*Research Article*

INTERNATIONAL JOURNAL OF ADVANCED ROBOTIC SYSTEMS

*International Journal of Advanced Robotic Systems*
November-December 2017: 1–10
© The Author(s) 2017
DOI: 10.1177/1729881417738884
journals.sagepub.com/home/arx
SAGE

# A machine learning-based visual servoing approach for fast robot control in industrial setting

**Francesco Castelli, Stefano Michieletto, Stefano Ghidoni and Enrico Pagello**

**Abstract**

Industry 4.0 aims to make collaborative robotics accessible and effective inside factories. Human–robot interaction is enhanced by means of advanced perception systems which allow a flexible and reliable production. We are one of the contenders of a challenge with the intent of improve cooperation in industry. Within this competition, we developed a novel visual servoing system, based on a machine learning technique, for the automation of the winding of copper wire during the production of electric motors. Image-based visual servoing systems are often limited by the speed of the image processing module that runs at a frequency on the order of magnitude lower with respect to the robot control speed. In this article, a solution to this problem is proposed: the visual servoing function is synthesized using the Gaussian mixture model (GMM) machine learning system, which guarantees an extremely fast response. Issues related to data size reduction and collection of the data set needed to properly train the learner are discussed, and the performance of the proposed method is compared against the standard visual servoing algorithm used for training the GMM. The system has been developed and tested for a path following application on an aluminium bar to simulate the real stator teeth of a generic electric motor. Experimental results demonstrate that the proposed method is able to reproduce the visual servoing function with a minimal error while guaranteeing extremely high working frequency.

**Keywords**

Visual learning, visual servoing, learning and adaptive systems, computer vision, robot programming by demonstration, Gaussian mixture model, visual control of robotic systems, sensor-based control

Date received: 6 February 2017; accepted: 4 July 2017

Topic: Special Issue – Robot Perception for Manufacturing
Topic Editor: Emanuele Menegatti
Associate Editor: Emanuele Menegatti

## Introduction

Robots currently operating in production plants are not equipped with perception systems (except for the mandatory safety systems). However, perception in industry is a very active research field, as it is an enabling technology for developing human–robot interaction in production plants, and more flexible production processes, leading to large-scale customization, which is one of the frontiers of the Industry 4.0 revolution.

The concept of Industry 4.0 revolution is already well established in the main technological advanced countries. Europe is moving in this direction by spending many efforts and resources with high priority. In particular, according to the pillars in which Industry 4.0 is based,[1]

Department of Information Engineering (DEI), Intelligent Autonomous Systems Lab (IAS-Lab), University of Padova, Padua, Italy

**Corresponding author:**
Stefano Michieletto, Department of Information Engineering (DEI), Intelligent Autonomous Systems Lab (IAS-Lab), University of Padova, Via G. Gradenigo, 6/B, I-35131 Padua, Italy.
Email: michieletto@dei.unipd.it

an objective is the promotion of the connection and cooperation between research community and industries, also by means of open projects. This work is part of one of these projects where our research activity is merged with the know-how of the industrial partners in order to solve real industrial problems.

The starting point is the electric engine manufacturing: this production needs a metal wire to be wrapped around a metal component, namely the stator. Automating the wrapping process has a strong technological and economical impact, since it is currently completed by humans for slots of up to 100,000 pieces. Indeed, this is a time-consuming task, and even small deviations from the optimal wrapping process have negative effects on the performance of the engine produced. In this work, we propose an early implementation of an advanced visual servoing based on machine learning for automatically deploying the copper wire around each stator tooth by means of a lightweight collaborative robot provided by a custom tool and just a single monocular camera as perception device. We simplify the problem by simulating the small space between two engine teeth with the U junction slot in a standard aluminium profile. They are very similar in shape, and we can extend experiments to engines of larger dimensions using a longer bar, instead of developing an ad hoc engine for our tests. Moreover, slots in aluminium bars are not as narrow as tooth gaps, giving us the perfect early stage setup.

Visual servoing is widely used while manipulating or inspecting objects. This tool can be decoupled into two modules: one is in charge of perception and the other handles robot motion. The main advantage is represented by the continuous feedback provided by the sensor exploited for driving the robot motion, following a control law which relates robot kinematics and object tracking. This mechanism provides several advantages: first, it offers the capability of recovering from motion drifts and inaccurate motion planning (provided by the feedback structure); second, the measurements provided by the sensor have increased accuracy as long as the robot gets closer to the object – this happens when the sensor is mounted on the robotic arm, which is very often the case when dealing with visual servoing. However, the direct feedback of the vision sensor on the robot motion often becomes a weak spot of visual servoing systems. Image processing algorithms are often computationally intensive, and they are the main cause of latency in the system, which limits the refresh rate of the commands provided to the robot. Such phenomena are particularly undesirable for robots, which are inherently real-time systems. To better understand the difference in the time scales of perception and action, consider that fast computer vision algorithms analysing standard-size images could have running frequency lower than 10 Hz, while research and industrial robots can accept commands with a frequency significantly higher than 100 Hz. As the image processing block required by visual servoing is usually the bottleneck of this kind of systems, optimization on this side leads to strong benefits.

We developed a visual servoed path following system in order to scan the slot of the extrusion bar for simulating the wire deployment in the real stator teeth. We took care that the tool pin was kept continuously inserted in the gap at a fixed height and orientation, and avoid collisions ensuring an high control rate.

This work tackles the latency introduced by the sensory system by proposing an alternative method for driving the robot motion starting from images. Several possible choices are available to decrease the processing time: the most straightforward choice is code optimization, which can lead to sensible improvements, but it is not the best choice if the required improvement is by an order of magnitude. A second option is graphics processing unit (GPU) processing, which can provide huge improvements by one or more orders of magnitude, but requires dedicated hardware and software; more importantly, the speed-up strongly depends on the algorithm structure and which portions can be parallelized. A third option is hardware implementation, which, however, suffers from strong flows, the main ones being low flexibility and a strong hardware design effort, with high costs.

Our innovative approach overcomes the bottleneck of the general class of image-based visual servoing (IBVS) systems using a different technique. Since the time scales of image processing and robot control in an IBVS system typically differ for an order of magnitude, the solution proposed to reduce the processing time is to substitute the image processing module by learning the visual servoing function. A machine learning algorithm is trained using a small portion of the image and the corresponding robot control command, to be considered, respectively, as input and output data in the testing phase. The model chosen in this study is a Gaussian mixture model (GMM). After training, the model can be used in place of the IBVS system: this novel way of exploiting machine learning for synthesizing an IBVS system leads to a strong decrease of the processing time, bringing the working frequency of the visual servoing system close to the typical values of robotic systems.

The remainder of this article is organized as follows. In 'State of the art' section, related work on visual servoing and machine learning for robot control is discussed; details on our approach will be given in 'System' section, and the results of our tests, both in terms of precision and speed-up, will be detailed in 'Experiments and results' section. Final remarks and conclusions are reported in 'Conclusions' section.

## State of the art

In the context of IBVS, the literature shows different attempts to avoid image processing and feature tracking for increasing the processing speed. Direct visual servoing approaches addressed this issue by considering the image

as a whole: the task is defined as a minimization problem between the current and the desired image.[2] An interesting alternative is represented by the introduction of new visual features based on luminance.[3] The approach called photometric visual servoing[4] is mainly based on the simple extraction of the image gradient, consequently removing any other necessity for image processing. The main drawbacks of photometric visual servoing are the strong nonlinearities in the system dynamics: to address this problem, the use of photometric moments has been introduced. Moments capture the characteristics of an unknown distribution. They have been applied to image intensity in order to obtain a large convergence domain[5] and extended including spatial weights to contrast the disappearance of portions of the scene.[6] A second way to increase the convergence domain of the photometric visual servoing is to represent each intensity pixel as a Gaussian distribution extending its influence to the neighborhood.[7] The goal is to minimize the difference between the desired Gaussian mixture and the current one. Photometric and photometric moments are geometrical interpretations of a more general class of methods called kernel-based visual servoing[8] in which spatial sampling functions called 'kernels' are used to find so-called abstract visual features.

Another recent technique exploits mutual information,[9] defined as the quantity of information shared by two signals (images in our case) to align two images to be robust to appearance variations. On the contrary, our approach does not properly belong to the class of direct visual servoing. Indeed, it relies on a machine learning algorithm to model the control law implemented by a 'traditional' feature-based approach. The idea is very similar to the one proposed by Hafez et al.,[10] where the feature tracking step has been removed by modelling image features with a Gaussian mixture. We expanded this concept by removing both image processing and feature tracking from the loop, moving to a completely different direction with respect to other works aiming at the same objective.

On the other hand, Hafez et al.'s work[10] is not the only attempt of using machine learning for improving robot motion control. Reinforcement learning (RL) based on neural networks (NNs) has been used to enhance visual servoing for a manipulator in the case of visibility problems, incorrect calibration parameters, white noise and modelling errors[11] or for reducing the required amount of information in tasks, such as reaching and grasping.[12] In a different paper,[13] an adaptive distributed fuzzy proportional–derivative (PD) controller served as a map between the image error vector and the joint velocities of the robot. This avoided the need to compute both pseudo-inverse robot Jacobian and inverse interaction matrix, yet maintaining the image processing phase. With Sadeghzadeh et al.,[14] again the available information is limited and the system is able to learn online new tasks by means of fuzzy NNs and RL.

In a very recent work,[15] deep learning has been used to learn hand–eye coordination for grasping purposes. The approach consists of two parts: one predicts the probability of success of a certain command from the camera image and the other functions as the continuous servoing. This approach substantially removes any kind of image process and feature tracking using a huge amount of data (more than 800,000 grasp attempts) for training the deep convolutional NN driving the algorithm. Nevertheless, the actual aim of this work was a task generalization more than a computational redaction. Indeed, no analysis has been performed regarding control rate or responsiveness of the system.

With respect to the majority of these works, our working setup is restricted and we already know the essential information about the environment in advance. Our key contribution consists in having the entire visual servoing process replaced by a probabilistic framework for improving the robot control rate and with only a limited number of examples coming from a traditional visual servoing at disposal to train the model acting as control law.

## System

As outlined in 'Introduction' section, this study aims at creating a visual servoing system based on machine learning – this offers the major advantage of a much faster processing and increased reactivity while keeping precision at almost the same level of a traditional visual servoing used for training the system.

Our system is divided into two modules: an offline phase and an online phase. The offline phase exploits an IBVS system running on a custom data set, in which the cavity in the middle of an extrusion bar is visible, as shown in Figure 1. The robot task is to follow the cavity. Such task requires a high precision, since the bar cavity is narrow.

We recorded several runs with the robot following the cavity driven by IBVS: acquired images and resulting velocities were used to train the machine learning framework acting as visual servoing. A GMM has been used for data representation, while Gaussian mixture regression (GMR) has been exploited for computing the output during the online phase. We selected such framework mainly, because Gaussian mixtures generally need a smaller number of examples with respect to other techniques in order to obtain significant results. Moreover, the regression process is very fast, which is crucial to achieve a high frame rate in the whole process and therefore a higher robot control rate.

### *Visual servoing*

We considered the problem of scanning a gap on a straight object with a camera mounted on the robot end-effector. The control of the robot has been performed through a visual servoing approach with an eye-in-hand configuration,[16,17] assuming the camera frame defined as in Figure 1.

The desired pose of the camera with respect to the bar is with the $z$-axis perpendicular to its direction, at a fixed distance and the scanning direction parallel to the $y$-axis,

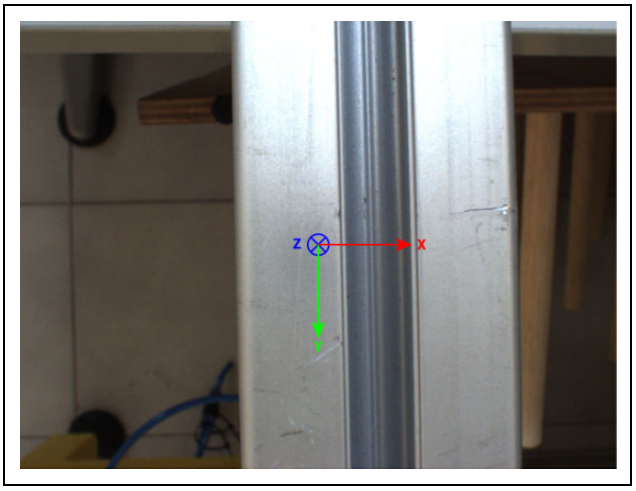


**Figure 1.** Image of the bar taken from the camera mounted on the robot tool center point (TCP). We consider as the camera frame the one fixed in the center of the image, with the *z*-axis coincident with the optical axis of the camera and the *x*- and *y*-axis parallel to the horizontal direction and vertical direction of the view, respectively.

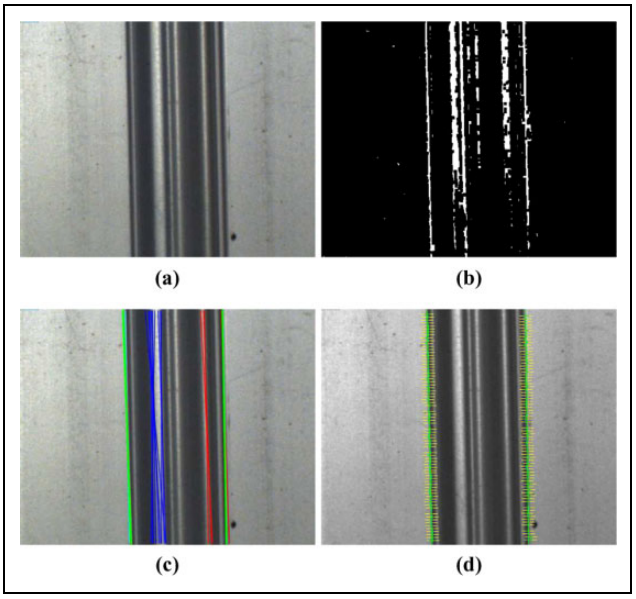


**Figure 2.** (a) Source image. (b) Edges image. (c) Clusters of lines (red, blue) and the final selected lines (green). (d) ViSP moving edge tracker result. ViSP: visual servoing platform.

going towards the negative values. In this configuration, the $x$-axis will be always perpendicular to the bar direction. These choices do not affect the generality of the system.

We define, with $\xi \in \mathbb{R}^6$, the actual camera configuration in the Cartesian space and, with $\mathbf{s} \in \mathbb{R}^m$, a set of $m$ scalar values representing a parametrization of the image features, which can be extracted from the scene, such as points, lines and ellipses. Camera configuration and image features representation are related by a general non-linear function $\Gamma(\cdot)$ as in (equation (1))

$$\mathbf{s} = \Gamma(\xi) \tag{1}$$

The main purpose of visual servoing is to provide a closed-loop control law to drive the camera from $\xi$ to a desired pose $\xi^*$ in such a way, the image feature parameters assume the desired values $\mathbf{s}^*$. In general, this is achieved by relating the camera displacement $\mathbf{v_c} \in \mathbb{R}^6$, that is, three linear velocities $v_x, v_y$ and $v_z$ and three angular velocities $\omega_x, \omega_y$ and $\omega_z$, to the visual features error $\mathbf{e} = \mathbf{s} - \mathbf{s}^*$, according to the general formulation[18]

$$\dot{\mathbf{e}} = L\mathbf{v_c} \tag{2}$$

where $L \in \mathbb{R}^{m \times 6}$ is the interaction matrix built from the knowledge of the visual features in the 3D space and the actual values of a parametrization of these visual features in the image plane. We implemented an IBVS scheme[18] by exploiting the tools provided by an open-source visual servoing library for C++, namely visual servoing platform (ViSP).[19,20] In this case, we selected two line features corresponding to the two edges of the object cavity of Figure 2(a), represented with the common 2D formulation

$$\mathcal{L} = \{(x,y) \in \mathcal{I} : x\cos\theta + y\sin\theta - \rho = 0\}$$

where $\theta$ denotes the angle with the $x$-axis, while $\rho$ defines the distance of the line from the origin. This convention provides a subsequent parametrization $\mathcal{L}_i(\rho_i, \theta_i)\, i = 1.2$. Since $m = 4$ in our case, $L$ is formed by two stacked blocks as the one in (equation (3))

$$L = \begin{bmatrix} \lambda_\rho \cos\theta & \lambda_\theta \cos\theta \\ \lambda_\rho \sin\theta & \lambda_\theta \sin\theta \\ -\lambda_\rho \rho & -\lambda_\theta \rho \\ (1+\rho^2)\sin\theta & -\rho\cos\theta \\ -(1+\rho^2)\cos\theta & -\rho\sin\theta \\ 0 & -1 \end{bmatrix} \tag{3}$$

$$\lambda_\rho = -\frac{A}{D}\rho\cos\theta - \frac{B}{D}\rho\sin\theta - \frac{C}{D}$$

$$\lambda_\theta = -\frac{A}{D}\sin\theta + \frac{B}{D}\cos\theta$$

where $A, B, C$ and $D$ are the coefficients of a 3D plane containing the line. In order to ensure the exponential decoupled decreasing of the positioning error $\mathbf{e}$, the camera velocity can be expressed as

$$\mathbf{v}_c = -\lambda L^{\dagger}\mathbf{e} \tag{4}$$

where $\lambda$ is a proportional gain involved in the exponential convergence of $\mathbf{e}$ and $\dagger$ denotes the pseudo-inverse. $L$ is a time varying matrix and depends on both 3D and 2D values of the visual features; since we do not operate any depth estimation, $L$ is approximated using the desired values of the features. This approximation presents desirable

properties only in an area nearby the convergence configuration. The convergence region is restricted and the trajectory is not optimal. This solution is suitable for our case, since we start from a solution nearby convergence. Further details concerning the interaction matrix approximation can be found in Chaumette and Hutchinson.[21]

Since we have a contour following problem, the camera motion must be controlled in order to follow the desired path, which should go along the two lines. Our main goal is to maintain the camera aligned with the gap and move along the bar for its whole length with respect to the camera's $y$-axis, and therefore, the $y$ velocity component $V_y$ is controlled depending on the features error $\mathbf{e}$. In particular, $V_y$ is forced to be linearly dependent by the sum of squares of the error $\mathbf{e}$, since it is not involved in the alignment task. The formulation is

$$V_y = V_{\max}\left(1 - \frac{\|\mathbf{e}\|^2_{\text{sat}}}{\|\mathbf{e}\|^2_{\max}}\right) \tag{5}$$

where $\|\mathbf{e}\|^2_{\max}$ denotes the maximum allowed error (in sum of squares) for the forward motion of the camera, and $\|\mathbf{e}\|^2_{\text{sat}}$ is the output of a saturation function

$$\|\mathbf{e}\|^2_{\text{sat}} = \max\{\min\{\|\mathbf{e}\|^2, \|\mathbf{e}\|^2_{\max}\}, \|\mathbf{e}\|^2_{\min}\}$$

with $\|\mathbf{e}\|^2_{\min} = 0$ the error value for which the velocity must be maximum. We selected this solution in order to maintain a linear velocity between minimum bounded error and a saturation value. More complex laws can be found in the study by Marchand.[19]

Actual values of the visual feature parameters $(\rho_i, \theta_i)$, $i = \{1, 2\}$ needed to compute $L$ are continuously updated by a feature tracker. In particular, the algorithm is initialized by detecting lines using an edge detector (Figure 2(b)) on the input image (Figure 2(a)). Since edges are almost vertical lines, we used a first-order Sobel derivative filter with a Scharr kernel along the $x$- and $y$-directions. The edge image is then evaluated using a binary threshold on the weighted sum of the two derivatives. Since lines are almost vertical within the image in the case at hand, the $y$-direction derivative has been weighted at 30% and the $x$-direction at 70%, in order to give more relevance to vertical edges. This way to detect edges is simple and computationally very efficient, even though it leads to some noise in the detection: a morphological erosion is therefore exploited for reducing noise.

The Hough line transform[22] has been employed in order to retrieve all the line parameters $\mathcal{L}_i(\rho_i, \theta_i)$, as shown in Figure 2(c). Due to the presence of noise in the input images, several lines could be detected. A K-means clustering[23] is run on the $\rho$ values of the detected lines to group together close lines related to the same edge of the cavity. Among the available clustering methods,[24] the K-means algorithm needs an a priori knowledge of the number of clusters, which adapts to our case. Indeed, we know the number of edges to be found in the image, and therefore, this clustering method provides a stable output. Clusters have been sorted according the average $\rho$ value and the two desired lines are selected in order to be tracked.

The tracker selected for following the cavity edges is a moving-edge line tracker,[25] which demonstrated good performance for the problem at hand; moreover, a fast and reliable implementation can be found in ViSP.

## GMM and GMR

The GMM is a parametric probability density function represented as a weighted sum of Gaussian component densities, which best fit the training data set. Naming $K$ the number of Gaussian components with which the problem must be approximated, the model can be characterized by the list of parameters

$$\boldsymbol{\theta} = \{\tau_1 \dots \tau_K, \boldsymbol{\mu}_1 \dots \boldsymbol{\mu}_K, \Sigma_1 \dots \Sigma_K\} \tag{6}$$

where:

- $\tau_i \in \mathbb{R}$ is the prior probability of the $i$ th Gaussian component, such that $\sum_{i=1}^{K} \tau_i = 1$.
- $\boldsymbol{\mu}_i \in \mathbb{R}^d$ is the mean vector of the $i$ th Gaussian component.
- $\Sigma_i \in \mathbb{R}^{d \times d}$ is the covariance matrix of the $i$ th Gaussian component.
- $d \in \mathbb{R}$ represents the dimensionality of the problem.

All the parameters ($\tau$, $\mu$ and $\sigma$) can be estimated from a training data set through an iterative expectation maximization (EM) algorithm.

Considering a single training data $\mathbf{x} \in \mathbb{R}^d$, its probability density function is assumed to be a weighted sum of normal probability density functions

$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{j=1}^{K} \tau_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \Sigma_j) \tag{7}$$

where

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \Sigma) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x}-\boldsymbol{\mu})}$$

The EM algorithm is applicable only if $K$ is known a priori. A method to estimate $K$ is to use the *Bayesian information criterion*,[26] but we used an empirical method, based on experimental observations. Indeed, few components cannot completely describe the system, while by selecting many components, the model becomes much complex introducing redundant information and then giving a null weight to some prior.

GMR estimates output data by specifying the desired input. Therefore, input and output data together represent a possible occurrence of training data set, where the input is selected by the user and the output is an estimation calculated using GMM.[27] In particular, a single data element can be rewritten as

$$\mathbf{x} = \begin{bmatrix} u \\ y \end{bmatrix} \tag{8}$$

Gaussian model parameters are partitioned in a similar way

$$\boldsymbol{\mu}_j = \begin{bmatrix} \mu_j^u \\ \mu_j^y \end{bmatrix}, \quad \Sigma_j = \begin{bmatrix} \Sigma_j^u & \Sigma_j^{uy} \\ \Sigma_j^{yu} & \Sigma_j^y \end{bmatrix}, \quad \sum_{j=1}^{K} \tau_j = 1 \tag{9}$$

Then, the regression function assumes the form

$$\mathbf{m}_y(\mathbf{u}) = E[\mathbf{y}|\mathbf{u}] = \sum_{j=1}^{K} \pi_j(\mathbf{u}) \mathbf{m}_j(\mathbf{u}) \tag{10}$$

with

$$\begin{aligned} \pi_j(\mathbf{u}) &= \frac{\tau_j \mathcal{N}(\mathbf{u}, \mu_j^u, \Sigma_j^u)}{\sum_{i=1}^{K} \tau_i \mathcal{N}(\mathbf{u}, \mu_i^u, \Sigma_i^u)} \\ \mathbf{m}_j(\mathbf{u}) &= \mu_j^y + \Sigma_j^{yu} \Sigma_j^{u\,-1} (\mathbf{x} - \mu_j^u) \end{aligned} \tag{11}$$

### Offline phase

This step aims to acquire a valid data set, suitable to train a GMM, as described in 'GMM and GMR' section. In particular, several scanning trials are performed with the visual servoing approach of 'Visual servoing' section, starting from different initial positions and with small external perturbations in order to make the system more robust. Visual servoing needs images with a high resolution in order to extract good features to be tracked, with a consequent frame rate limitation.

The dimension of the input data is a crucial parameter for any machine learning algorithm; in particular, high-dimensional input vectors stimulate the curse of dimensionality, leading to poor performance and need for huge training data sets. To reduce the number of inputs, one single image row (called patch) is provided to the GMM: this is feasible for our problem, as the input images are invariant along the $y$-axis.

During each trial, for each control loop, namely for each frame $\mathcal{I}(t)$, the camera velocity $\mathbf{v_c}(t)$ computed from the control law is saved in correspondence to the patch $\mathcal{P}(\mathcal{I}(t))$ extracted from the considered frame. In the examined setup, each saved patch corresponds to a subset of intensity values of a single line of pixels transversal to the bar length. In this way, it is possible to store all the information of the cavity position with respect to the camera (see Figure 3).

A data set composed of many couples $(\mathbf{v_c}(t), \mathcal{P}(t))$ has been used to train a GMM with $K = 4$ empirically chosen Gaussian components. For all the tests, we adopted a $(6 + 100) \times N$ dimension training data with six camera velocity components, 100 pixel intensity values sampled from a significant line of the raw image, while $N$ is the number of samples recorded from the simulations.

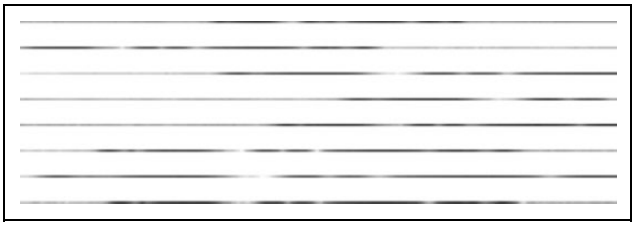

**Figure 3.** Each patch has a gray level distribution related to the bar pose with respect to the camera, that is, all the information that needed to compute the camera displacement can be extracted from the image as IBVS does. IBVS: image-based visual servoing.

### Online phase

In the online phase, the camera resolution has been physically reduced to an area which correspond to the patch extracted during the offline step. For each frame, the relative camera velocity has been estimated through GMR by exploiting the GMM trained during the offline phase. In this way, it is possible to emulate the visual servoing control law with a statistical model with the advantage of eliminating all the image processing on high-resolution images. This corresponded a reduction in the processing time, and the consequent possibility to increase the camera frame rate and the overall control rate.

## Experiments and results

### Experimental setup

We tested our system in a real environment reproducing an industrial setup (Figure 4) composed by a lightweight collaborative 6 degrees of freedom (DOFs) manipulator (*Universal Robots UR10*) equipped with a fully calibrated camera *PointGrey Grasshopper 3* with the following specifications:

- *Model*: GS3-U3-28S4C-C.
- *Sensor*: Sony ICX687 CCD, 1/1.8″, 3.69 $\mu$m.
- *Megapixel*: 2.8MP.
- *Interface*: USB3.
- *Resolution max*: 1928 $\times$ 1448.
- *Frame rate*: 26 fps at full res.
- *Optic*: Computar 8 mm 1:1.4 2/3.

The aluminium bar has been fixed horizontally to the robot support to represent the engine tooth to be scanned, a gap of 1.2 cm between the borders, as shown in Figure 2(a). All the developed code has been written in C++ within the ROS framework, exploiting the tools provided by the open source libraries *OpenCV*[28,29] and *ViSP*. Simulations have been launched and monitored in a personal computer (PC) with the following specifications:

- *Processor*: Intel Xeon CPU E3R25 at 3.10 GHz $\times$ 4.
- *OS*: Ubuntu 14.04 (x86-64).

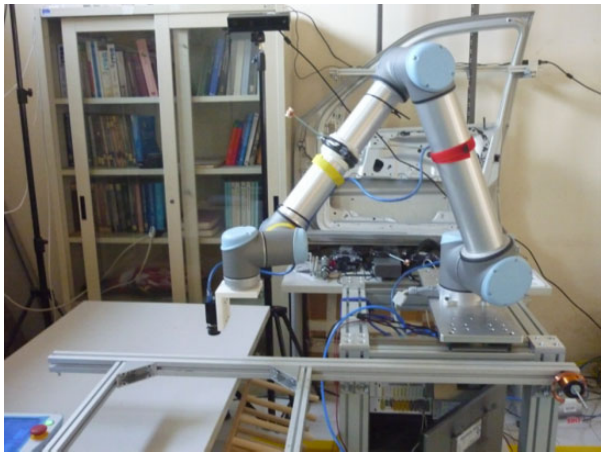

**Figure 4.** System used for simulations.

- *RAM*: 4GB.
- *Graphics*: NVIDIA GT218.

## Results

A training data set has been built by acquiring several scans of the bar cavity based on the IBVS framework presented in 'Visual servoing' section. The camera employed has a frame rate of 26 fps at full resolution, which could not be sufficient for applications requiring high responsiveness. For this reason, we reduced the resolution of the images in input to the IBVS system. A size of 500 × 400 at 55 fps has been selected as a good trade-off between frame rate and feature quality for detection and tracking. Performance decreased without a significant increase in the frame rate by further reducing the resolution.

The tune-up of the IBVS system consisted in the definition of the desired lines configuration $(\rho_i^*, \theta_i^*)\, i = \{1, 2\}$ and in the setting of the forward velocity parameters of (equation (5)). The desired view and motion of the camera led to the feature target values

$$\begin{cases} \rho_{1,2}^* = \pm 0.0045 \text{ m} \\ \theta_{1,2}^* = 0^\circ \end{cases} \tag{12}$$

that is, we assumed the lines as vertical; according to the camera parameters, these values correspond to a height of the camera from the bar of about 15 cm. The parameters regulating the camera motion have been selected with empirical observations after some trials, resulting as

$$\begin{cases} V_{\max} = -0.01 \text{ m/s} \\ \| \mathbf{e} \|_{\max}^2 = 0,0002 \\ \| \mathbf{e} \|_{\min}^2 = 0 \end{cases} \tag{13}$$

Several trials starting from different initial positions of the camera were recorded. The velocity has been computed using the IBVS control law, while the patches have been derived from a subsampling operation on the central pixel row, leading to an image of size 500 × 1.

Patch selection is a critical step of this approach: it must maximize the information contained while minimizing the size for both reducing the dimensionality of the problem and increasing the frame rate during the online phase. In particular, we discarded 100 pixels from both sides of the image, since they contained background; from the remaining 300 elements, only one over three pixels has been selected as training data, obtaining a 100 × 1 patch as shown in Figure 3. Our data set was composed of approximately $N \simeq 10,000$ samples, where about 4000 samples came from scanning started from position aligned with the bar or with small perturbations, while 3000 samples were extracted from left misalignments, and the last 3000 were obtained from right misalignments. As already claimed, in this work, we supposed a correct initial height of the camera, and therefore, only lateral displacement has been considered for the tests. Finally, we trained a GMM composed of $K = 4$ Gaussian components. The number of components has been selected empirically by comparing results from models created with the same input data, but with increasing $K$ values.

In the testing phase, the resolution of the camera has been reduced to the minimum size height allowed by the camera, that is, 500 × 2. A 100 × 1 patch has then been obtained by selecting the first row and subsampling it. Using these settings, it was possible to reach a frame rate of 94 fps, the maximum achievable by the camera, according to its datasheet. This represents an improvement of about 41.5% with respect to the framerate used during the offline phase (Figure 5).

The real visual servoing was compared to our learning-based method in order to verify the performance in terms of path followed, velocity set, precision and control rate improvements. In Figure 6, position and rotation of the camera have been monitored over the samples for the two considered systems (real visual servoing and GMM-based). The learning-based system was able to follow the gap in the aluminium bar, the recorded poses differed from the ones set by the traditional visual servoing system only for a small tolerance. This result is crucial, as it demonstrates that a learning engine is able to emulate the entire visual servoing behaviour when properly trained.

In order to better evaluate the performance of our system, commanded velocity has been monitored within the same experiment, and plotted in Figure 7. Oscillating behaviours in the $y$ component of the IBVS are projected into each component of the learning-based system, and are caused by the use of a distribution obtain through a sum of Gaussians. Indeed, the oscillations coming from the forward velocity control have been modelled as input data without a clear separation of the single velocity components, leading to a correlation between the parts and a consequent propagation of oscillations. Moreover, GMM performs an intrinsic smoothing of the data, resulting in

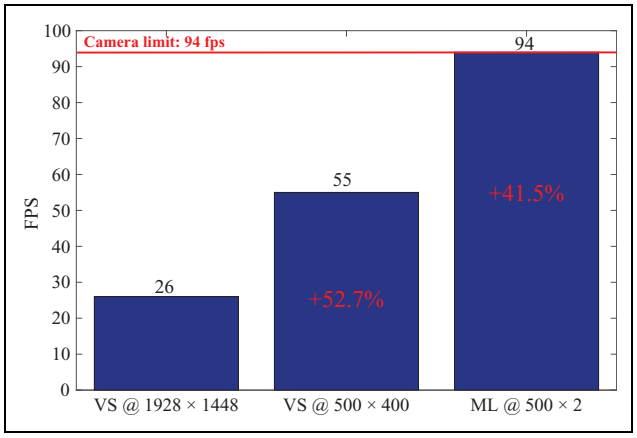


**Figure 5.** Image resolution of the camera has a dominant impact on the frame rate. Our method allows to perform a visual servoing task with a single row of pixels at a frame rate not achievable with standard image processing techniques.

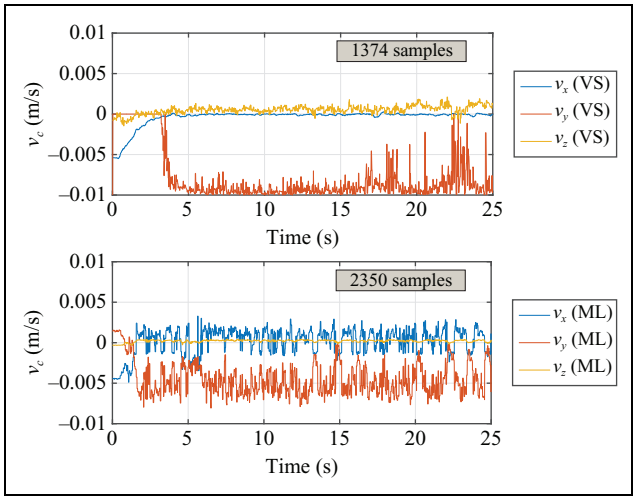


**Figure 7.** Velocity command output comparison. Only linear velocity is here reported as more informative data.

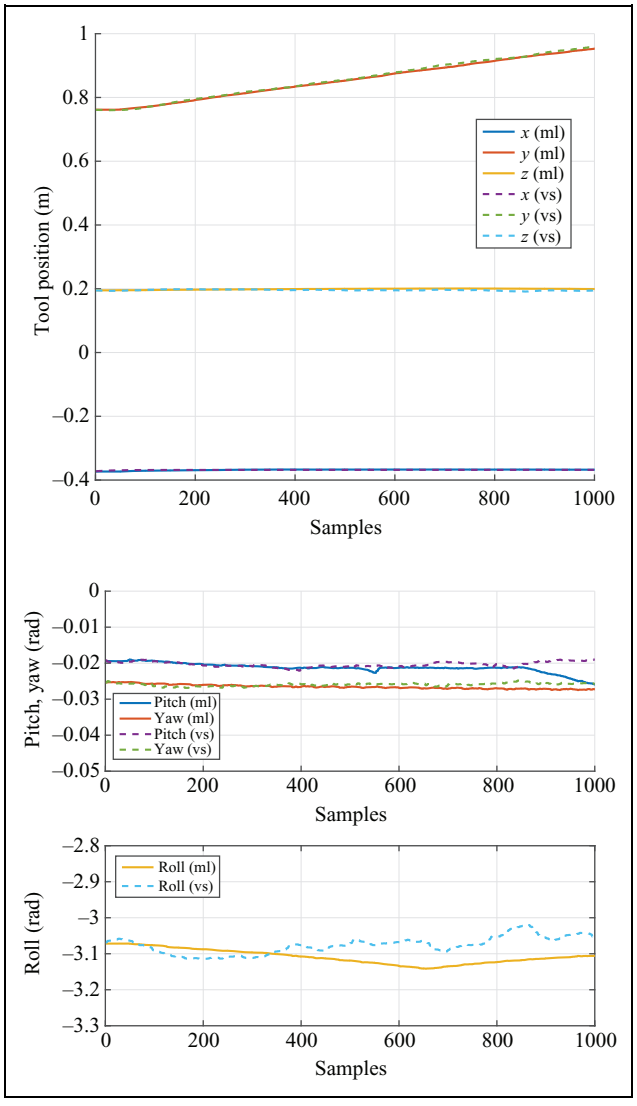


**Figure 6.** Camera path comparison, in terms of position and rotation.

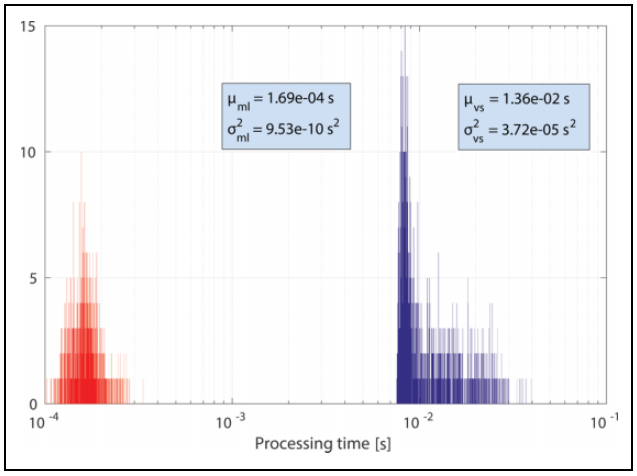


**Figure 8.** Time performances comparison, mean and variance over more than 1.000 acquisitions by fitting a Gaussian function. The *x*-axis has logarithmic scale.

low velocity values. In particular, for the forward velocity $v_y$, we reached an average value of about $-0.005$ m/s starting from a desired value of $-0.01$ m/s. It should be observed that during the 25 s (Figure 7) of simulation, the difference in the number of samples due to the increased frame rate allows the learning system to perform the velocity control more times with respect to the standard IBVS, with a consequent improvement in responsiveness.

The slight degradation of the control performance caused by velocity oscillations is counterbalanced by the higher frame rate. For IBVS, the average processing time is of about 13.6 ms, which allows a maximum throughput rate of about 73 fps without frame loss. Therefore, even if a powerful camera is used, the effective throughput of the control law would be limited by the constraint given by the processing time. Using the learning-based control, on the other hand, leads to the average processing time of about 0.17 ms, 100 times lower than the previous case (Figure 8).

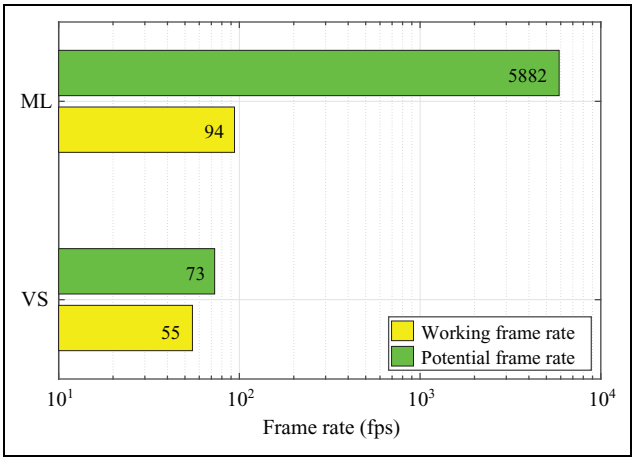


**Figure 9.** Comparison of the maximum achievable frame rate improvement. The *x*-axis has logarithmic scale.

The maximum allowed theoretical throughput would be of about 5882 fps with a dramatic performance improvement (Figure 9). The proposed solution offers a good trade-off between accuracy and responsiveness.

## Conclusions

In this article, we dealt with a real industrial setting in which the problem of automatically deploying copper wire around the narrow stator teeth of an electrical engine has been considered. In particular, we modelled the problem as a path following task to be performed with a collaborative robot provided by a monocular camera.

We presented an innovative approach to visual servoing, based on a machine learning technique, for boosting the control rate during the robot motion in order to avoid collisions. The starting point was an IBVS to perform a precise scanning of a cavity in a metal bar. Traditional visual servoing systems usually work with high-resolution images at low frame rate, which makes them unsuitable for highly responsive applications.

Our approach aims at avoiding all image processing on high-resolution images by replacing the control structure of a standard visual servo control scheme with a learning engine which emulates it given a very low-resolution informative image as input. A GMM has been trained with a custom data set extracted from several visual servoing scanning experiments. The data samples are patches of the camera view coupled with the velocity command computed by the traditional servo control law.

The GMM trained with such samples was then used to replace the IBVS. Velocities for robot control have been estimated based on a GMR algorithm analysing only a small patch as input, allowing a very high control rate. The learning-based framework has been able to emulate the standard solution within a limited configuration.

The proposed system shows a dramatic increase in control rate at the cost of a slightly lower accuracy. The effective increase of the frame rate is of about 41% for the tested setup, with a theoretical maximum control rate of 5882 fps.

Our approach exploited constraints and information knew in advance about the selected real case, as the image invariance along one axis. The effectiveness of the algorithm is surely influenced by such a priori knowledge, since we were able to use a single row image instead of larger portions. Indeed, the scalability to more general and complex scenarios needs further investigation. Nevertheless, we believe the proposed approach could be extended effectively.

In the near future, we plan to improve the system accuracy by testing advanced and smoother control laws while comparing the efficiency of different learning engines and regression techniques.

### Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

### Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was financially supported in part by the European Community Seventh Framework Programme under grant no. 608849 (European Robotics Challenge Project).

### References

1. Rüßmann M, Lorenz M, Gerbert P, et al. Industry 4.0: the future of productivity and growth in manufacturing industries. *Boston Consult Group* 2015; 14: 1–14.
2. Silveira G and Malis E. Direct visual servoing with respect to rigid objects. In: *2007 IEEE/RSJ international conference on intelligent robots and systems*, pp. 1963–1968. IEEE. https://www.scopus.com/inward/record.uri?eid=2-s2.0-51349099347&doi=10.1109%2fIROS.2007.4399487&partnerID=40&md5=79c7a43ced6bb68aa1cdd6663417cb72. DOI: 10.1109/IROS.2007.4399487.
3. Han S, Censi A, Straw AD, et al. A bio-plausible design for visual pose stabilization. In: *2010 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 5679–5686. IEEE. https://www.scopus.com/inward/record.uri?eid=2-s2.0-78651501074&doi=10.1109%2fIROS.2010.5652857&partnerID=40&md5=49142d04ba83ca744f66c3322fd59963. DOI: 10.1109/IROS.2010.5652857.
4. Collewet C and Marchand E. Photometric visual servoing. *IEEE Trans Robot* 2011; 27(4): 828–834.
5. Bakthavatchalam M, Chaumette F and Marchand E. Photometric moments: new promising candidates for visual servoing. In: *2013 IEEE international conference on robotics and automation (ICRA)*, pp. 5241–5246. IEEE. https://www.scopus.com/inward/record.uri?eid=2-s2.0-84887277678&doi=10.1109%2fICRA.2013.6631326&partnerID=40&md5=ea8d5aa260741c66e86854b789c1b7a5. DOI: 10.1109/ICRA.2013.6631326.

6. Bakthavatchalam M, Chaumette F and Tahri O. An improved modelling scheme for photometric moments with inclusion of spatial weights for visual servoing with partial appearance/disappearance. In: *2015 IEEE international conference on robotics and automation (ICRA)*, pp. 6037–6043. IEEE. https://www.scopus.com/inward/record.uri?eid=2-s2.0-84938268922&doi=10.1109%2fICRA.2015.7140046&partnerID=40&md5=41a6d533dd44e388d2b23628a038e6f4. DOI: 10.1109/ICRA.2015.7140046.
7. Crombez N, Caron G and Mouaddib EM. Photometric Gaussian mixtures based visual servoing. In: *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 5486–5491. IEEE. https://www.scopus.com/inward/record.uri?eid=2-s2.0-84958206435&doi=10.1109%2fIROS.2015.7354154&partnerID=40&md5=7dafe4e8ec101ef024b4b9f864007dff. DOI: 10.1109/IROS.2015.7354154.
8. Kallem V, Dewan M, Swensen JP, et al. Kernel-based visual servoing. In: *2007 IEEE/RSJ international conference on intelligent robots and systems*, pp. 1975–1980. IEEE. https://www.scopus.com/inward/record.uri?eid=2-s2.0-51349129639&doi=10.1109%2fIROS.2007.4399546&partnerID=40&md5=35dd414d1c9df8b49e009b5a96f356f3. DOI: 10.1109/IROS.2007.4399546.
9. Dame A and Marchand E. Mutual information-based visual servoing. *IEEE Trans Robot* 2011; 27(5): 958–969.
10. Hafez AHA, Achar S and Jawahar CV. Visual servoing based on Gaussian mixture models. In: *2008 ICRA 2008 IEEE international conference on robotics and automation*, pp. 3225–3230. IEEE. https://www.scopus.com/inward/record.uri?eid=2-s2.0-51649126259&doi=10.1109%2fROBOT.2008.4543702&partnerID=40&md5=d473d76858f4a03883c6d7b5b3f8363a. DOI: 10.1109/ROBOT.2008.4543702.
11. Miljković Z, Mitić M, Lazarević M, et al. Neural network reinforcement learning for visual control of robot manipulators. *Expert Syst Appl* 2013; 40(5): 1721–1736.
12. Lampe T and Riedmiller M. Acquiring visual servoing reaching and grasping skills using neural reinforcement learning. In: *The 2013 international joint conference on neural networks (IJCNN)*, pp. 1–8. IEEE. https://www.scopus.com/inward/record.uri?eid=2-s2.0-84893561637&doi=10.1109%2fIJCNN.2013.6707053&partnerID=40&md5=ce344620b2537462a18daa96d8e78ad6. DOI: 10.1109/IJCNN.2013.6707053.
13. Siradjuddin I, Behera L, McGinnity TM, et al. Image-based visual servoing of a 7-dof robot manipulator using an adaptive distributed fuzzy pd controller. *IEEE/ASME Trans Mechatronics* 2014; 19(2): 512–523.
14. Sadeghzadeh M, Calvert D and Abdullah HA. Self-learning visual servoing of robot manipulator using explanation-based fuzzy neural networks and q-learning. *J Intell Robotic Syst* 2015; 78(1): 83–104.
15. Levine S, Pastor P, Krizhevsky A, et al. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The Int J of Robotics Res* 2016. https://doi.org/10.1177/0278364917710318. DOI: 10.1177/0278364917710318.
16. Corke PI. Visual control of robot manipulators – a review. *Vis Ser* 1993; 7: 1–31.
17. Hutchinson S, Hager GD and Corke PI. A tutorial on visual servo control. *IEEE Trans Robot Autom* 1996; 12(5): 651–670.
18. Chaumette F and Hutchinson S. Visual servo control. i. basic approaches. *IEEE Robot Autom Mag* 2006; 13(4): 82–90.
19. Marchand E. Visp: a software environment for eye-in-hand visual servoing. In: *1999. Proceedings. 1999 IEEE international conference on robotics and automation*, Vol. 4, pp. 3224–3229. Piscataway, NJ, United States: IEEE. https://www.scopus.com/inward/record.uri?eid=2-s2.0-0032640449&partnerID=40&md5=2a2ece1c71bca6e337abc2b238c98cbb.
20. Marchand É and Chaumette F. Feature tracking for visual servoing purposes. *Robot Auton Syst* 2005; 52(1): 53–70.
21. Chaumette F and Hutchinson S. Visual servoing and visual tracking. In: Siciliano, Bruno, Khatib, Oussama (eds) *Springer handbook of robotics*, 2008, pp. 563–583. Verlag Berlin Heidelberg: Springer.
22. Ballard DH. Generalizing the Hough transform to detect arbitrary shapes ☆. *Pattern Recognit* 1981; 13(2): 111–122.
23. Jain AK. Data clustering: 50 years beyond k-means. *Pattern Recognit Lett* 2010; 31(8): 651–666.
24. Jain AK, Murty MN and Flynn PJ. Data clustering: a review. *ACM Comput Surv (CSUR)* 1999; 31(3): 264–323.
25. Bouthemy P. A maximum likelihood framework for determining moving edges. *IEEE Trans Pattern Anal Mach Intell* 1989; 11(5): 499–511.
26. Schwarz G. Estimating the dimension of a model. *Ann Stat* 1978; 6(2): 461–464.
27. Cohn DA, Ghahramani Z and Jordan MI. Active learning with statistical models. *J of Artifi Intell Res* 1996; 129–145. https://www.scopus.com/inward/record.uri?eid=2-s2.0-0029679131&partnerID=40&md5=77620557046f3a845dd4ab05fb1dba6d.
28. Itseez. Open source computer vision library. https://github.com/itseez/opencv (accessed 24 October 2017).
29. Bradski G. The OpenCV Library. *Dr Dobb's Journal of Software Tools* 2000; 25(11): 120–123.