IMAGE ANALYSIS ALGORITHM FOR IMAGE BASED VISUAL SERVOING OF A ROBOTIC ASSEMBLY LINE

Jan-Iliuță-Romeo COJOCARU[1], Dan POPESCU[2]

This paper proposes an image analysis algorithm for a robotic assembly line workflow. Using efficient Matlab® routines, we devise a standalone application implemented as a computing server for the visual servoing of the robotic assembly line. The computing server is running on a PC and communicates with a Modbus server integrated on the assembly line PLC. On primary detection routine of the proposed algorithm is obtained 100% detection rate at approximately 14 FPS. Moreover, the computing server is an open system and can be easily advanced to new image analysis features, thus answering to new assembly line iterations.

Keywords: image analysis, image based visual servoing, robotic assembly line

1. Introduction

The Image Based Visual Servoing (IBVS) concept was first introduced in [1] to suggest a complete design and computation scheme of image features (area and centroid coordinates of the object) for robot dynamic control. There are a multitude of approaches for visual servoing systems, classified as Image Based and Position Based Visual Servoing (PBVS), and as a hybrid method.

Paper [2] proposed a three-stage control scheme for an industrial robot equipped with a monocular camera positioned in eye-in-hand configuration. This configuration allowed decupled rotational and translational movement of the manipulator, while overcoming some of the drawbacks of the IBVS approach. Also, in paper [3] the authors had improved the earlier control scheme by adding a feature reconstruction algorithm based on Kalman filter. In paper [4], an IBVS control mechanism was presented for continuous visual feedback in object occlusion situations based on homography with image feature prediction. A curve-feature-based depth-independent shape IBVS method for 3D object overseeing by a seven degree-of-freedom manipulator with revolute joints was presented in [5]. The authors in [6] had described the cross-domain couplings in high-speed IBVS systems that depend on many factors and constraints like image sensors, processing systems, and vision algorithms.

[1] PhD Student, Dept. of Automatic Control and Industrial Informatics, University POLITEHNICA of Bucharest, Romania, e-mail: romeo.cojocaru@upb.ro

[2] Prof., Dept. of Automatic Control and Industrial Informatics, University POLITEHNICA of Bucharest, Romania, e-mail: dan.popescu@upb.ro

In paper [7] a PBVS scheme was proposed for dynamic path tracking by correcting the movement of the end effector in real time constraints, using eye-to-hand photogrammetry and experimental results showed a significant improvement of precision for line and circle shapes. Paper [8] developed an eye-in-hand IBVS used by a concentric tube robot capable of working in a confined space. The main requirement for IBVS is the convergence to zero for image feature errors. More recently, in paper [9] an inversion-free neural controller scheme was proposed for the IBVS supplying an asymptotic convergence to zero. The novel approach of cloud computing was used in [10] to implement a cloud based IBVS module that used dynamic feedback control to minimize relative distance between the robot and the image space target.

For IBVS applications in agriculture, authors of [11] described an IBVS for harvesting robot using an adaptive algorithm based on reinforcement learning for image segmentation task. As aerial IBVS applications, an approach for payload stabilization using two axes video camera control was presented in [12].

The assembly line which we rely on our studies had various development stages. An early development stage had considered a Cognex® enclosed system for the image analysis operations that integrates a low-resolution greyscale video camera for basic shape and orientation tasks. Another assembly line iteration assessed an IFM® 3D visual sensor that offers voxel-based operations for distance measurement. The current iteration of the image analysis algorithm for the visual servoing of the robotic arm, uses an Arecont® Vision color camera.

Our goal is to deliver an efficient mechanism for the IBVS requirements. Developing a standalone Matlab® application as an open platform for further developments, allow us to reach such requirements in a more reasonable system that those produced by Cognex® and IFM®. Also, using simple and optimized routines for image analysis tasks, the processing time is compatible to IBVS requirements (constant image features stream and zero convergence for errors).

2. Methodology for image analysis algorithm

The methodology for the proposed algorithm for the visual servoing of the robotic assembly line, has two major components. One component is the computing server responsible for the processing of the image analysis routines implemented in Matlab®. The other is the communication server implemented as a Modbus server on M3 PLC (Programmable Logic Controller) of the assembly line. The assembly line is composed by five independent modules (Fig. 1), each one having its own PLC unit. The results calculated by the computing server are sent to the assembly line using the communication server and integrated for the robotic arm.

For the proposed image analysis algorithm, a 5 Mega pixel CMOS sensor is used. The Arecont® IP camera (Fig. 1) is connected to 24 VDC in the module M2 of the assembly line. One socket of M5 PLC is used as Ethernet data connection for the video camera. Physically, the Arecont® video camera is bound to the safety cage of the robotic arm in M3 as an eye-to-hand configuration. The video camera acquires live images of the assembly product parts (type C) placed on a horizontal loading stand. Type C parts, along with type A, B, and D form up a prototype for an assembly product (Fig. 2).

Figure 1 (developed by the authors in [13]) outlines the assembly line, the computing server and points out all the communication server assemblies, the IP addresses, and several useful descriptions. The computing server is represented by a PC with Matlab® R2019a and Matlab® RT v96. The software application that includes the image analysis algorithm for the visual servoing is installed on the PC and process commands from the assembly line through the existing Modbus server. The computing server is connected to the assembly line by Ethernet connection. The communication server or Modbus server is created on M3 PLC. The computing and communication servers work with read and write commands to the Modbus data that are structured as holding registers.

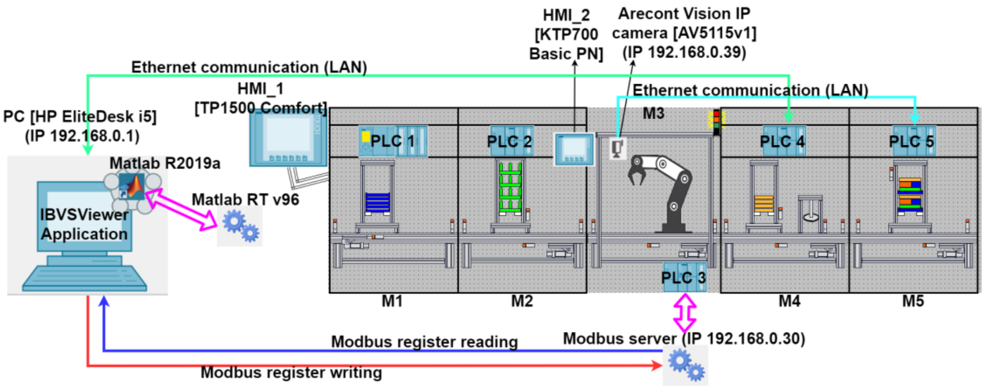


Fig. 1. Outline for the robotic assembly line, the computing and communication servers

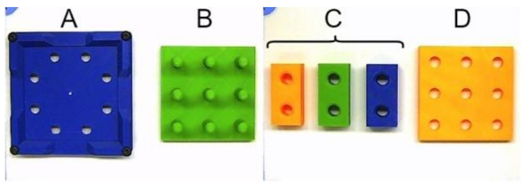


Fig. 2. Type parts for the assembly product prototype

2.1. The computing server for image analysis

The computing server can be viewed as a driven Matlab® application that runs on a dedicated PC and communicates with a Modbus server implemented on the assembly line. This computing server process multiple Matlab® routines, some of them being evaluated in an earlier work [14]. The computing server has four major routines: a calibration routine for Arecont® camera, a primary detection routine for different colors of type C parts, a secondary detection routine for validating of type C parts in the assembly process, and an IBVS routine for computing the parameters of the type C part in order to be manipulated by the robotic arm. Figure 3 presents the IBVSViewer application workflow developed in PhD thesis [15] and [13] along with each action executed by the assembly line, communication server, and computing server, respectively.

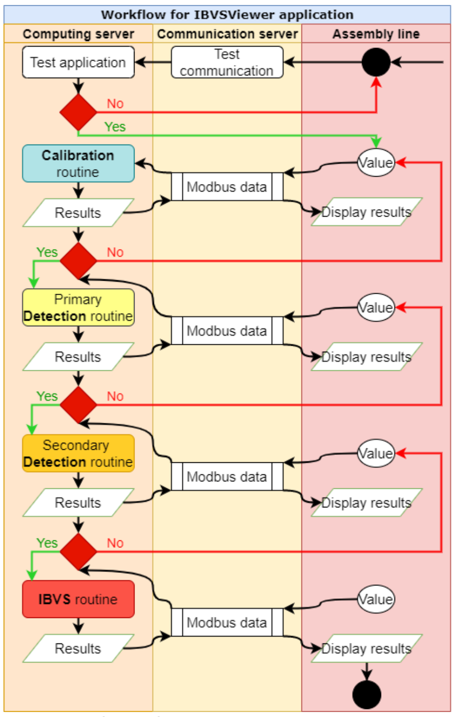


Fig. 3. Workflow for the IBVSViewer application

Starting from the assembly line level (Fig. 3), the communication and the computing servers are tested for activity. If everything is operational, then the calibration routine for video camera can be executed through a request onto the assembly line level using the Modbus data integrated onto the communication server. From this moment on, each executed routine will depend on the results calculated by the previous routine, otherwise the user being notified by warning or error messages. At the end of the IBVSViewer application workflow, the results computed by the IBVS routine are sent to the robot controller unit to be integrated for the assembly process.

The first step in the workflow of the image analysis algorithm for the visual servoing, is the calibration routine. Its purpose is to detect the area of the horizontal loading stand and to isolate it for the rest of the algorithm workflow. A maximum resolution image (2560×1920 pixels) is acquired from the video camera whereupon the white color of the loading stand is segmented. For this purpose, the RGB image is converted to YCbCr color space. A narrow interval of pixel values is stored only from the luminance (Y) histogram, leaving the other two histograms unchanged. The resulted binary image is sent to the calibration routine where a region analysis function is used. The binary image is processed by two types of region analysis properties: area and bounding box. First, only the object with the biggest area value in pixels is found (in the acquired RGB image, having the maximum resolution, there can be other white objects in addition to the loading stand surface), and second, its bounding box represents the entire area of the horizontal loading stand. In this way, the output of the calibration routine is a series of coordinate values along with the width and height of the loading stand.

The next stage of the image analysis algorithm for the visual servoing is directly dependable on the calibration results, so the primary detection routine must have the coordinates of the type C part loading stand. This routine purpose is to find all the assembly type C parts from an acquired scene (image) with the video camera. Just like the previous segmentation procedure, a segmentation for each color of type C parts is produced, now. The same YCbCr color space is used for segmentation and is worth to mention that several color spaces have been assessed for this task. Next, the primary detection routine is applied on the binary image obtained with the segmentation procedure. The same region analysis function is deployed, but only for area property and a list of detected regions (objects) is returned. This list of objects is iteratively cleaned by same color type C parts that are attached to each other or other anomalies that can appear due to illumination differences. Only objects that qualify into a predetermined interval for area value are kept in the final list. The total number of objects in the final list is the number of the detected type C parts for a single color. This procedure is done for all three existing colors of type C parts.

Just like the primary detection routine of the proposed image analysis algorithm, the secondary detection routine needs that the calibration routine to be preliminary executed. Its purpose is to validate the type C parts for the assembly process, considering the rules and constraints of the robot manipulator. The segmentation procedure is the same as previously explained and the region analysis function uses much more properties: bounding box, centroid, orientation, major and minor axis length, and area. Unlike the primary detection routine, the secondary detection routine works with all the type C part colors at once. After the iteratively cleaning process of the detected list of objects and some simple calculus for the orientation property, the secondary detection routine starts a procedure for type C part validation (removing the joined type C parts that the robot manipulator cannot assembly). Thus, in Fig. 4a) a scene acquired with the video camera can be visualized and for each type C part numbered from 1 to 9, there is a robot manipulator mark (red color). This mark is calculated considering the type C part orientation, resulting three situations: a vertical mark (when type C part has a horizontal orientation), a horizontal mark (when type C part has a vertical orientation), and a rotational mark (when type C part has neither vertical nor horizontal orientation). Moreover, one must implement special cases when the shape of the robot manipulator mark is outside of the image space (for example the vertical mark for number 1 and 3 type C parts or rotational mark for number 6 type C part). This procedure is based on the considering that the centroid of the robot manipulator is always on top of the type C part centroid that is next to be assembled. In Fig. 4b is presented the mathematical procedure for calculating the rotational mark of a type C part.

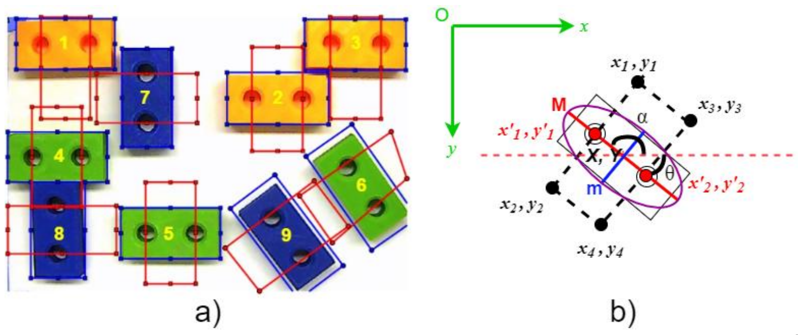


Fig. 4. a) A scene acquired with the video camera along with robot manipulator mark for each type C part; b) The mathematical procedure for calculating the rotational mark

To calculate the rotational mark of any type C part (for example number 6 and number 9 type C parts from Fig. 4a), knowing the major (*M* - red color) and minor (*m* - blue color) axis of the ellipse (purple color) according to [16] that approximates the type C part, the angle θ between the *Ox* and *M* axes [16], α angle (when $\theta < 0^o$, then $\alpha = 180^o + \theta$, otherwise $\alpha = \theta$), and *X* and *Y* (centroid coordinates of the type C part) in Fig. 4b., the following relations are used:

$$\begin{aligned} x_{1,3} &= x'_{1,2} + m \times \cos(\alpha - 90^o), \\ y_{1,3} &= y'_{1,2} + m \times (-\sin(\alpha - 90^o)), \\ x_{2,4} &= x'_{1,2} + m \times (-\cos(\alpha - 90^o)), \\ y_{2,4} &= y'_{1,2} + m \times \sin(\alpha - 90^o), \end{aligned} \tag{1}$$

where $x'_1 = X + M/4 \times \cos\alpha$, $x'_2 = X + M/4 \times (-\cos\alpha)$, $y'_1 = Y + M/4 \times (-\sin\alpha)$, and $y'_2 = Y + M/4 \times \sin\alpha$; and (x_1, y_1), (x_2, y_2), (x_4, y_4), and (x_3, y_3) are the coordinates of the rotational mark points for any type C part.

To calculate the horizontal and vertical marks of the type C parts, one can use only the centroid and bounding box properties of those parts. Thus, the coordinate of the four points of the robot manipulator mark for the type C part are obtained by fractions of bounding box sizes subtracted from the centroid coordinates.

In Fig. 4a, the secondary detection routine finds the intersection area for each robot manipulator mark with each bounding box of type C parts. If there is any intersection area, then a type C part is invalid, as is the case of number 4 for which the calculated mark intersects the number 8 bounding box. All the other type C parts, number 1, 2, 3, 5, 6, 7, 8, and 9 will be valid, since their calculated mark doesn't intersect with any type C part bounding box.

To execute the final routine (IBVS) of the image analysis algorithm workflow for the visual servoing, a segmentation is required only to be applied on a desired color for type C parts. Also, an index number for the previously detected type C part for the desired color should be needed. After these parameters are satisfied (color and index number for the type C part), the IBVS routine can be executed. The region analysis function uses bounding box, centroid, orientation, and area as properties. The output of the IBVS routine is a sequence of parameters for the selected type C part (X and Y coordinates of the centroid and value of θ for orientation) that are used by the robotic arm in the assembly process.

2.2. The communication server

The communication server is a Modbus server created on M3 PLC unit of the assembly line. The Modbus server works with structured data called as holding registers. These registers can hold integer values that can be read or written by the computing server, other assembly line components, and most important, by the robot controller unit. Among the implemented holding registers stands out: the index zero of the holding registers is a "keep alive" value for the computing server; the index one of the holding registers is the execution mechanism for the routines of the proposed image analysis algorithm; other holding registers store the total number of the detected type C parts, the number of orange, green, and blue type C parts, the number of invalid type C parts, the X and Y centroid coordinates, the orientation of the type C part (θ), and other mechanisms, like the control mechanism for the information and error messages.

2.3. The description of the image analysis algorithm for the IBVS

The proposed algorithm for the visual servoing (Fig. 5) can be used in Matlab® session or as compiled executable with Matlab® RT v96. Also, there are two versions of the proposed algorithm: one simpler (Windows® Cmd) with only information and error messages and another version with user interface (Matlab® R2019a App Design). Preferable each version should be used as a compiled executable which ensures a smooth processing time due to faster response of the Matlab® RT then the Matlab® session.

```
Input: Color image at 685×470 pixel resolution according to camera calibration
Output: Type C part parameters (centroid coordinates, orientation, color)
1.  try Modbus connection, catch errors, if any;
2.  initialize infinite loop
```

```
3.      acquire camera frame (user interface version);
4.      show current frame (user interface version);
5.      if camera calibration routine is activated
6.          acquire camera frame at full resolution ($2560\times1920$ pixels);
7.          segmentation of the loading stand of type C parts;
8.          run the calibration routine;
9.          display results (user interface version) and information message;
10.     if primary detection routine for type C parts is activated
11.         check if camera calibration routine has been run;
12.         acquire frame using calibration parameters (no user interface version);
13.         for each type C part color (orange, green, blue)
14.             segmentation of the type C part color;
15.             run the primary detection routine;
16.         display results (user interface version) and information message;
17.         if IBVS routine for type C parts is activated
18.         check if primary and secondary detection, and calibration has been run;
19.         acquire frame using calibration parameters (no user interface version);
20.         check if color and index number for type C part has been chosen;
21.         segmentation of the chosen type C part color;
22.         run the IBVS routine;
23.         if type C part can be manipulated
24.             write IBVS results (centroid coordinates and orientation);
25.             display IBVS results (user interface version) and information message;
26.         if type C part cannot be manipulated
27.             write invalid results;
28.             display results (user interface version) and information message;
29.       if secondary detection routine for the joined type C parts is activated
30.         check if camera calibration routine has been run;
31.         acquire frame using calibration parameters (no user interface version);
32.         run the secondary detection routine;
33.         calculate the invalid type C part number;
34.         display results (user interface version) and information message;
35.     if "Close connection" for Modbus server is activated
36.         break from the infinite loop;
37. end the infinite loop
```

Fig. 5. The proposed image analysis algorithm for the visual servoing of the robotic assembly line

The proposed image analysis algorithm was implemented and tested on a computing platform with Intel i5 (2.53 GHz) and 8 GB RAM. The operating system was Windows 7 Professional. The proposed algorithm integrates ten routines written in Matlab® R2019a. Overall, the algorithm has an $O(N\times\log(N))$ computational complexity. The most computational demanding routine is the secondary detection routine with an $O(N^4\times\log(N))$ computational complexity.

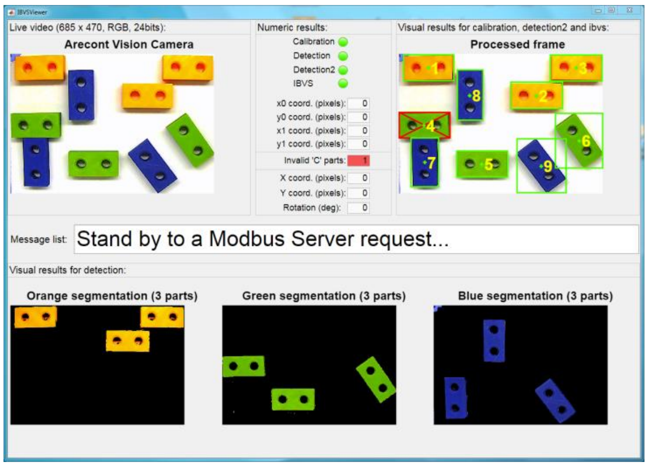
3. Experimental results

In this section, we present the experimental results obtained with the IBVSViewer application that integrates the image analysis algorithm for the visual servoing. The results for the primary and secondary detection routine are presented in Fig. 6. Thus, the results of primary detection are displayed in the bottom of the figure as three type C parts are segmented for orange, green, and blue, respectively. In the top-right corner of figure the results for secondary detection routine are presented. Only one type C part is invalid, namely the number 4 (considering the methodology from Fig. 4a). If the number 2 type C part is considered, the results of the IBVS routine are: the X coordinate and Y coordinate of the centroid are 463 pixels and 141 pixels, respectively and θ is 0^{o}.

In Fig. 7, we present 12 experiments for the secondary detection routine considering different type C part arrangements, since this routine of the proposed image analysis algorithm is the most complex one. In experiment number 3, only three type C parts are valid and the rest of them are not detected in the color segmentation process. This situation appears when multiple type C parts with the same color are joined (two orange, green, and blue parts). Other situations, like experiment number 6, no human intervention is needed since the robot arm can pick first the valid type C parts and after that the invalid type C parts will become valid as the assembly space depletes. This is not the case for experiment number 10, where the human intervention is needed to clear the assembly space for the robotic arm.

Table 1 presents the temporal results for the proposed image analysis algorithm in different configurations: Matlab® application with and without user interface (IBVSViewer and IBVSCmd) and compiled Matlab® application with and without user interface (compiled IBVSViewer and IBVSCmd). The best values for the average execution times from 50 experiments with type C part arrangements using the proposed algorithm for the visual servoing are highlighted with gray color. The best average execution time is obtained for the primary detection routine at 0.074 s, resulting approximately 14 FPS. Of those 50 experiments of type C part arrangements, 13 experiments are presented: one experiment is in Fig. 6 and 12 experiments are in Fig. 7.

Performance indices for the best execution time configuration for the Matlab® application, namely the no user interface compiled IBVSCmd software application, are presented in Table 2. The detection rate of ROI (region of interest for the loading stand surface and the type C parts) and the detection rate of ROI pixels are calculated as a ratio between the positive and the total cases multiplied by 100. The results for the detection rate are obtained using YCbCr color segmentation.


IBVSViewer

Live video (685 x 470, RGB, 24bits):

Numeric results:

Visual results for calibration, detection2 and ibvs:

Calibration	●
Detection	●
Detection2	●
IBVS	●
x0 coord. (pixels):	0
y0 coord. (pixels):	0
x1 coord. (pixels):	0
y1 coord. (pixels):	0
Invalid 'C' parts:	1
X coord. (pixels):	0
Y coord. (pixels):	0
Rotation (deg):	0

Message list: Stand by to a Modbus Server request...

Visual results for detection:
Arecont Vision Camera
Processed frame
Orange segmentation (3 parts)
Green segmentation (3 parts)
Blue segmentation (3 parts)

Fig. 6. The results for the IBVSViewer application

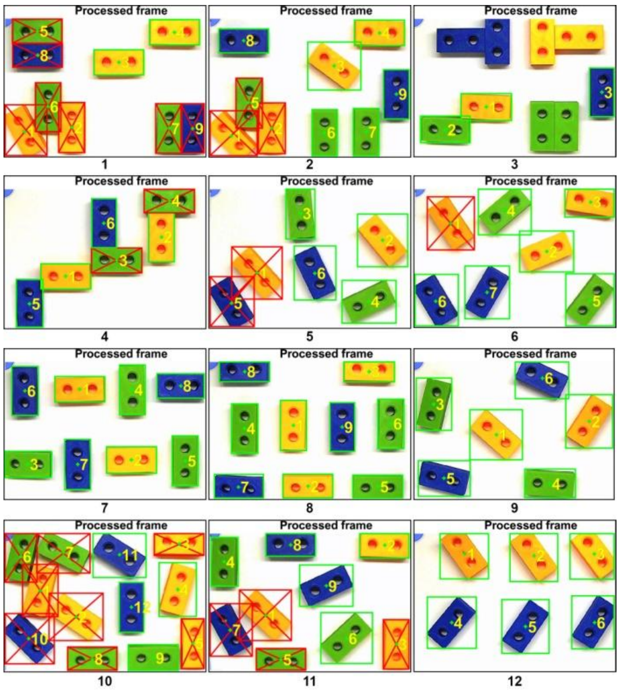


Fig. 7. Results for different type C part arrangements using the secondary detection routine

Table 1

Computation times of the proposed algorithm in different configurations

Routine	IBVSViewer (s)	IBVSCmd (s)	Compiled IBVSViewer (s)	Compiled IBVSCmd (s)
Calibration	0.76	0.13	0.342	0.096
Primary detection	0.43	0.092	0.241	0.074
Secondary detection	0.518	0.232	0.476	0.224
IBVS	0.385	0.128	0.286	0.106

Table 2

Performance indices for no user interface compiled IBVSCmd software application

Routine	Color segmentation	ROI detection rate (%)	ROI pixels detection rate (%)	Experimental conditions
Calibration	White	100	> 95	Uncontrolled light source
Primary detection	Type C parts (orange, green, and blue)	100	> 97	
Secondary detection				
IBVS				

Aside from YCbCr, L*a*b* color space was not able to perform well on illumination differences like day-night cycle or on mixed lighting (natural with artificial). Poor segmentation results have been obtained especially on green type C parts using this color space.

Regarding the performances of the IBVSViewer software application 100% detection rate is obtained for primary detection routine at approximately 14 FPS. In terms of classification, the type C parts are classified based on color segmentation used by the region analysis function to determine the area property. The type C parts are classified by means of a predetermined interval of area value.

The inclusion of the command interface for the computing server, from the local structure of M3, to the M1 general assembly cycle of the robotic assembly line can assure the increase of the computing speed for the command parameters of the robot controller unit. Also, by placing multiple Arecont® Vision video cameras for different stages of the general assembly cycle assures the identification of many assembly type errors. The general assembly cycle is the assembly process of each type parts to obtain the assembly product prototype.

6. Conclusions

Our proposed solution for the image based visual servoing of the robotic assembly line is an easy to implement algorithm, yet a robust and convenient one for further assembly line developments. The proposed image analysis algorithm stands for the part identification process for robotic assembly control. Significant performances were obtained in the part identification process of the image analysis algorithm for the visual servoing of the robotic assembly line due to implementation of the following techniques: color segmentation, region analysis from a binary image, and 2D shape geometrical calculus.

The proposed solution for the visual servoing of the robotic assembly line can be implemented on various industrial scenarios with low development effort. The experimental results validate the IBVSViewer software application, this representing a real-time platform prepared for the assembly scenarios.

This paper submits two contributions: the development of the IBVSViewer software application that includes the image analysis algorithm for the image based visual servoing of the robotic assembly line and the fitting of the mathematical procedure for the type C part validation onto the image analysis algorithm workflow for the visual servoing.

For further developments, the command interface for the computing server can be included in the general assembly cycle to obtain computing speed for the command parameters of the robot controller unit. As well, by commissioning multiple video cameras many general assembly cycle errors can be detected.

Acknowledgment

This work has been funded by the CIDSACTEH project, 78_PCCDI/2018.

R E F E R E N C E S

[1]. *L. E. Weiss*, Dynamic Visual Servo Control of Robots: An Adaptive Image-Based Approach, CMU-RI-TR-84-16, Carnegie-Mellon University, Pittsburgh, 1984.

[2]. *A. Ghasemi, P. Li and W.-F. Xie*, “Adaptive Switch Image-based Visual Servoing for Industrial Robots”, in Inter. Journal of Control, Aut. and Sys., **vol. 18**, 2019, pp. 1324-1334.

[3]. *A. Ghasemi, P. Li, W.-F. Xie and W. Tian*, “Enhanced Switch Image-Based Visual Servoing Dealing with FeaturesLoss”, in Electronics, **vol. 8**, no. 8, 2019, 903.

[4]. *H. Shi, G. Sun, Y. Wang and K.-S. Hwang*, “Adaptive Image-Based Visual Servoing With Temporary Loss of the Visual Signal”, in IEEE Trans., **vol. 15**, no. 4, 2019, pp. 1956-1965.

[5]. *H. Wang, B. Yang, J. Wang, X. Liang, W. Chen and Y.-H. Liu*, “Adaptive Visual Servoing of Contour Features”, in IEEE Trans. on Mechatronics, **vol. 23**, no. 2, 2018, pp. 811-822.

[6]. *Z. Ye, H. Corporaal, P. Jonker and H. Nijmeijer*, Cross-Domain Modeling and Optimization of High-Speed Visual Servo Systems, 15th International Conference on Control, Automation, Robotics and Vision (ICARCV), 2018, pp. 1791-1798.

[7]. *T. Shu, S. Gharaaty, W.-F. Xie, A. Joubair and I. A. Bonev*, “Dynamic Path Tracking of Industrial Robots With High Accuracy Using Photogrammetry Sensor”, in IEEE / ASME Transactions on Mechatronics, **vol. 23**, no. 3, 2018, pp. 1159-1170.

[8]. *A. V. Kudryavtsev, M. T. Chikhaoui, et al.*, “Eye-in-Hand Visual Servoing of Concentric Tube Robots”, in Robotics and Automation Letters, **vol. 3**, no. 3, 2018, pp. 2315-2321.

[9]. *Y. Zhang and S. Li*, “A Neural Controller for Image-Based Visual Servoing of Manipulators With Physical Constraints”, in IEEE Trans. NNL&S, **vol. 29**, no. 11, 2018, pp. 5419-5429.

[10]. *N. Tian, A. K. Tanwani, J. Chen, M. Ma, R. Zhang, B. Huang, K. Goldberg and S. Sojoudi*, A Fog Robotic System for Dynamic Visual Servoing, ICRA, 2019, pp. 1982-1988.

[11]. *A. Ostovar, O. Ringdahl and T. Hellström*, “Adaptive Image Thresholding of Yellow Peppers for a Harvesting Robot”, in Robotics, **vol. 7**, no. 1, 2018, 11.

[12]. *D. Cabecinhas, S. Bras, R. Cunha, C. Silvestre and P. Oliveira*, “Integrated Visual Servoing Solution to Quadrotor Stabilization and Attitude Estimation Using a Pan and Tilt Camera”, in IEEE Transactions on Control Systems Technology, **vol. 27**, no. 1, 2019, pp. 14-29.

[13]. CIDSACTEH Project, STAGE II, 2019, Project 2: Activity 2.4., Romanian Project, UEFISCDI, PN-III-P1-1.2-PCCDI-2017-0290, http://www.cidsacteh.ugal.ro

[14]. *J.-I.-R. Cojocaru, D. Popescu and Loretta Ichim*, Image Based Fault Detection Algorithm for Flexible Industrial Assembly Line, 22nd CSCS, 2019, pp. 541-546.

[15]. *J.-I.-R. Cojocaru*, Sisteme de analiză complexă a imaginii cu aplicații în robotică și agricultură (Image complex analysis systems for robotics and agriculture applications), PhD Thesis, University Politehnica of Bucharest, 2020.

[16]. *R. M. Haralick and Linda G. Shapiro*, Computer and Robot Vision, Addison-Wesley, **vol. 1**, pp. 74-658, 1992.