area, or data from a distance sensor and gyroscope. The development of the system may also enable the implementation of the robot communication protocol directly into the program. With the possibility of sending motion commands, the necessity of using an intermediate program would be eliminated. The system can also be extended to operate in a three-dimensional space, which will be subject to further research and testing.

Author Contributions: Conceptualization, O.C. and J.J.; methodology, J.J.; software, J.J. and O.C.; validation, J.J., O.C. and M.S.; formal analysis, M.S. and O.C.; investigation, J.J.; resources, J.J. and M.S.; data curation, J.J.; writing—original draft preparation, J.J.; writing—review and editing, M.S.; visualization, M.S.; supervision, O.C.; project administration, O.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Ministry of Science and Higher Education of Poland (No. 0614/SBAD/1529).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Hefele, J.; Brenner, C. Robot pose correction using photogrammetric tracking. In Proceedings of the Intelligent Systems and Smart Manufacturing; International Society for Optics and Photonics, Boston, MA, USA, 12 February 2001; pp. 170–178.
2. Salmi, T.; Väätäinen, O.; Malm, T.; Montonen, J.; Marstio, I. Meeting new challenges and possibilities with modern robot safety technologies. In *Enabling Manufacturing Competitiveness and Economic Sustainability*; Springer International Publishing: Montreal, QC, Canada, 2014; pp. 183–188.
3. LMI Technologies. Available online: http://lmi3d.com/products (accessed on 30 April 2019).
4. Suszyński, M.; Wojciechowski, J. No Clamp Robotic Assembly with Use of Point Cloud Data from Low-Cost Triangulation Scanner. *Tehnički Vjesnik* **2018**, *25*, 904–909.
5. Wojciechowski, J.; Suszyński, M. Optical scanner assisted robotic assembly. *Assem. Autom.* **2017**, *37*, 434–441. [CrossRef]
6. Point Cloud Library (PCL). Available online: http://pointclouds.org/documentation/tutorials/ (accessed on 29 July 2018).
7. Open Source Computer Vision Library. In Github. Available online: https://github.com/opencv/ (accessed on 30 June 2017).
8. NET Programming with C++/CLI. In MSDN. Retrieved 2 July 2009. Available online: https://msdn.microsoft.com/en-us/library/68td296t.aspx (accessed on 2 July 2017).
9. Rafajłowicza, E.; Rafajłowicza, W.; Rusiecki, A. *Algorytmy Przetwarzania Obrazów i Wstęp do Pracy z Biblioteką OpenCV*; OWPW: Wrocław, Poland, 2009; pp. 120–190.
10. Load, Modify, and Save an Image. In OpenCV Documentation. Available online: https://docs.opencv.org/2.4/doc/tutorials/introduction/load_save_image/load_save_image.html (accessed on 10 July 2018).
11. Otsu Thresholding. In the Lap Book Pages. Available online: http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html (accessed on 10 July 2018).
12. Image Thresholding. In OpenCV Documentation. Available online: https://docs.opencv.org/3.3.1/d7/d4d/tutorial_py_thresholding.html (accessed on 15 July 2018).

13. Binaryzacja. In Analiza Obrazu. Available online: http://analizaobrazu.x25.pl/articles/17 (accessed on 17 July 2020).
14. Structural Analysis and Shape Descriptors. In OpenCV Documentation. Available online: http://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=findcontours#findcontour (accessed on 20 July 2017).
15. Visvalingam, M.; Whyatt, J.D. Line Generalisation by Repeated Elimination of the Smallest Area. Discussion Paper, CISRG, The University of Hull. Available online: https://hydra.hull.ac.uk/assets/hull:8338/content (accessed on 28 July 2018).
16. Liu, J.; Li, H.; Yang, Z.; Wu, K.; Liu, Y.; Liu, R.W. Adaptive Douglas Puecker algorithm Thresholding for AIS-based vessel trajectory compression. *IEEE Access* **2019**, *7*. [CrossRef]
17. Furkan, G.; Alev, M.; Orhan, A. Analysis of Ramer-Douglas-Peucker algorithm as a discretizatiom method. In Proceedings of the 26th Signal Processing and Communications Applications Conference (SIU), Cesme, Izmir, 2–5 May 2018.
18. Rockafellar, R.T.; Wets, R.J. *Variational Analysis*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 117–118.
19. Henrikson, J. Completeness and Total Boundedness of the Hausdorff Metric (23 June 2002). Available online: https://web.archive.org/web/20020623095720/http://www-math.mit.edu/phase2/UJM/vol1/HAUSF.PDF (accessed on 8 September 2017).
20. Marosevic, T. The Hausdorff distance between some sets of points. *Math. Comun.* **2018**, *23*, 247–257.
21. ABB AB Robotics. Technical Reference Manual RAPID Instructions, Functions and Data Types. 2014. Available online: https://library.e.abb.com/public/688894b98123f87bc1257cc50044e809/Technical%20reference%20manual_RAPID_3HAC16581-1_revJ_en.pdf/ (accessed on 3 September 2017).
22. ABB Automation Technology Products AB. RAPID reference manual, 3HAC 7774-1 Revision B. 2003. Available online: http://futurecnc.code.arc.cmu.edu/wp/wp-content/uploads/2011/12/RAPID-Reference-Manual-Instructions.pdf (accessed on 3 September 2017).