# Robotic Learning for Increased Productivity: Autonomously Improving Speed of Robotic Visual Quality Inspection

Andrej Gams[1], Simon Reberšek[1], Bojan Nemec[1], Jure Škrabar[2], Rok Krhlikar[3], Jure Skvarč[2] and Aleš Ude[1]

***Abstract*— Robotic learning has shown many impressive achievements in laboratory environments, but it is not yet very prominent in the industry, where hardware, engineering solutions and careful structuring of the environment are typically needed to successfully accomplish the desired task. In this paper we show one example of learning applied to industrially relevant problems, where a learning algorithm is applied for the optimization of the velocity of robotic motion for quality inspection. Through learning of optimal velocity of motion, which is done only at the start of the production, we show that we can achieve faster cycle times and thus greater productivity. The described approach is general and can be used with different types of learning and feedback signals.**

## I. INTRODUCTION

In modern factories, and even more so in the factories of the future, many operations will be done with autonomous robots, which will use visual feedback [1]. Visual feedback can be used for moving around the working space and avoiding obstacles, to identify and locate parts, etc. Different vision techniques, such as structured light, time of flight and laser triangulation, line scan cameras, stereo vision, and even monocular vision with a standard area scan camera and an RGB camera are widely used for inspection and quality control processes in the industry [2], [3]. Different modes of quality inspection exist. For example, a robot can discretely check an object from a few viewpoints and compare the acquired images to predefined templates [4]. Another option is to continuously acquire images with an in-hand camera while the robot moves around the object to be inspected and look for anomalies. Likewise, the camera might be stationary and the object is moved around it. Many advanced methods for quality inspection have been proposed, including deep learning methods [5].

For effective visual quality control or any other vision-based operation, the machine vision hardware needs to be properly set-up and tuned. In large-scale automated production, it is typically set-up once, and then it remains in the same configuration throughout its life cycle. As a consequence, machine vision hardware is often designed in a way that some adjustments can only be carried out manually. Consequently, the vast majority of machine vision lenses have a fixed focal length and manual adjustment of the iris and focus [6]. However, even if the hardware is set up and optimized for production only once, this can be a tedious and demanding task for the robot programmer/operator. For example, continuous visual inspection, such as visual inspection of weld seams [7], requires the robot to follow the seam with the camera attached to its tip. The image has to be focused in all the positions and with all the velocities. Thus, for continuous visual quality control, the operator has to define not only the correct robot path, but also the correct speed. Figure 1 shows the effect of foo fast motion on the image sharpness, when the robot used an in-hand area scan camera. Too fast motion results in a blurry image.

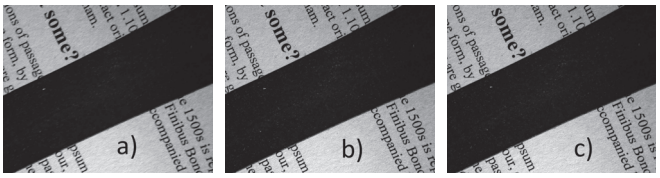


Fig. 1. Effect of motion speed of the sharpness of the image intended for quality inspection, with dummy text attached to a curved surface standing in for the object. a) Extremely fast motion; b) Extremely slow motion; c) Motion with autonomously learned velocity. Blurring is observable in image a).

The required cycle times usually set high demands on how fast the quality control has to be, and how it can be performed [8]. While the path can be properly configured by exporting the object CAD data and appropriate robot-to-object calibration, the speed of robot motion is typically left to the operator, who spends a considerable amount of time hand-tuning it. In this paper we show how such hand-tuning can be automated by employing learning algorithms.

### A. Problem Specification

In this paper we investigate autonomous learning of motion speed and apply it to visual quality inspection of products using a camera attached to the tip of the robot[1]. The system should

- follow a predefined path,
- allow easy optimization of motion velocity,
- employ learning algorithms to autonomously set the velocity of motion, such that
- the velocity of motion does not introduce blurring, i. e., a reduced focus measure in the visual feedback, and
- the system should be able to use different visual features.

The implementation is subject to the following assumptions. i) An accurate robot path (trajectory) with both re-

[1]Humanoid and Cognitive Robotics Lab, Department of Automatics, Biocybernetics and Robotics, Jožef Stefan Institute, Jamova cesta 39, 1000 Ljubljana, Slovenia, name.surname@ijs.si
[2]Kolektor Orodjarna d.o.o. - BU Vision, Vojkova 10, 5280 Idrija, SI.
[3]Kolektor Group d.o.o., Vojkova 10, 5280 Idrija, Slovenia.

[1]The system presented in this paper is a part of an industrial solution implemented for a company. Due to the demands of the company, the actual product, the process and the company name can't be disclosed.

978-1-7281-0356-3/19/$31.00 ©2019 IEEE

quired positions and orientations can be exported from a CAD system, and proper robot-object calibration can be achieved. This is the cornerstone of all off-line robot programming tool-kits, such as, for example, Robcad, RobotStudio, etc. ii) The system operates under constant lighting conditions.

The proposed system relies on the ability to easily modulate the robot's velocity, which is a feature of Dynamic movement primitives (DMPs) developed by Ijspeert et al. [9]. In this paper we used a variant of DMPs called Cartesian Space Dynamic Movement Primitives [10] for the trajectory encoding, which enables the specification of Cartesian space position and orientation trajectories. Other trajectory encoding approaches could easily be applied, for example dynamical systems [11] or Gaussian Mixture Models [12]. The proposed velocity modulation in this paper utilizes Iterative Learning Control (ILC) [13], [14]. Again, other methods, such as reinforcement learning [15], [16] could be applied, especially if also the robot trajectory would have to be adjusted.

The rest of this paper is organized as follows. In the next section, we provide details on the visual feedback quantities, namely the horizontal squared gradient focus measure for an area scan camera (II-A), and picture deformation from line scan image acquisition (II-B). Section III provides a summary on the used trajectory encoding – Cartesian Space Dynamic Movement Primitives with added temporal scaling. Section IV provides the details of the applied learning algorithm, i.e, Iterative Learning Control. Section V provides the results of the proposed algorithm. A short discussion and conclusion follow in Section VI and VII, respectively.

## II. VISUAL QUALITY MEASURE

In our experimental setup we used two different focus measures and two different cameras, as described in Section V.

### A. Focus Measure Using an Area Scan Camera

As stated in [6], there are only a few industrial camera/lenses on the market available that provide autofocus. Furthermore, there is little information about how focus is determined in these cameras. Visual quality inspection requires sharp, focused images. We used robot-driven autofocus as described in [6] to set our fixed-focus camera at the right distance from the object for inspection.

Many different focus measures exist, and are based on different orders of differentiation (first or second), image histogram, correlation and data compression [17]. Methods employing first-order gradients use different operators, such as squared gradient, Sobel (horizontal, vertical, combined), Laplacian, Scharr, and others. As stated in [17], the versions based on just the horizontal gradient alone work better than all other alternatives. It was shown in [6] that horizontal squared gradient performs well.

We therefore used horizontal squared gradient focus measure to evaluate the sharpness of the image. It is defined as [17], [18]

$$\phi = \frac{1}{A(B-1)} \sum_{x=m}^{A+m-1} \sum_{y=n}^{B+n-2} \left(I(x,y+1) - I(x,y)\right)^2 . \tag{1}$$

Here the region of interest is sized $A \times B$, starting at pixels $m$ and $n$, with $I(x,y)$ are the intensity values at pixels $(x,y)$.

The squared gradient focus measure has a distinct bell-shape characteristics, with the best focus achieved at the peak. We can exploit this to achieve autofocus with the robot. The robot moves the camera perpendicularly to the object of inspection, away and towards the object. After detecting the peak value (the focus measure begins to decrease), the robot reverses its motion and travels in the other direction at a slower speed, again until crossing the peak value. These movements are repeated until the accurate position resulting in peak focus measure $\phi$ is obtained. Details of this method and results showing that the achieved focus measure is higher than the one achieved by manually positioning the camera, are presented in [6].

Using this approach we can set the camera into focus for one point, for example above the starting point of the path of inspection. We assume that the desired inspection path has been extracted from a CAD model of the inspected object. To obtain the reference values $\phi(t)$ for all points on the inspection trajectory, the robot first moves along the desired inspection path at a slow speed.

It should be noted that in order to compare $\phi(t)$ for motions at different speeds, we need to anchor them to the phase of motion $s$. This phase $s$ is the phase of the trajectory encoding, i.e. Cartesian space dynamic movement primitives. Since the values of $\phi$ at different phases are repeatable apart from noise, a learning algorithm can be applied to compute the optimal speed of motion with respect to the reference values $\phi(t)$. The evolution of focus measure at different speeds is presented in Fig. 2. In the bottom plot we can see a clear difference in the focus measure for different speeds of motion.

### B. Line Scan Image Aquisition

A line scan camera acquires lines of pixels at a very high frequency, typically in the range of several kHz. The camera is moved over the object (or the object under the camera) at a speed that is synchronized with the acquisition rate. An incorrect speed results in a deformed picture, either the features are too wide if the motion is too fast, or they are too narrow, if the motion is too slow. Figure 3 shows the effect of motion speed on the deformation of the acquired picture of a structured pattern, called the Gray coded pattern.

By using such a structured pattern, we can accurately evaluate whether the speed of motion is too fast or too slow. By measuring the number of pixel lines between color transitions, and at a known sampling frequency, width of the line and number of pixels in the line, we can calculate the speed of motion that results in square pixels. Such calibration of speed is required for any utilization of line scan cameras. However, typically objects of inspection do not have Gray

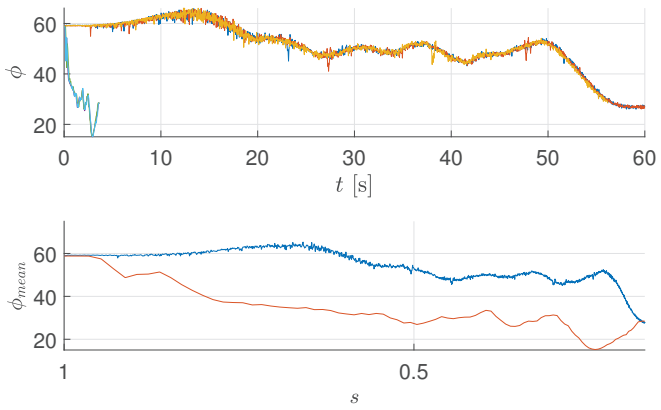


Fig. 2. Focus measure is repeatable, as demonstrated in both plots. Top plot shows $\phi$ as a function of time, top three lines for 60s motion, lower and shorter 3 lines for 3s motion. Bottom plot shows the average focus measure values $\phi$ as a function of phase $s$, red for 3s motion and blue for 60s motion.

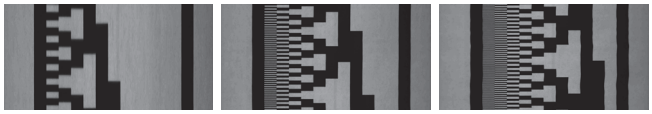

Fig. 3. Line scan image acquisition of the binary coded Graycode pattern at three different speeds of motion - left to right - slow to fast. Features of the pattern get elongated with higher speeds.

coded patterns printed on them, so some other features in the picture can be accurately measured and then estimated for correct aspect ratios at different speeds. This process is typically performed manually, is very tedious, and takes a long time. In this paper we used the aforementioned Gray coded pattern for a proof of concept. The output of speed measure is the number of pixels between color transitions, but with calibration it can be transformed into[2] mm/s as shown in Fig. 4.

## III. TRAJECTORY ENCODING

For completeness of the paper a brief recount of Cartesian space dynamic movement primitives (CDMPs) is provided in the following. In this paper we expanded on the original formulation from [10] with temporal scaling, as originally proposed for standard DMPs in [19].

Cartesian space Dynamic Movement Primitives are of composed position and orientation orientation parts. The position part of the trajectory is the same as in standard DMPs [9]. The orientation part of the CDMP, however, is represented by unit quaternions. Unit quaternions require special treatment, for both the nonlinear dynamic equations and the integration of these equations.

The following parameters compose a CDMP: weights $\boldsymbol{w}_k^p,\ \boldsymbol{w}_k^o \in \mathbb{R}^3,\ k = 1, \ldots, N$, which represent the position and orientation parts of the trajectory, respectively; trajectory duration $\tau$ and the final desired, goal position $\boldsymbol{g}^p$ and

[2]Given a direct required velocity, we could have simply set the velocity of the motion on the predefined path. This is only possible in calibrated settings and with the use of structured patterns. Here we use it only to show the proof of concept.

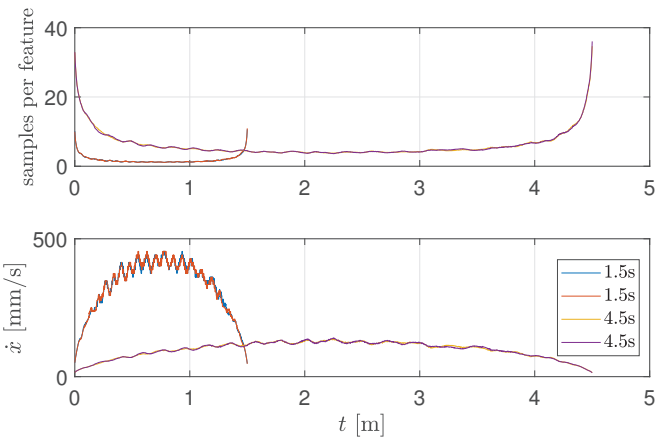


Fig. 4. Output of speed measure as samples per feature in the top plot, transformed into mm/s in the bottom plot. This is only possible when using a coded pattern. A difference with different input speeds is observable. The input velocity profile for the motion was a minimum-jerk trajectory.

orientation $\boldsymbol{g}^o$ of the robot. Variable $N$ sets the number of radial basis functions that are used to encode the trajectory. The orientation is in CDMP represented by a unit quaternion $\boldsymbol{q} = v + \boldsymbol{u} \in \mathrm{S}^3$, where $\mathrm{S}^3$ is a unit sphere in $\mathbb{R}^4$. $v \in \mathbb{R}$ is its scalar and $\boldsymbol{u} \in \mathbb{R}^3$ its vector part. To encode position ($\boldsymbol{p}$) and orientation ($\boldsymbol{q}$) trajectories we use the following differential equations:

$$\nu(s)\tau\dot{\mathbf{z}} = \alpha_z(\beta_z(\mathbf{g}^p - \mathbf{p}) - \mathbf{z}) + \mathbf{f}_p(s), \quad (2)$$

$$\nu(s)\tau\dot{\mathbf{p}} = \mathbf{z}, \quad (3)$$

$$\nu(s)\tau\dot{\boldsymbol{\eta}} = \alpha_z\left(\beta_z 2\log\left(\boldsymbol{g}^o * \bar{\boldsymbol{q}}\right) - \boldsymbol{\eta}\right) + \mathbf{f}_o(s), \quad (4)$$

$$\nu(s)\tau\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{\eta} * \boldsymbol{q}, \quad (5)$$

$$\nu(s)\tau\dot{s} = -\alpha s. \quad (6)$$

Variable $\nu(s)$, as a function of the phase, provides temporal scaling. Parameters $\boldsymbol{z}$, $\boldsymbol{\eta}$ denote the scaled linear and angular velocity ($\boldsymbol{z} = \tau\dot{\boldsymbol{p}}$, $\boldsymbol{\eta} = \tau\boldsymbol{\omega}$). For details on quaternion product $*$, conjugation $\bar{\boldsymbol{q}}$, and the quaternion logarithm $\log(\boldsymbol{q})$, see [10]. The nonlinear parts, termed also forcing terms, $\mathbf{f}_p$ and $\mathbf{f}_o$ are defined as

$$\mathbf{f}_p(s) = \boldsymbol{D}_p \frac{\sum_{k=1}^N \mathbf{w}_k^p \Psi_k(s)}{\sum_{k=1}^N \Psi_k(s)} s, \quad (7)$$

$$\mathbf{f}_o(s) = \boldsymbol{D}_o \frac{\sum_{k=1}^N \boldsymbol{w}_k^o \Psi_k(s)}{\sum_{k=1}^N \Psi_k(x)} s. \quad (8)$$

Forcing terms contain parameters $\boldsymbol{w}_k^p,\ \boldsymbol{w}_k^o \in \mathbb{R}^3$. They have to be learned, for example directly from an input Cartesian trajectory $\{\boldsymbol{p}_j, \boldsymbol{q}_j, \dot{\boldsymbol{p}}_j, \boldsymbol{\omega}_j, \ddot{\boldsymbol{p}}_j, \dot{\boldsymbol{\omega}}_j, t_j\}_{j=1}^T$. The scaling matrices $\boldsymbol{D}_p,\ \boldsymbol{D}_o \in \mathbb{R}^{3\times3}$ can be set to $\boldsymbol{D}_p = \boldsymbol{D}_o = \boldsymbol{I}$. Other possibilities are described in [10]. The nonlinear forcing terms are defined as a linear combination of radial basis functions $\Psi_k$

$$\Psi_k(x) = \exp\left(-h_k\left(x - c_k\right)^2\right). \quad (9)$$

Here $c_k$ are the centers and $h_k$ the widths of the radial basis functions. The distribution of weights can be, as in [20],

$c_k = \exp\left(-\alpha_x \frac{k-1}{N-1}\right)$, $h_k = \frac{1}{(c_{k+1}-c_k)^2}$, $h_N = h_{N-1}$, $k = 1, \ldots, N$. The time constant $\tau$ is set to the desired duration of the trajectory, i. e. $\tau = t_T - t_1$. The goal position and orientation are usually set to the final position and orientation on the desired trajectory, i. e. $\boldsymbol{g}^p = \boldsymbol{p}_{t_T}$ and $\boldsymbol{g}^o = \boldsymbol{q}_{t_T}$. Detailed CDMP description and auxiliary math are explained in [10].

Temporal scaling $\nu(s)$ provides a trajectory that defines a speed profile of the motion. It is composed of a weighted combination of kernel functions

$$\nu(s) = \frac{\sum_{k=1}^{R} \mathbf{w}_k^\nu \Psi_k(s)}{\sum_{k=1}^{R} \Psi_k(s)}. \tag{10}$$

Here $R$ defines the number of kernel functions, given in (9), for temporal scaling. For simplicity, this number can be the same as $N$ in (7). The weights $\mathbf{w}_k^\nu$ need to be learned in the same manner as the weights for position and orientation trajectories.

## IV. LEARNING

As shown in Fig. 2, focus measure is i) repeatable, and ii) there is a clear difference in $\phi(s)$ for different motion speeds. Therefore, we can use $\phi(s)$ as the feedback for learning. The same is valid for learning of the speed of motion using a line scan camera, as shown in Fig. 3.

The goal of learning in this paper is i) to learn a fastest possible velocity profile, where there will be only little or even no degradation of the focus measure. Thus, the motion will be executed as fast as possible, and the sharpness of the image, used for quality inspection, will not degrade. And ii) the correct speed of motion for using a line scan camera.

It should be noted that with the chosen parametric speed profile representation, different means of learning open up, as was shown in [16], or in [21]. In this paper we have chosen one of the variations of iterative learning control, which learns directly at the weights of the velocity profile $\nu(s)$. The advantage of using a learning control method is that it requires very few iterations to achieve substantial improvement. On the other hand, such methods never truly converge, but only asymptotically approach the target value [13]. Since the overall goal of the paper is not to present a new learning algorithm, but demonstrate how it can be used in the industry, and how to apply it to different measurable quantities, such an algorithm suffices. A combination of ILC and reinforcement learning (RL), as was shown in [22], could be applied, or even simply RL, which would take many more iterations. Because learning for an industrial setting is to be only performed to tune the motion once, before the start of production, a large number of repetition for RL should not pose a problem.

The chosen learning algorithm for learning was previously applied for coaching of robot motion through human intervention [23]. A short recap is provided for completeness of the paper. Its basis is learning of weights of CDMPs, but in this case it is used for the learning of the weights of the velocity profile $\nu$. The weights of the velocity profile $\boldsymbol{w}^\nu$ are iteratively updated (for 1DOF) with

**procedure** LearnProfile
  record $\phi_s$ for slow (practically static) motion;
  record $\phi$ for fast motion with $w_i^\nu = \text{const}$;
  **while** $\phi_{\text{latest}} >$ threshold
    execute motion with current $w^\nu$
    calculate new error of $\phi$ with $\phi_s - \phi_{\text{latest}}$
    **update** $\mathbf{w}^\nu$ using (11), (12) and (14)
  **end**

Fig. 5. Procedure for learning the velocity profile using the squared gradient focus measure. The procedure for velocity learning for using a line scan camera is the same, but with different feedback quantity.

$$w_{i,j+1}^\nu = w_{i,j}^\nu + \Gamma_{i,j+1} P_{i,j+1} r e_j \tag{11}$$

$$P_{i,j+1} = \frac{1}{\lambda}\left(P_{i,j} - \frac{P_{i,j}^2 r^2}{\frac{\lambda}{\Gamma_i} + P_{i,j} r^2}\right) \tag{12}$$

$$e_j = f_{\text{targ},j} - w_{i,j}^\nu r. \tag{13}$$

Here $j+1$ stands for the next time sample and $i$ for the selected weight. $P_i$, is the inverse covariance of $w_i$, $r$ is the amplitude gain. To apply this algorithm for modifying the speed profile based on the focus measure $\phi$, we replace (13) with

$$e_j = k * (\phi_{\text{slow motion}} - \phi_{\text{fast motion}}). \tag{14}$$

here $k$ is a positive constant gain. The whole algorithm is described in procedure of Fig. 5. The learning takes place until a predefined threshold of $e_j$ is reached. This threshold can be determined empirically. For learning of the correct velocity using a line scan camera, we use

$$e_j = k * (\text{px}_{\text{des}} - \text{px}), \tag{15}$$

where $\text{px}_{\text{des}}$ stands for desired pixels per feature and px for the measured.

Instead of learning directly on the weights, one can also simply generate the velocity profile from the weights and add to it a scaled $e_j$,

$$\nu_{l+1}(t) = \nu_l(t) + k e_j(t), \tag{16}$$

where the gain $k$ is set empirically and $l$ stands for iteration. The resulting $\nu_{l+1}(t)$ is then again encoded into weights, for example iteratively using (11) – (13), or with a batch conversion, as shown in [9].

## V. EXPERIMENTAL EVALUATION

### A. Experimental Setup

Our experimental setup consisted of the UR-10 robot with two different cameras mounted at the end-effector. The Robot was controlled using a modified version of the ur modern driver [24] from ROS in soft real-time at 125Hz, which is the maximum control frequency for this robot. The desired position of the robot, obtained from integrating a temporally-scaled CDMP, was updated in every control cycle.

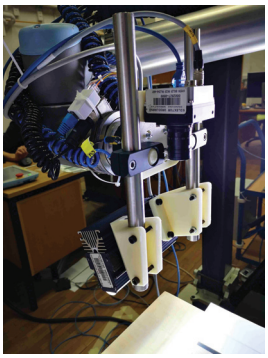

Fig. 6. Left: mock-up cell for quality inspection, composed of the UR-10 robot, the Basler acA1300-60gm area scan camera, a dedicated light source and the object at a calibrated distance from the robot. Right: Basler Racer (raL2048-48gm) line scan camera and lighting system for line scan image acquisition, attached at the end of the robot.

Fig. 7. The dummy object of inspection with a line standing in for an edge, and random text around for a random background.

For the area scan camera, we used an industrial grade GigE camera - Basler acA1300-60gm with resolution of 1282x1226 and a maximum frame-rate of 60 fps at full frame. 30Hz frame-rate was used in the experiments. A dedicated led light was used for constant lighting conditions. The set-up is depicted in Fig. 6, left. The line scan camera was the Basler Racer (raL2048-48gm). An additional lighting system to account for the high lighting condition demands of such cameras was added. The setup is depicted in Fig. 6, right. Since there is no possibility to synchronize the rate of camera line acquisition with the actual movement of the robot at higher frequencies (the robot is controlled maximally at 125Hz), a constant line frequency of 2kHz was used.

### B. Object for Inspection

Using an area scan camera we tested the algorithm for inspection of a flat surface, and for an object with a curved surface. Figure 7 shows the curved object of inspection. A line on a curved surface with both convex and concave curves represents a generic object, while the random text around it represents a random background[3]. Line scan camera was used only with a flat surface.

### C. Area Scan Camera Results

An area scan camera and horizontal squared gradient focus measure was first used on a flat object. While the velocity for a flat object can be quite intuitively set manually, we used it to demonstrate the concept. After recording the focus measure $\phi$ across the complete trajectory of motion at a very slow speed, another one was recorded at a fast speed and then the proposed learning algorithm was used to modify the speed.

[3]The actual object of inspection in the industrial setting cannot be shown.

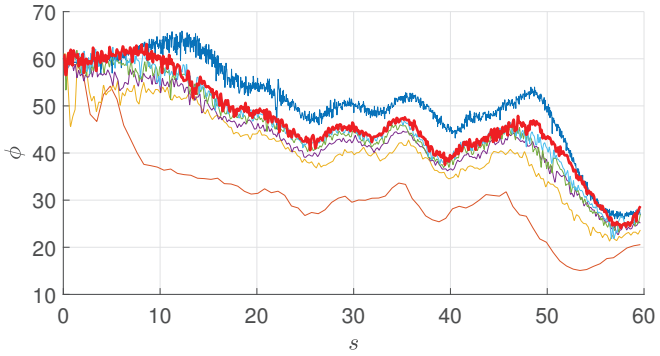


Fig. 8. Results of learning to achieve optimal velocity profile for a flat object. The top lines shows $\phi$ for slow motion, lasting 60s. The bottom line shows $\phi$ for fast motion, lasting 3s. $\phi$ over motion in 5 iterations is shown in the lines between, with the final, red line reaching practically the reference, but at 19.12s.

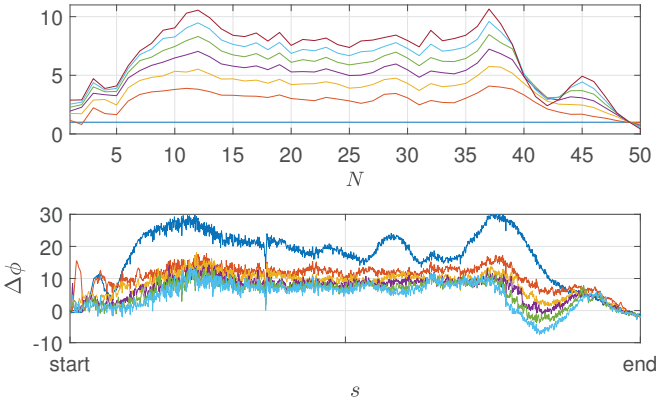


Fig. 9. Top: Evolution of weights $\mathbf{w}^{\nu}$ over iterations for a flat object, from all being set to 1, to final values, depicted in the top line. Bottom: difference of $\phi$ with respect to $\phi_{60}$ recorded for the motion of 60s.

It should be noted that the trajectory of motion was obtained from a CAD depiction of an object (even in the case of the flat surface). A CAD depiction of an object returns only the points in space, but not a time parametrization of the motion. While for a flat surface we could naively set all the points at equal distances and assign equal time intervals between them, such an approach cannot be applied for more complex objects and trajectories. In this concrete example, we set the initial velocity profile of motion to be a minimum-jerk trajectory with zero initial and final speeds of motion. Thus, the motion was extremely smooth, but made the optimization of the velocity profile more interesting. It should also be noted that having initial and final velocities set to zero, increasing the velocity of motion through changing $\nu$ in (2) – (6) will have no effect at the start and end of the motion. Results in Fig. 8 show the adaptation from the fastest motion at 3s (bottom line), to the end of learning at 19.12s (red line), which approaches a very slow motion at 60s (top line). Fig. 9 shows in the top plot the evolution of

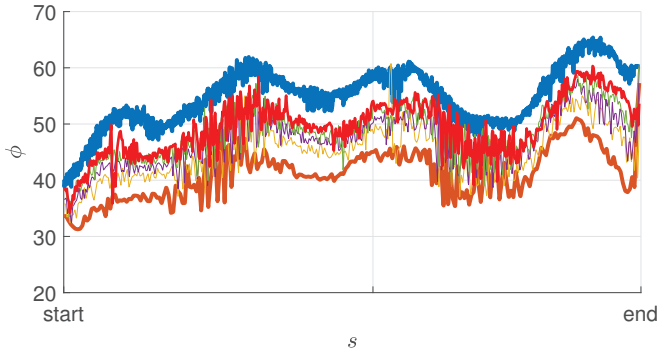


Fig. 10. Results of learning to achieve optimal velocity profile for the curved object. The top, blue line shows $\phi$ for slow motion, lasting 51.8s. The bottom, brown line shows $\phi$ for fast motion, lasting 7.77s. $\phi$ over motion in 4 iterations is shown in the lines between, with the final, red line reaching practically the reference, but at 19.9s.

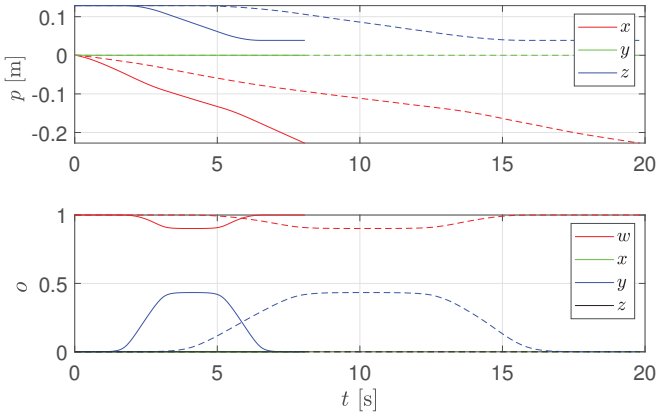


Fig. 11. Inspection trajectory on the surface of the object. Positions in the top plot, orientations as a unit quaternion in the lower plot. In both plots the solid lines show the initial, too-fast trajectories, and the dashed lines the trajectories after learning.

the weights $\boldsymbol{w}^{\nu}$, and in the bottom plot the error of $\phi$ used for learning. Minimum-jerk velocity profile was used for the initial trajectory.

As expected, the error was decreasing, but it never reaches 0, it only asymptotically approaches it. Here one could play with adaptive gains, but this might make the learning algorithm unstable [13]. Nevertheless, as can be seen in Fig. 1 c), the resulting image is practically as focused as is the one for the slow motion in Fig. 1 a), despite the motion of the robot being much faster. Even with noisy $\phi$ measure, using CDMPs and a weighted RBFs for the velocity profile results in smooth trajectories.

For the curved object we set our initial velocity profile to a constant velocity. Focus measure $\phi$ for the curved object, depicted in Fig. 10, shows the same trend of approaching $\phi$ of the very slow motion. The algorithm effectively achieved a reduction of inspection time (from 51.8s to 19.9s), at an only slightly decreased focus measure. The trajectory of the inspection point on the object, given as desired positions and orientations to the robot, is depicted in Fig. 11. A sequence of still images depicting the motion of the robot along the object is shown in Fig. 14.

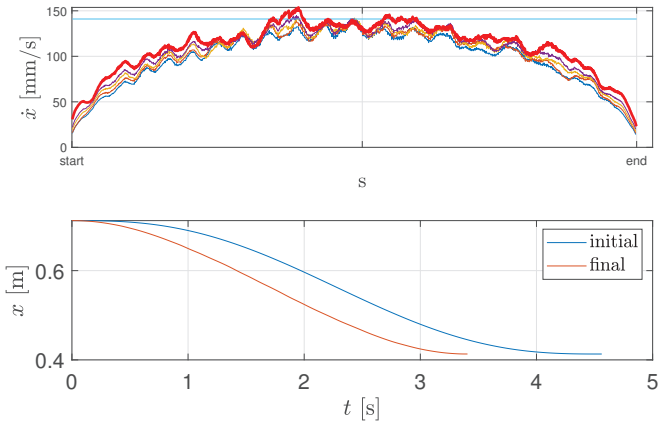


Fig. 12. Top: measured velocity of motion using the Gray coded pattern, the top, red line depicts the final values after 4 iterations. The straight line shows the optimal speed for the given camera settings. Bottom: position of the camera in the direction of motion over time.

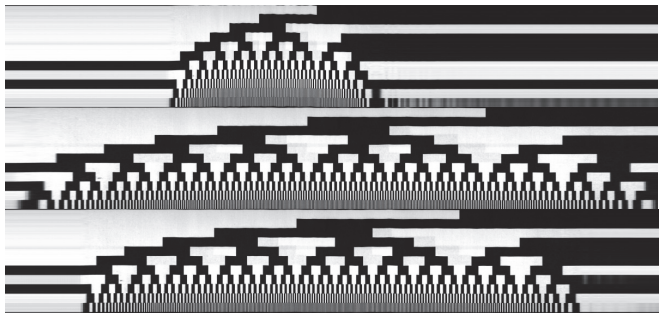

Fig. 13. Gray coded pattern recording under three velocities: top in 1.5s, middle in 4.5s, bottom after optimization in 3.40s.

### D. Line Scan Camera Results

We used line scan camera on a flat object with a Gray coded pattern. Our initial velocity profile for the line scan camera motion was again the minimum-jerk, with zero initial and final velocities. Results in Fig. 12 show that the shape of the profile was changing to account for the slow start and end of the motion. However, a small gain was used and the effect is rather small. Nevertheless, we can see the same effect of asymptotically approaching the desired value, albeit at a rather slow pace. In the bottom plot we can see that the initial minimum-jerk profile is slowly becoming more straight. Figure 13 shows the recorded Gray coded pattern under too slow, too fast and final learned velocity.

## VI. DISCUSSION

The result show improved behavior for all three cases: quality inspection speed of a flat object and of a curved object using the area scan camera, and quality inspection speed of a flat object using the line scan camera.

When using the focus measure, the question is when should the learning/optimization stop. We can see in Fig. 1 that the right-two pictures are practically identically sharp, but the focus measure is not the same. Focus measure recorded for 60s, as shown in Fig. 8, would be exactly the same (minus noise) for 90s motion , but just slightly lower for 45s (not depicted). However, the difference in image

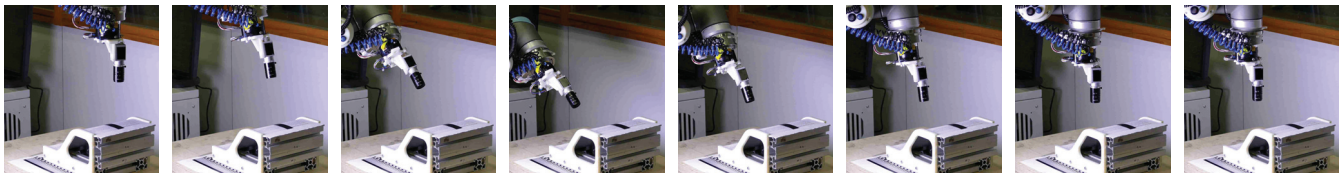

Fig. 14. Sequence of still images showing the robot tracing the edge (line) on the curved surface, which represents a generic object.

sharpness is not observable. Learning was stopped after 5 iterations, the resulting motion was 3 times faster than the referential motion. Determining when the learning should stop remains an open research question.

The proposed learning algorithm adapts the behavior of the robot based on measured data. We could combine this with calculated, theoretical data, and optimize the motion in advance, for example using some constraint optimization methods, such as [25]. Applying first theoretical optimization, followed by self-adaptation to account for the real-world situation and noise, is a more complex approach, but one that can potentially bring further improvements to the process.

When using the line scan camera, we need to compare image features to achieve square pixel. Having a coded pattern, as used here for proof of concept, directly outputs the desired velocity of motion. In real applications, however, the tuning of the inspection speed is very tedious, as sampling frequencies of up to 20kHz are not unusual.

## VII. CONCLUSION

The use of learning algorithms has great potential to add to the productivity of factories not only in the future, but already today. As the results show, self-adaptation algorithms can improve the performance of the robot, and this can be effectively applied in setting up and optimizing production processes, which is now mostly entirely left to the operators/engineers. Fine-tuning and calibration of the processes is a tedious, long process, requiring a lot of effort. Time and money can be saved both in the set-up as well as in the improved productivity.

## REFERENCES


[1] L. Pérez, Í. Rodríguez, N. Rodríguez, R. Usamentiaga, and D. F. García, "Robot guidance using machine vision techniques in industrial environments: A comparative review," *Sensors*, vol. 16 3, 2016.

[2] N. Herakovic, "Robot vision in industrial assembly and quality control processes," in *Robot Vision*, ch. 26, Rijeka: IntechOpen, 2010.

[3] C.-S. Cho, B.-M. Chung, and M.-J. Park, "Development of real-time vision-based fabric inspection system," *IEEE Transactions on Industrial Electronics*, vol. 52, pp. 1073–1079, Aug 2005.

[4] T. Ivanovska, S. Reich, R. Bevec, Z. Gosar, M. Tamosiunaite, A. Ude, and F. Wörgötter, "Visual inspection and error detection in a reconfigurable robot workcell: An automotive light assembly example," in *VISIGRAPP*, 2018.

[5] D. Racki, D. Tomazevic, and D. Skocaj, "A compact convolutional neural network for textured surface anomaly detection," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1331–1339, March 2018.

[6] R. Bevec, T. Gašpar, and A. Ude, "Robot-driven autofocus control mechanism for an in-hand fixed focus camera," in *Advances in Service and Industrial Robotics* (N. A. Aspragathos, P. N. Koustoumpardis, and V. C. Moulianitis, eds.), (Cham), pp. 551–559, Springer International Publishing, 2019.

[7] D. Schreiber, L. Cambrini, J. Biber, and B. Sardy, "Online visual quality inspection for weld seams," *The International Journal of Advanced Manufacturing Technology*, vol. 42, pp. 497–504, May 2009.

[8] O. Semeniuta, S. Dransfeld, and P. Falkman, "Vision-based robotic system for picking and inspection of small automotive components," in *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, pp. 549–554, Aug 2016.

[9] A. Ijspeert, J. Nakanishi, P. Pastor, H. Hoffmann, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Computation*, vol. 25, no. 2, pp. 328–373, 2013.

[10] A. Ude, B. Nemec, T. Petrič, and J. Morimoto, "Orientation in cartesian space dynamic movement primitives," in *IEEE Int. Conference on Robotics and Automation (ICRA)*, pp. 2997–3004, 2014.

[11] S. S. M. Salehian, N. Figueroa, and A. Billard, "A unified framework for coordinated multi-arm motion planning," *The International Journal of Robotics Research*, vol. 37, no. 10, pp. 1205–1232, 2018.

[12] S. Calinon, "Robot learning with task-parameterized generative models," in *Proc. Intl Symp. on Robotics Research (ISRR)*, 2015.

[13] D. A. Bristow, M. Tharayil, and A. G. Alleyne, "A survey of iterative learning control," *IEEE Ctrl. Sys. M.*, vol. 26, no. 3, pp. 96–114, 2006.

[14] A. Gams, B. Nemec, A. J. Ijspeert, and A. Ude, "Coupling movement primitives: Interaction with the environment and bimanual tasks," *IEEE Transactions on Robotics*, vol. 30, pp. 816–830, Aug 2014.

[15] J. Kober and J. Peters, "Policy search for motor primitives in robotics," *Machine Learning (MLJ)*, no. 1-2, pp. 171–203, 2011.

[16] M. P. Deisenroth, G. Neumann, and J. Peters, "A survey on policy search for robotics," *Foundations and Trends in Robotics*, vol. 2, no. 1-2, pp. 1–142, 2013.

[17] H. Mir, P. Xu, and P. van Beek, "An extensive empirical evaluation of focus measures for digital photography," in *Digital Photography X*, vol. 9023 of *procspie*, p. 90230I, 2014.

[18] S. Yousefi, M. Rahman, N. Kehtarnavaz, and M. Gamadia, "A new auto-focus sharpness function for digital and smart-phone cameras," in *2011 IEEE International Conference on Consumer Electronics (ICCE)*, pp. 475–476, Jan 2011.

[19] B. Nemec, A. Gams, and A. Ude, "Velocity adaptation for self-improvement of skills learned from user demonstrations," in *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 423–428, Oct 2013.

[20] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-specific generalization of discrete and periodic dynamic movement primitives," *IEEE Transactions on Robotics*, vol. 26, pp. 800–815, Oct 2010.

[21] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[22] B. Nemec, M. Simoni, N. Likar, and A. Ude, "Enhancing the performance of adaptive iterative learning control with reinforcement learning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2192–2199, Sep. 2017.

[23] A. Gams, T. Petrič, M. Do, B. Nemec, J. Morimoto, T. Asfour, and A. Ude, "Adaptation and coaching of periodic motion primitives through physical and visual interaction," *Robotics and Autonomous Systems*, vol. 75, pp. 340 – 351, 2016.

[24] T. T. Andersen, "Optimizing the universal robots ros driver.," tech. rep., Technical University of Denmark, Department of Electrical Engineering, 2015.

[25] L. Žlajpah, "On time optimal path control of manipulators with bounded joint velocities and torques," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1572–1577 vol.2, April 1996.


## ACKNOWLEDGMENT


This research has been funded in part by the GOSTOP programme C3330-16-529000, co-financed by Slovenia and EU under ERDF, and by the EU's Horizon 2020 IA QU4LITY (GA no. 825030).