Open source Platform for Vision Guided Robotic Systems Integrated in Manufacturing

Silviu Răileanu, Theodor Borangiu, Andrei Silişteanu, Silvia Anton, Florin Anton

University Politehnica of Bucharest, Dept. of Automation and Applied Informatics, Bucharest, Romania
e-mail: {silviu.raileanu, theodor.borangiu, andrei.silisteanu, silvia.anton, florin.anton}@cimr.pub.ro

Abstract: The paper describes the design, implementation, testing and validation of an open source machine vision framework based on OpenCV (Open Source Computer Vision) library. This framework was developed for smart manufacturing control. Material conditioning and handling processes involving industrial robots are the processes that benefit from the proposed solution. The solution offers the following functionalities: acquisition of video streams from multiple sources, image analysis, object recognition, localization and interaction with industrial equipment using standard, open communication protocols. The paper covers several design aspects: system architecture, data acquisition and standardization of the image representation to be used by the analysis algorithms and object recognition module, input/output interaction protocols, camera-robot calibration. Results are reported for an implementation of the framework using a commercial image acquisition device and an industrial robot.

Keywords: interaction protocol, industrial robot, visual robot guidance, open source, OpenCV, TCP.

1. INTRODUCTION

Given the importance of robotized solutions in manufacturing (IFR, 2020a), the research and development of vision systems used in robot guidance, material flow monitoring and product control have continuously increased during the last decade (Ford, 2015). These systems work in unstructured shop floor areas containing parts randomly located usually in 2D workplaces and recently in 3D environments, where machine vision assists industrial robots to qualify, handle and sort components of the material flows Kang et al, 2016). There are also applications in which vision systems perform quality control based on part geometry and shape analysis or route products according to their type (Borangiu, 2004).

In the current global economic context, a high number of manufacturing enterprises needing automation cite high investment and operating costs as a major obstacle, while digitalization is perceived as inaccessible (expensive and complex) by many companies (Saam et al., 2016; McFarlane et al., 2020). The scope of this research is to demonstrate that an open source machine vision framework for modern manufacturing with industrial robots can be easily developed from available open source general vision processing libraries and commercial image acquisition devices. It will be demonstrated that this solution has similar characteristics with a commercial one: accuracy, speed, and connectivity with a wide range of sensors, control and computing resources.

The realization of this machine vision framework aims at lowering the total cost of robot-vision projects by reducing to zero the cost of the image processing application and making feasible the use of alternative, possibly non-industrial image acquisition devices that are much cheaper than their industrial counterparts. Currently there are commercial applications which offer similar functionalities (part detection, recognition and location) such as Cognex Vision Pro (cognex.com), MVTec Halcon (mvtec.com), Omron Adept SmartVision MX (omron247.com) but their drawbacks are the high price and the fact that they use specific video input and only a limited number of input devices. Concerning open source solutions, to the best of our knowledge there are currently either generic libraries (OpenCV, Accord) or applications (Matlab) that perform standard computations on images, or applications dedicated to image handling and processing like ImageJ (https://imagej.nih.gov/) or Micro-Manager (https://micro-manager.org/) for laboratory equipment (e.g., control of microscopes). In this context the developed solution offers for industrial applications: a) the needed functionality for part recognition and locating based on its contour shape and body features (area, perimeter, number of holes, blob moments), b) extension and standardization of the input sources, and c) a standardized protocol for the interaction with industrial equipment (e.g., robot controller, programmable controller).

Some external preliminary results consist of existing image processing frameworks (OpenCV (OpenCV, 2020)], Accord.NET (Accord, 2020) as an extension of AForge.NET) which facilitate the operation with different image and video formats, are able to apply standard filters and offer a set of useful functionalities like operation with blobs and shapes (polylines).

These preliminary results represented the starting point of the research, being considered in order to improve the design and accelerate the solution implementation. The novelty consists in the open source nature of the project with all the associated advantages, in the task-, material flow- and environment-orientation of the solution and in the mixed object modelling technique that uses a combination of body features (e.g., invariant moments) and contour shape descriptors.

The article is structured as follows: Section 2 details the components of the machine vision framework, their role and interconnection in the hardware - software platform.

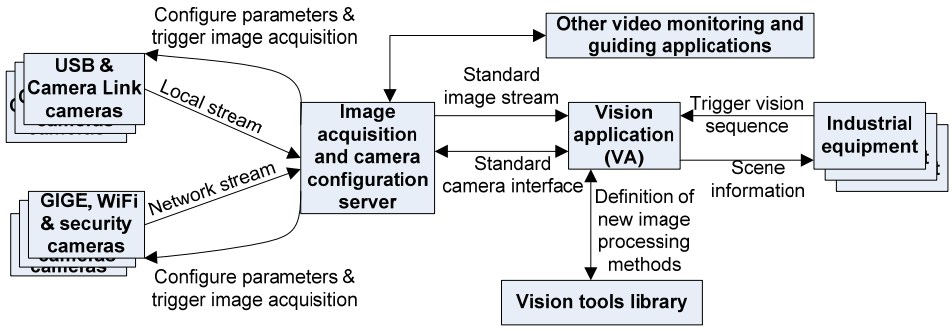

Fig. 1. Machine vision framework structure and information exchange between its components.

Section 3 presents the interaction protocol that controls the remote operation of the vision platform with main types of industrial equipment. Section 4 describes the image acquisition and calibration processes. Section 5 presents a set of experiments for object handling using vision and a comparison with a commercial software. Section 6 formulates conclusions and future research and development directions.

2. STRUCTURE OF THE MACHINE VISION PLATFORM

The proposed machine vision framework performs the following tasks: image acquisition from multiple sources (ranging from widely available inexpensive WiFi cameras to industrial cameras which offer information in different formats), basic image processing (e.g., region of interest (ROI) selection, parameter selection, a.o.), advanced image processing (training and identifying object models and features, object locating), and interconnection with other control applications using standardized industrial communication protocols. The framework was designed to work together with any type of industrial control device to which it provides information about the handled object's dimension (*measurement*), type (*detection* and *recognition*), position and orientation (*locating*). The industrial control devices considered are robot controllers, Programmable Logic Controllers (PLCs), embedded control devices and industrial PCs. The research is validated in conjunction with an industrial robot for material handling tasks. The structure of the generic machine vision framework is shown in Fig.1. It consists of an **artificial vision application** VA (for image processing), a **video server** offering the input to the artificial vision application and a **network communication module** for integration with the control equipment in manufacturing shop floor applications.

The main characteristics of the video server are: i) *streaming*: consists in the conversion of generic streams and image formats into a standardized and open format accepted by the artificial vision application, and ii) *standardization of camera interface*: identification, open/close stream, adjust image size, trigger image acquisition. Industrial cameras have dedicated drivers which limit the number of concurrent clients to one, guaranteeing thus a high frame rate acquisition. By inserting this middle module (the vision server), it will be possible to connect multiple clients (e.g., multiple artificial vision robot guidance applications in order to compare image processing performance, and run in parallel multiple video surveillance and monitoring applications). The stream standardization will not restrict the resolution of the original stream/image, which can be modified through the camera interface protocol. Besides the above characteristics it is also possible to adjust at video server level the region of interest (ROI) from the original input image. Thus, by limiting the dimension of the input data structure (a matrix), the image processing time decreases; a lower processing time is desired in real-time applications such as visual servoing of robot manipulators.

The advantage of operating on a variety of video streams is that it will be possible to compare the performance of the part detection, identification and locating functions and to add new functionalities (e.g., collision avoidance during robot interaction with the material flow) or recommend significant cheaper hardware for the same process automation project. Another advantage of this architecture is that the vision application can be located on a cloud infrastructure offering vision services (based on standardized interaction protocols) to manufacturing resources eliminating thus the cost with dedicated hardware and making it easy to replicate the vision framework for additional industrial projects.

3. INTERACTION PROTOCOL BETWEEN THE VISION PLATFORM AND RESOURCE CONTROLLERS

The interaction protocol provides a standard set of primitives (Table 1), each one with a well-defined semantics (Bellifemine et al., 2007). It is implemented as a socket-based synchronous communication between the vision application and the resource controllers needing information from the visualised scene (e.g., work-place, conveyor belt window). In this configuration the vision application is the server waiting incoming connections, and the resources are the clients. Depending upon the received command a specified process is executed and the information is returned to the client in the form of a response. The robot-vision calibration and object identification and localization commands and responses exchanged between the vision system and a robot controller are defined in Table 1. The commands were chosen for an industrial robot that uses vision to identify and locate parts to handle them with an impactive gripper.

Other primitives were developed: to evaluate features of blobs (unknown regions): scalar - area, perimeter, roundness, hole number, eccentricity; space domain of the boundary - radial signature and of the body - skeleton; to define the parameters of robot grasping models and retrieve them for robot interaction with recognized parts; to define the gripper fingerprint model and use it to authorize collision-free access to objects.

Table 1. Vision application network interface.

Functionality	Type	Description
Locate part	Command	The Locate Parts command is issued by the robot controller in order to start an image acquisition, followed by the identification and localization of a specified part type which was learnt offline (see Section 5). Example: `Locate 1` // Locate the parts which belong to type 1
	Response	This message is sent by the vision application to the robot controller. It consists of a sequence of characters representing the number of parts located followed by their handling location and orientation. This sequence will be analysed by the robot controller in order to pick the parts. Example: `number of recognized objects, for each recognized object (location on X, location on Y, rotation about the Z axis)`
Add calibration point	Command	This command is issued for each point that goes into the calibration process. The command is followed by the location of the point in the robot coordinate system. Following this command, the vision systems searches the predefined calibration target and adds the tuple consisting of the coordinates of the calibration point in the robot frame and in the vision frame to the set of calibration points. It is assumed that a single calibration target is in the image plane. Example: `Store, target position on X in vision frame, target position on Y in vision frame`
	Response	This message is used to inform the robot controller whether the calibration tuple could be added successfully. If no target or more than one target are found an error is issued. Examples: `OK` //The command was executed successfully `ERROR 1` //No target found `ERROR 2` // More than one target found
Delete calibration point	Command	This command requests the elimination of a specified calibration tuple Example: `Delete 1` //Delete tuple with ID 1
	Response	The message is sent in response to a delete command confirming the deletion of the specified tuple.
Compute calibration	Command	This command is issued by the robot controller for the vision system to compute the transformation linking the robot coordinate system to the vision coordinate system (Section 4).
	Response	This message is sent by the vision application to the robot controller. It contains the 3 elements of calibration transformation, namely a translation from the robot plane to the vision plane (distance along X, distance along Y) followed by a rotation about the Z axis. The Z component will be added by the robot by overriding it with the value of the calibration points in the robot coordinate system. Example: `Relative location on X, relative location on Y, relative rotation about the Z axis`
List calibration points	Command	This command requests the listing of all the calibration tuples Example: `List`
	Response	The message is sent in response to a list command. Example For each tuple in the calibration set list `target position on X in vision coordinate system, target position on Y in vision coordinate system, target position on X in robot coordinate system, target position on Y in robot coordinate system`

4. CAMERA AND ROBOT-CAMERA CALIBRATION

To pass from pixels measured by the camera system to millimetres of robot displacements or part geometry description and further from the robot world frame to the camera coordinate system a set of two calibration procedures have been defined. It must be mentioned that:

a) Image processing algorithms were designed to work with black and white images obtained from grayscale images by applying an adaptive binarization threshold;
b) Objects are seen as dark spots (blobs) on a light background;
c) The image plane is parallel with the XOY reference plane of the robot;
d) The camera plane (physical mounting) is also parallel with the XOY world plane of the robot and
e) The end effector is perfectly aligned with the robot's tool control point so that the robot points to the same object location and orientation as the vision system.

The two calibration procedures are described below.

4.1 Camera calibration

Under the assumption that lens distortions do not greatly influence the system accuracy and considering that a pixel is rectangular, the outputs of the camera calibration are the dimensions of the two sides of the pixel's rectangle. In order to keep it simple a calibration target (circular disk) of known radius was used. After target identification and computation of its bounding rectangle the target radius was divided by the number of pixels on X and Y obtaining the two components of the camera calibration (pixel-to-mm and X/Y ratio).

4.2 Robot-camera calibration

Robot-camera (or hand-eye) calibration is the problem of computing the relative transformation between the robot base frame and the image frame.

This transformation is computed offline and used online to determine the location in the robot frame of an object viewed by the camera. Under the assumptions that the robot XOY plane, the image plane and the camera plane are parallel, and using a camera that is fixed relative to the robot base, the transformation has only 4 elements: a translation of the robot frame (dx, dy, dz) followed by a rotation about the Z axis. By knowing the location of the reference point (represented by the calibration target) in both coordinate systems, finding the transformation is similar to solving a 2D puzzle avoiding thus complex matrix computations (Sharifzadeh et al., 2020).

Thus, the robot-camera calibration can be performed in a two-step process using simple geometrical formulas: first the location of the vision plane relative to the robot frame is computed (Fig. 2 up) and then the rotation about the Z axis of the robot is determined (Fig. 2 down). Knowing the location of the reference point in both robot and vision coordinate systems one can plot a set of possible locations for the origin of the vision plane (Fig. 2 up).

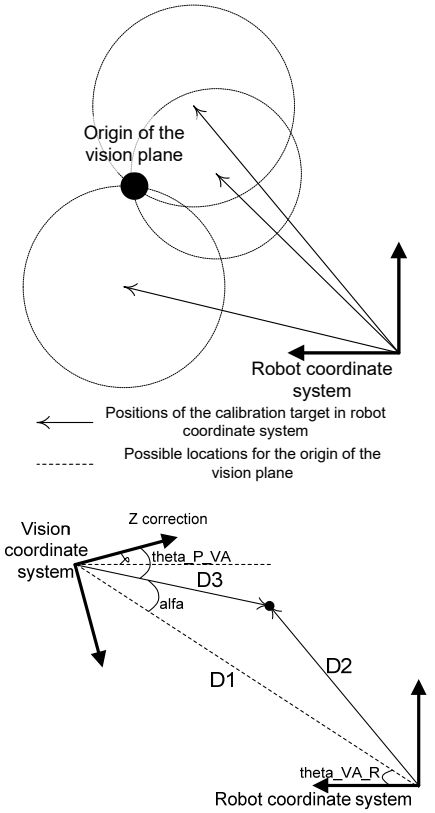

Fig. 2. Finding the location (up) and rotation (down) of the vision plane relative to the robot base frame.

These possible locations are located on circles whose centres are the locations of the associated calibration target in the robot coordinate system and the radius is equal to the distance from the origin of the vision system to the location of the target in the vision frame. Each two circles have two intersection points and theoretically all the circles should have a common intersection point. In practice this is not true because the information comes from measurements affected by errors (e.g., caused by different lightings) resulting thus a zone in space where it is most likely that the vision system's origin is located (Fig. 2 up). For a set of N calibration points, (N-1)*N intersection points are computed; from these computed intersections, those that are grouped are chosen and their average location on X and Y will be the origin of the vision coordinate system. After establishing the origin of the vision frame, the cosine law is applied in the triangle from Fig. 2 down to compute the Z rotation correction. The Z offset will be the Z location of the calibration target.

The minimum set of points (learnt in robot and vision coordinate systems) is 3 in order to obtain a system of three equations with three unknowns (offset X, offset Y, rotation Z). In order to improve the accuracy of the system this minimum number is increased choosing the points as far away from each other as possible. Due to lighting issues, four vision samples rotated with 90 degrees are taken for each robot calibration point and an average location in vision coordinate system is computed as the average of the intersections on X and Y.

For a visually guided robot-object interaction task, the output of the vision system is a unique composite transformation for

each object type and for each robot grasping style (an object can be grasped in several ways, to be learned off line). This transformation translates and rotates the coordinate system attached to the object into the coordinate system attached to the gripping location; the result of this composition is then expressed in the robot frame by the motion control by applying the a priori learned robot-camera calibration transformation.

5. RECOGNITION, LOCALISATION AND DETECTING OBJECT ORIENTATION

In industrial applications the shape of the object is of interest for its recognition (Borangiu, 2004). In this respect there are two recognition techniques: a) matching contour features for greyscale input images, and b) matching blobs operating on black and white (binary) images obtained from greyscale images by applying a threshold that usually optimizes the average contrast curve of the input image (Adept, 2012). The open source vision framework uses the second method.

A common mathematical approach when working with binary images is the use of moment-based analysis (Korta et al., 2014). This method offers information about the area, centre of gravity and orientation of a blob, and the derived image Hu moments (Huang et al., 2010). Since these features are invariant at image translation, scaling and rotation, they can be used to describe the shape which is used for object recognition. These characteristics are of interest for object identification (what type of object), location (where it is positioned) and association to a fixed coordinate system (how is it rotated) allowing thus to compute a unique transformation needed by the robot to handle correctly the object.

For a discrete binary image, the moment of order "p+q" M_{pq} and the central moment μ_{pq} are defined as follows:

$$M_{pq} = \sum_x \sum_y x^p \, y^q I(x,y) \tag{1}$$

$$\mu_{pq} = \sum_x \sum_y (x-\bar{x})^p \, (y-\bar{y})^q I(x,y) \tag{2}$$

where $I(x,y)$ is the pixel intensity at coordinates x, y.

Orientation is defined as the angle relative to the X-axis of an axis through the centre of gravity of the object that gives the lowest moment of inertia (AIM). This value is computed using the formula (3) based on the second order central moments:

$$\theta = \frac{1}{2} tan^{-1} \frac{2\mu_{11}}{\mu_{20} - \mu_{02}} \tag{3}$$

This formula gives the object's direction, which will be taken as X axis. In order to associate a direction to the X axis, we adopt the convention that X is along the longest path of the AIM through the object as depicted in Fig.3.

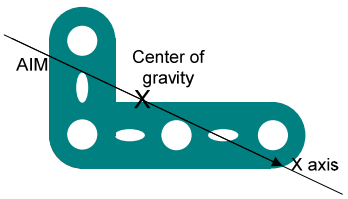

Fig. 3. Object-attached frame for general use.

Derivatives of the image moments which have invariant features with respect to translation and rotation (as is the case of objects situated on a vision plane) can be constructed. Examples of such expressions are given in the work of (Hu Mallick et al., 2020), where 7 such invariants are proposed which can also detect if an object is mirrored.

From the above formulas, of interest are: M00 - equation (4) representing the blob's area, the first order moments that are used to calculate the coordinates of the blob's centre of gravity (5), the second order central moments which are used to calculate the major and minor axes of the blob and thus its orientation (6), and the Hu invariants which together with the blob area are used to define an object prototype (binary model)

$$\text{Area} = M_{00} \tag{4}$$

$$X_{CG}=M_{10}/M_{00}, \quad Y_{CG}=M_{01}/M_{00} \tag{5}$$

$$\text{Orientation} = \theta \text{ if } \mu_{20} > \mu_{02}, \ \theta+90 \text{ otherwise} \tag{6}$$

The object's prototype is defined as the *distance between the Hu invariants* (7) representing the desired shape combined with the *7th Hu invariant* which indicates whether the object is mirrored (essential in the case of robot applications), and the *object area*. Tolerances are defined for each class in order to discriminate between existing objects.

$$D(shape1, shape2) = \sum_{i=1}^{7} |Hu\_invariant_i^{shape1} - Hu\_invariant_i^{shape2}| \tag{7}$$

The computed Hu invariants have a large dispersion being not comparable in magnitude. They were brought in the same range for experiments using eq. (8) (Mallick, 2020).

$$H_i = -sign(Hu_{invariant_i}) * log_{10}|Hu_{invariant_i}| \tag{8}$$

The second component of the prototype model is created from boundary descriptors, obtained in four steps: 1) connectivity analysis; 2) chain encoding of perimeters; 3) approximations of runs with primitive edges; 4) contour approximations with lines and arcs - best fit to primitive edges that approximate the object's boundaries (with a *conformity* tolerance specified by the user). Object identification based on the prototype finder tool is a tree-type, structural recognition process executed at run time as a three-step sequence denoted 5) - 7):

5) *Classification of features*: The vision system classifies all connected edge-edge pairs and associates them with learned feature classes of all prototypes of interest. For an invariant direction of contour inspection (e.g., clockwise), the following feature classes are considered: *line-line*; *line-arc* respectively *arc-line*; *arc-arc*; *line-any* and *arc-any*, respectively *any-line* and *any-arc*. For each feature class of a prototype (edge-edge pair or "corner class") a number of parameters are associated:

- *length* domain for a line in the feature class;
- *angular* domain, *radius* domain, and *convexity/concavity* indication for an arc in the feature class;
- range of accepted variation of the *angle between the edges of a corner* feature

Each corner and each edge in the acquired image will be compared with all feature classes defined during prototyping. Whenever the types of classes match and the edge- and corner parameters are within the accepted limits, the respective corner and edge are placed in a list of matches for that feature class.

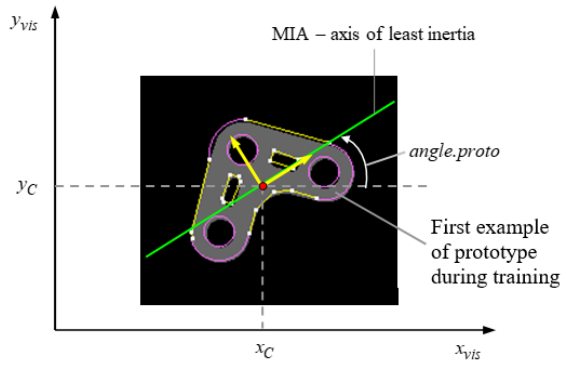

Fig. 4. Best fit line segments and arcs for a gear's boundary. Region 1 (body) is approximated by 7 lines and 3 arcs. Regions 2-6 (holes) are defined by a total of 10 lines and 3 arcs. The object's frame $(x_c, y_c angle.proto)$ is shown, *C* - mass centre.

Each corner and each edge in the input image will be compared with all feature classes defined during prototyping. Whenever the types of classes match and the edge- and corner parameters are within the accepted limits, the respective corner and edge are placed in a list of matches for that feature class.

6) *Prototype-to-image match proposals*: The proposals of prototype-to-image match are founded on the classification of features done in step 5. Feature classes are ordered during the learning stage according to their "uniqueness" and "certainty" (clarity). To recognize objects at run time in this established order, each class will be checked until either all edges in the image were considered or all match proposals were made. Every image- and prototype- feature which were associated to a feature class represents a *match*. However, the proposed matches must be confirmed before a definitive proposal can be formulated and step 7 started. This confirmation is a partial verification which is necessary when more than one prototype feature is associated to one feature class.

7) *Match verification*: Given a single proposal for a prototype-to-image match, this last step verifies the prototype's presence in the image. The prototype's boundary representation is transformed and applied to the image with the translation and rotation that were proposed in step 6, and a search is started for image edges which are aligned with the prototype's edges.

If a "sufficient fraction" of the prototype's boundary is present in the image, the match proposal is accepted. The "sufficient fraction" is defined during the stage of prototype training by the parameter *verify percent*.

Assume that at training time, the prototype's regions and contour edges were assigned weights, respectively $w_{Ri}, 1 \leq i \leq r$ for weighting its *r* regions, and $w_{lj}, 1 \leq j \leq N_i, 1 \leq i \leq r$ for weighting the N_i edges of each region *i*. Then, the weighted length of the prototype's boundary is:

$$weighted\_length = Lw = \sum_{i=1}^{r}[w_{Ri} \cdot \sum_{j=1}^{N_i}(w_{lj} \cdot l_j)] \quad (9)$$

If the assigned verify percentage [%] is *V*, then the "sufficient fraction" of the boundary's weighted length in the image which must verify the prototype to accept the match proposal is:

$$Verify\_length = Lv = V \cdot Lw \quad (10)$$

The membership of a visualized object to a trained prototype is established at run time by checking the body feature and contour descriptor matches parameterized with *conformity* and *verify percentage*; these parameters are configured by the user off-line function of the environment conditions (e.g., lighting).

The computation of the object's position and orientation is done by analysing the image's spatial moments. The object's position (x_c, y_c) and orientation $\angle(\text{MIA}, x_{vis})$ are accurately computed because they influence the repeatability of the robot-vision system since the grasping point will be computed starting from its location (centre of gravity) shifted with the offsets of the learned grasping model: $x_{off}, y_{off}, rz_{off}$ (Anton et al., 2017).

For a given object class there can be defined multiple grasping styles that comply with current handling constraints or from which the vision application selects a collision-free one in the case objects are close to each other in the scene and the gripper might collide with other parts in the neighbourhood.

From OpenCV the authors have used base image processing functions and the contributions are to the implementation of the main process flows (ROI select, thresholding, object learning, ETHERNET integration, automated recognition), the model definition based on region inspection and the interaction protocol.

6. EXPERIMENTS

Experiments have been done using an industrial robot from Adept (Cobra s850) working with the proposed machine vision framework and with Adept Sight commercial vision software to compare the two vision solutions. The proposed vision platform was fed with images from a smartphone while the commercial application took images from an industrial camera, both devices capturing the same scene. The smartphone was used to prove a concept: cheap sensors and electronics (embedded in a proper case with proper power supply) can be used in industrial applications with the same proprieties as industrial cameras. We evaluated the accuracy of the calibration processes and the robot-vision systems, and the runtime of the vision applications. A circular target has been used for camera calibration in the proposed machine vision framework producing a pixel (width, length) = (0.8, 0.8), while the camera calibration utility of Adept Sight produced a pixel (width, length) = (0.6, 0.6).

The second experiment concerned the robot-camera calibration described in section 4. A set of four points from the vision plane have been located four times (each location with the object rotated with 90°). Table 2 on lists the calibration points in the robot and vision coordinate systems, while Table

2 bottom presents the possible locations of the vision frame. Each information is presented in Fig. 5. The computed deviation for the robot-camera calibration process is 0.42mm (on X axis), 0.82mm (on Y axis), 0.531º (about the Z axis).

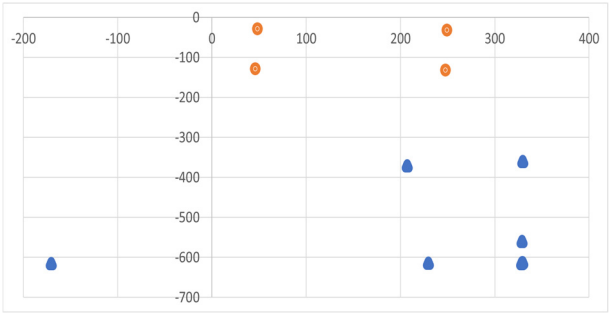

Fig. 5. Robot-camera calibration points in robot frame.

Table 2. Set of calibration points (top) and candidate vision frame points (bottom).

Robot coordinate system		Vision coordinate system	
X (mm)	Y(mm)	X (mm)	Y(mm)
79	-586.545	248	-33
79	-586.545	250	-33
79	-586.545	249	-32
79	-586.545	250	-32
79	-586.545	249	-32
279	-586.545	48	-29
279	-586.545	49	-29
279	-586.545	48	-29
279	-586.545	49	-29
79	-486.545	247	-132
79	-486.545	247	-132
79	-486.545	249	-132
79	-486.545	249	-132
279	-486.545	46	-129
279	-486.545	46	-128
279	-486.545	46	-128
279	-486.545	47	-129

Vision coordinate system	
X (mm)	Y(mm)
328.9	-613
328.9	-560
328.6	-615.4
-170.5	-615.4
328.8	-613.7
207.2	-370.3
328.5	-613.8
229.5	-613.8
329.7	-613.3
329.7	-359.7

The third experiment concerned the accuracy in the process of determining the location of a known point in the robot frame using the location of the object in the vision system and the robot-camera transformation determined in the previous experiment. Using a set of 100 randomly generated points in the vision plane, the robot placed the testing object at the designated position and its location was computed using the proposed vision system. The average deviation is 0.71 (on the X axis), and 0.33 (on Y axis). To correctly handle the objects this deviation/positioning error should be less than the gripper opening which was the case for the testing scenario (10 mm gripper opening).

In the fourth experiment we tested how the components of the prototype model, defined in section 5, vary between different model classes, different locations and different positions.

The measurements have been made for the image shown in Fig. 6 which contains both objects from different classes and objects from the same class (instances of a model). The results for an offline measurement are offered Table 3.

By analysing the results in Table 3 it can be concluded that the membership of an object to a trained prototype can be established by computing the conformity and verify percentage values for each component of the model.

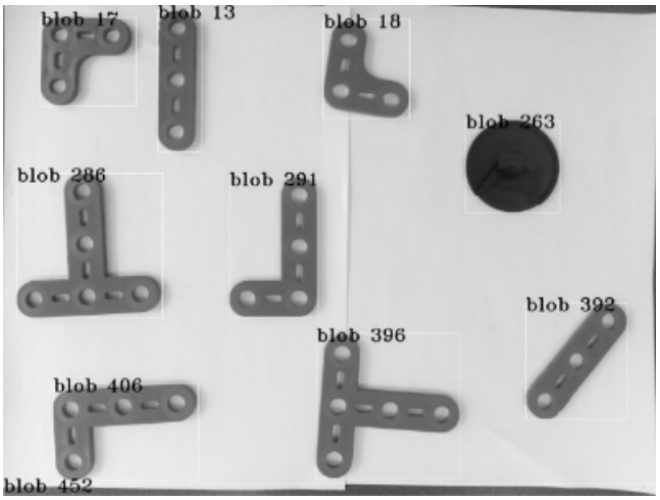

Fig. 6. Test image for computing prototype body components.

Concerning the runtime, the time between the issuing of the location command and the movement of the robot is less than 95 ms. The same image plane foreground was processed by the commercial application in 50msec.

7. CONCLUSIONS AND FUTURE RESEARCH

The objective of the paper is to describe an open source machine vision framework which operates with off-the-shelf equipment in manufacturing and robotics applications and to demonstrate that the performance of the system is similar with commercial industrial vision software. The methodology adopted for object recognition uses a set of Hu invariants of the body shape combined with boundary descriptors that create the object's prototype; blob representations of the foreground components of images are matched at search time with the prototypes of all classes of objects of interest in an application. The camera-robot calibration principle was presented and its results were detailed in the experiments section.

Table 3. Blob characteristics for objects in Fig. 6.

ID	type	area	Hu0	Hu1	Hu2	Hu3	Hu4	Hu5	Hu6
13	I	126787	0.52	1.18	4.35	4.78	9.35	5.40	-10.20
17	r	124837	0.67	2.05	2.41	3.61	-6.71	-4.69	-6.87
18	r	113107	0.65	1.98	2.37	3.56	-6.55	-4.56	-7.01
263	circle	171358	0.80	4.79	5.78	9.34	-16.91	-11.74	-17.70
286	T	221521	0.56	2.26	1.74	3.23	5.72	4.37	6.39
291	nonL	150324	0.49	1.32	1.93	2.78	5.81	4.21	5.14
392	I	117118	0.51	1.16	5.12	5.54	10.88	6.14	-11.71
396	T	202846	0.55	2.28	1.68	3.12	5.52	4.26	-6.40
406	L	167974	0.50	1.34	1.96	2.79	5.56	3.94	-5.20
452	rest	131626	-0.82	-1.64	-0.94	-0.94	-1.88	-1.76	-2.94

The functionalities of the proposed vision framework can be configured according to the characteristics and requirements of industrial applications of robotics and manufacturing (IFR, 2020b):

- *task orientation*: some parameters of the vision application (VA in Fig. 1) can be activated or configured according to the needs of the visual control;
- *adaptation to the characteristics of the material flow*: part ordering in unstructured scenes; identifying touching and overlapping objects; sorting partially occluded/incomplete and randomly arriving objects in the ROI;
- *adaptation to the environment*: periodic update of optimal image acquisition and transformation parameters, such as single or multiple binarization thresholds, filter values, a.o.

According to specialists in machine vision field (Lückenhaus , 2020) from the automotive sector some of the main requirements of a vision platform are: robustness, reliability, competitive price, long-term availability of the machine vision products and better integration and closer connection of machine vision components with programmable logic. Compared with an industrial solution our open source solution is better in terms of price (open source software with off the shelf cameras), long-term availability (the libraries and methods will be available in time, while hardware will even improve) and interconnection (open interconnection standards). Concerning robustness and reliability the application has been tested only in laboratory, but the results are promising for industrial usage.

Future work will be oriented towards extending the machine vision framework from 2D to 3D object recognition, locating and measurements, and defining new interaction protocols with PLCs controlling the material- and operation flows in unstructured manufacturing workplaces: vision-assisted part assembling, 3D geometry measurements, bin picking and developing a cheap acquisition device with lens.

8. ACKNOWLEDGEMENT

The work has been funded by the Operational Programme Human Capital of the Ministry of European Funds through the Financial Agreement 51675/09.07.2019, SMIS code 125125.

REFERENCES

Adept (2012). AdeptVision User's Guide Version 12.1, Part No. 00962-01330, Adept Technology Inc., San Jose, CA

Anton, F.D., Borangiu, T., Anton, S., Raileanu, S. (2017). Using Cloud Computing to Extend Robot Capabilities, RAAD 2017, Turin, Italy, 21-23 June, 2017

Bellifemine, F., Carie, G., Greenwood, D. (2007). Developing multi-agent systems with JADE, Wiley, ISBN 978-0-470-05747-6

Borangiu, T. (2004). Intelligent Image Processing in Robotics and Manufacturing, Romanian Academy Press, Bucharest, pp. 1-650, ISBN 973-27-1103-5

Company N (2019). Increase in small-batch production productivity using robots, Case Studies Industrial, International Federation of Robotics, https://ifr.org/ifr-press-releases/news/increase-in-small-batch-production-productivity-using-robots, consulted in August 2020

Ford M. (2015). Rise of the Robots: Technology and the Threat of a Jobless Future, Oneworld Publications, ISBN 978-0465059997

Huang, Z., Leng, J. (2010). Analysis of Hu Moment Invariants on Image Scaling and Rotation. Proc. of 2nd Int. IEEE Conf. on Computer Engineering and Technology (ICCET'10), pp. 476-480, Chengdu, China

International Federation of Robotics (2020) https://ifr.org/, consulted in April 2020

Kang S., Kim K., Lee J., Kim J., Robotic vision system for random bin picking with dual-arm robots, MATEC Web of Conferences 7, 2016, doi: 10.1051/matecconf/20167507003,

Korta, J., Kohut, P., Uhl, T. (2014). OpenCV based vision system for industrial robot-based assembly station: calibration and testing, PAK Vol. 60, no. 1/2014

Lückenhaus M., MVTEC Software GMBH, The Role of Machine Vision in the Automotive Industry, www.photonics.com/Articles/The_Role_of_Machine_Vision_in_the_Automotive/a58196, consulted in September 2020

Mallick, S., Shape Matching using Hu Moments (C++/Python), www.learnopencv.com/shape-matching-using-hu-moments-c-python/, 2018, consulted in April 2020

McFarlane, D., Ratchev, S., Thorne, A., Parlikad, A.K., Silva, L., Schonfuss, B., Hawkridge, G., Tlegenov, Y. (2020). Digital Manufacturing on a Shoestring: Low Cost Digital Solutions for SMEs, Service Oriented, Holonic and Multi-agent Manufacturing Systems for Industry of the Future. Proc. SOHOMA'19, SCI, Vol. 853, Springer

Open source computer vision - OpenCV, https://opencv.org/, consulted in April 2020

Saam, M. Viete, S., Schiel, et al. (2016). Digitalisierung im Mittelstand: Status Quo, aktuelle Entwicklungen und Herausforderungen ('Digitalisation in SMEs: status quo, current trends and challenges' - our translation, in German only), research project of KfW Group

Sharifzadeh, S., Biro, I., Kinnell, P. (2020). Robust hand-eye calibration of 2D laser sensors using a single-plane calibration artefact, Robotics and Computer-Integrated Manufacturing, Vol. 61, Feb. 2020

The Accord.NET Image Processing and Machine Learning Framework, http://accord-framework.net/index.html, consulted in April 2020