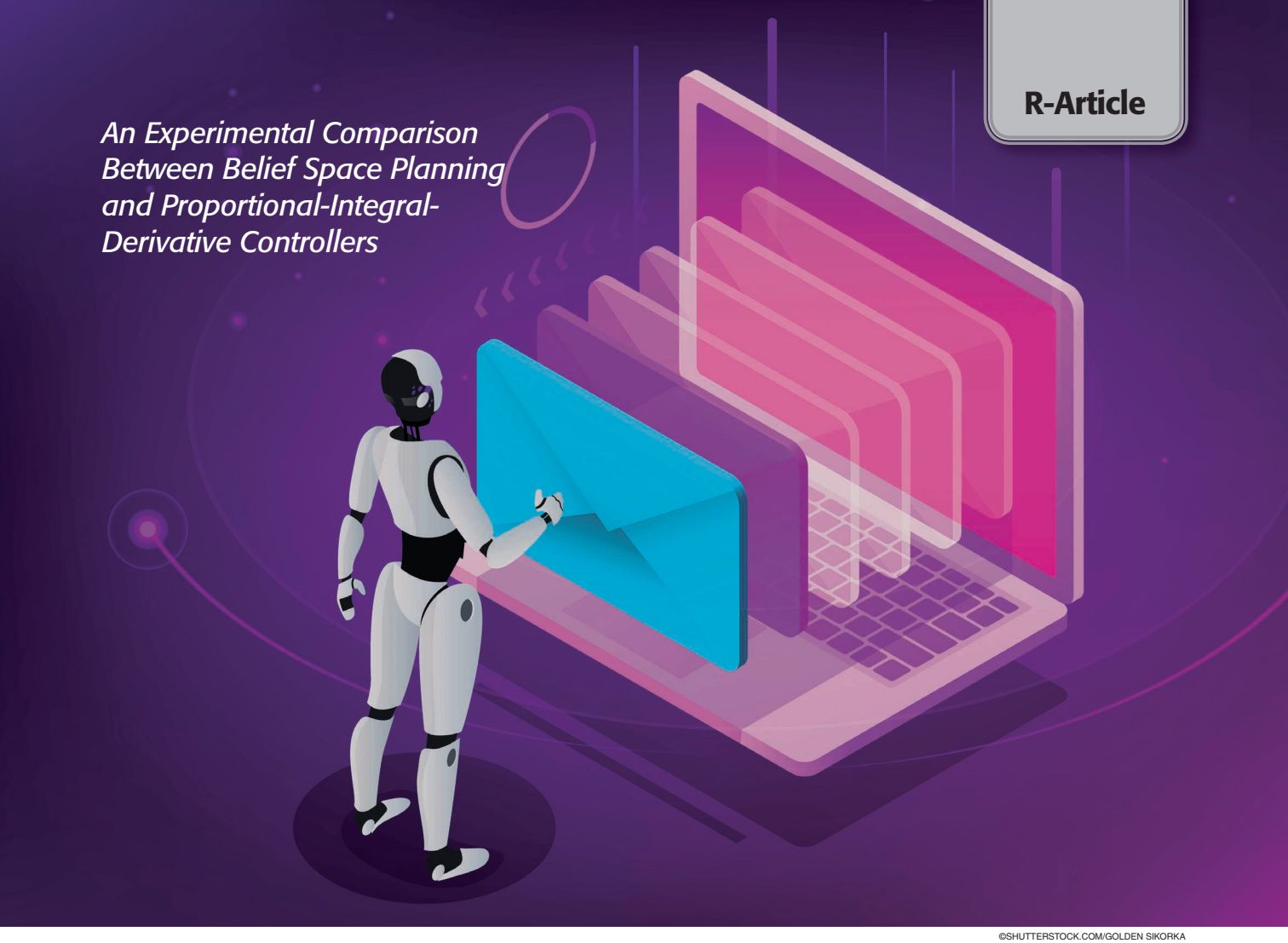
R-Article

An Experimental Comparison Between Belief Space Planning and Proportional-Integral-Derivative Controllers



A Simple Visual-Servoing Task on a Low-Accuracy, Low-Cost Arm

By Fabio Bonsignorio and Enrica Zereik

The main aim of this article is to provide an example of reproducible research in robotics. Despite the fact that a large number of researchers agree on the need of reproducibility in robotics and artificial intelligence, the practice of reproducible research is still in an embryonic phase. As a matter of fact, *IEEE Robotics and Automation Magazine* is, at the date of this article submission, the only top-tier robotics publication accepting "reproducible" articles. To this end, we have chosen a very common and traditional problem and approached it with a "first principles" attitude.

Many different methods have been proposed in the literature for the control of robotic arms. In this article, to validate our reproducible research platform and provide a template methodology for its usage, we have thoroughly compared, in a reproducible way, the performance of simple belief space planning (BSP) and proportional-integral-derivative (PID) controls when applied to a lightweight, low-accuracy, and compliant open source robot arm (i.e., H_2Arm). Both control

Digital Object Identifier 10.1109/MRA.2020.3014279
Date of current version: 2 September 2020

schemes were implemented in their simplest form and under basic assumptions. BSP significantly outperforms PID on this platform, but not with respect to (w.r.t.) all of the metrics. The findings are interesting by themselves. They also show how easily statistically weak results can lead to qualitatively wrong conclusions if the results are cherry-picked. This practice may be more widespread in the literature than would be desirable in many areas of robotics.

Research Reproducibility

Our main goal is to provide a complete software and hardware platform that allows for the statistical replication of our results and experimentation on other, more or less sophisticated control strategies and algorithms. We believe this is much needed because real-world comparison of control methods for robot arms is seldom seen in the literature [1], [2]. This makes it difficult to assess the real relevance and effectiveness of the many proposed solutions. It is interesting to observe that in the industrial domain—despite the wide number and variety of much more sophisticated existing control methods—the by far more widespread approach is based on PID control applied to heavy and highly accurate multibody rigid structures. The scarcity of reproducible experiments about the real-world application of control methods has thus far hampered the adoption of more advanced methodologies by industry and practitioners. In particular, it is a reasonable and widely shared idea that "classical" methods do not work properly on lightweight, low-accuracy, and compliant structures. The study of the most suitable control strategies for this kind of structure lends itself naturally to experimental investigation.

Among the many alternatives for the control of such structures, BSP methods are promising and comparatively simple to implement.

Among the many alternatives for the control of such structures, BSP methods are promising and comparatively simple to implement. As mentioned previously, we aim to contribute to the learning process of our community by providing a basic example of reproducible research in a traditional area such as robotic arm control and on a very affordable hardware and software platform. The affordability of the platform provides the additional benefit of leveling the ground and facilitating the inclusion within the research community of groups with limited resources and/or from disadvantaged areas of the world [3]. We follow a "first principles" approach. This means that we focus on the performed task rather than on the particular technical approaches used to accomplish it. Moreover, we deliberately compared two very different methodologies and implemented them with minimal technical sophistication. We hope that, in the future, other researchers will be able to build on our work. For example, it could be interesting to add an appropriate control observer to the PID controller and carry out a qualitative and quantitative comparison with our basic controller implementation. The possibility of comparing different approaches is still lacking today, and it is strongly needed for both the progress of basic research and the development of new applications.

The traditional "mechatronic" approach to robotics, as described in the major textbooks on the matter (e.g., [4]–[6]), mainly consists of the application of some (typically linearized) deterministic control strategy on a multi rigid-body (typically heavy) kinematic structure. Usually, the sensor measures are filtered by control observers. This basic structure underpins the great majority of "blind" robots that have been successfully utilized for many decades in automotive factories. Is this approach fit for the kind of structure we study? Although already obtained results seem suitable for application in structured or semistructured environments (such as manufacturing facilities or hospitals), they lack the robustness and adaptivity necessary to be employed in open-ended environments and, in general, for long-awaited and long-promised service robotics (elder care, home assistance, and so on).

Organization

This article is organized in three parts.

- *Magazine Article*: The text that you are currently reading. It provides an overview of our work, in particular the general idea, motivation, relevance, and obtained results. This could be enough for you if you do not aim to reproduce our work and results.
- *Supplemental Information*: This material can be found in supplemental information. It provides an extended version of our article. It details exploited Materials and Methods and obtained Results and Discussion. The description of the research is in accordance with the Euron GEM Guidelines [12]. It provides the user guide for colleagues who are interested in reproducing our results. It would be great if they could report their work as an R-article (see [1], [13]).
- *Data, Code, and Design Repository*: An online repository on IEEE DataPort [14] includes all the data, code, and hardware designs and details necessary and sufficient (we hope) to reproduce our experiments. For a discussion of the importance of open sourcing the hardware designs for robotics research, see [15].

Please note that, if you are interested in the scientific details of our work or in reproducing the results reported in this paper, you should check the supplemental information, where all the necessary details (not reported here for sake of brevity) are included.

A characteristic issue with "traditional" robotic arms deals with the modeling choices. The possibility to exploit a linear model for structures with nonlinear dynamics working at high speeds requires comparatively heavy weights for the structures themselves. This, consequently, has unwanted effects on the payload ratios of these manipulators, i.e., the ratio between the weight a robot can move and its own weight. A relationship of a 10-kg payload versus a 200–300-kg robot body is not uncommon. Another issue is that a rigid structure limits the grasping and manipulation capabilities of the robot in a number of situations (for example, when grasping partially unknown or limp objects). This limitation emerges even in partially unstructured environments or when (even slight) changes occur in the shapes and attitude of the object to be grasped and the operational space. Reducing weights and increasing compliance lead to a remarkable increase in nonlinearities and uncertainties in both dynamics and measures, thus rendering most widely used methods less reliable. These considerations make the experimental study of arm structures like the one we consider especially interesting.

In the industrial domain, the mainstream approach to robot arm control is the application of PID control schemes, usually framed into model-based and force control schemes (see, for example, [7]–[9]). The cost and functional benefits of totally or partially compliant arm mechanical structures are apparent. However, an extensive and reproducible study showing the limits of (even linear) control strategies applied to this kind of arm, not to mention other more sophisticated approaches, is missing. In robotics, control theory and engineering also need to be considered as an experimental discipline. In this article, we propose a platform for reproducible research on robotic arm control. To validate it, we carried out a performance comparison of standard PID and BSP controls implemented on the same open source hardware platform. Our work aims to provide a cheap open access and open source infrastructure, allowing objective comparisons of different control algorithms for lightweight tendon-driven arms. While doing this, we also contribute to the development of reproducible research best practices in robotics to cope with a pressing need in our community (see [1], [2], [10], and [11]).

Even from these few references, it is clear that objective performance comparison procedures are still missing. It is worth noting that an extensive BSP experimental study in the field and, in particular, on system platforms like ours has never been published.

System Setup

The main objective of our hardware/software platform consists of allowing the objective functional comparison of different approaches while implementing the same basic task. As will become clear to anyone trying to reproduce our results, giving others the opportunity to confirm (or falsify) experimental results requires a lot more information than what is usually provided, even in a detailed journal article [12]. Such details include experiment materials and methods but also adopted assumptions and criteria. We chose to perform an experiment as simply as possible, focusing on a visual-servoing problem: the control of the reaching movement of a low-cost and low-accuracy robot arm. The system, consisting of an arm and a basic in-hand video camera, was required to find and track a red blob (a wheel), as depicted in Figure 1. The

It is quite common for sophisticated control strategies to work very well in simulation but fail in the field.

(a) (b)

Figure 1. H_2Arm performing an experiment. (a) The robot tracking a red blob and (b) a vision algorithm detecting and segmenting a red blob.

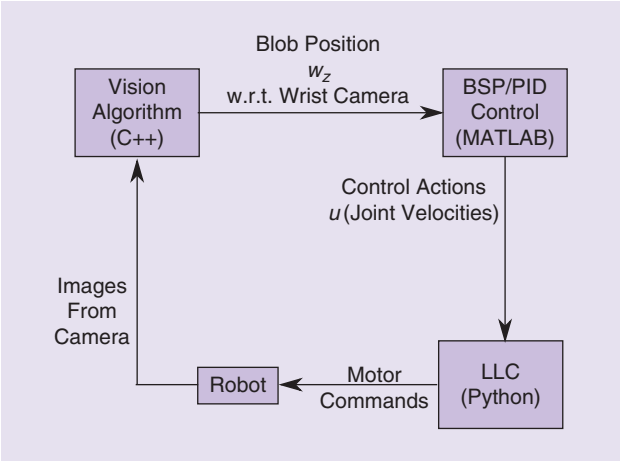


Figure 2. The software architecture. A block diagram of the system software architecture is 1) the vision algorithm is implemented in C++, relying on the OpenCV Library [16]; 2) the control scheme, both BSP and PID for comparison, is developed in MATLAB; and 3) the low-level control (LLC) generating the actual commands to motors is implemented in Python. Communication among the processes occurs through standard TCP/IP sockets.

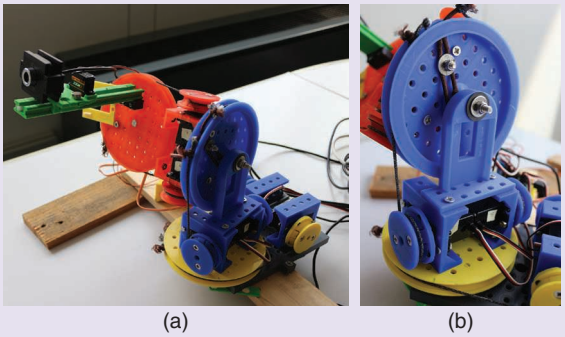


Figure 3. The 3D printed robot. The structure deliberately exhibits high inaccuracy when demonstrating the potentiality of the proposed control approach. (a) The overall 4-DoF structure and (b) a detailed view of joint J_2. DoF: degree of freedom.

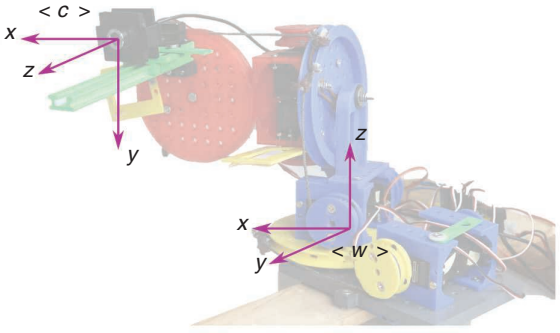


Figure 4. The reference frames chosen for the experiments. Symbol $< c >$ represents the camera frame centered on the webcam, and $< w >$ is the world frame, assumed to be placed in the center of the first joint (the big yellow horizontal disk).

arm reaches to the red object and stops when the blob, observed by the wrist-mounted camera, is in the image plane center, while the end effector is at a predefined distance from the blob itself.

In the "H_2Arm," "Vision Algorithm," and "Low-Level Motor Control" sections, the different components of the hardware and software architecture depicted in Figure 2 are described; the adopted control strategies are revealed in the "Control Strategies" section.

H_2Arm

H_2Arm is an open source arm that adapts the GummiArm concept to our purposes (see [17] and [18]). Both GummiArm and H_2Arm are intended as inaccurate platforms to challenge the development of suitable controllers, which necessarily cannot be "classical." The H_2Arm is far less expensive and more challenging to control. In fact, the GummiArm is endowed with accurate and precise motors that provide feedback from their encoders. The only sensor onboard the H_2Arm, on the contrary, is a wrist-mounted webcam (H_2Arm motors are not equipped with encoders and do not provide any feedback). We chose to employ a low-quality camera to keep our platform very cheap (its cost is estimated at less than €200 overall) but also to introduce errors and uncertainties on the only available measurement.

The research described in this article contributes to the long-term goal of developing robust and reliable algorithms able to govern visual sensor-guided manipulators affected by high uncertainties, low-quality actuation, low accuracy, and low repeatability in the physical structure. For this reason, we have designed a low-accuracy manipulator that is modular and 3D printed and equipped with a low-quality camera. This is a step toward our long-term targeted application to robustly address the underwater manipulation problem of a robotic arm mounted on a floating platform. In ongoing work not reported here, we aim to develop a robust and adaptive control framework able to cope with the problems deriving from measurement uncertainty, external environmental disturbance, and robot inaccuracy. These can be embedded in the algorithm and smoothly compensated for.

The developed robotic arm, depicted in Figure 3, has four degrees of freedom: the first three are rotational joints while the last is a prismatic joint. The first and second joints provide motion in the horizontal and vertical axes (x, y w.r.t. frame $< c >$ of Figure 4), respectively; the third joint is a wrist, rotating around axis z of reference frame $< c >$; while the last prismatic joint provides translation along axis z in reference to frame $< c >$.

Vision Algorithm

The computer vision algorithm, which seeks to retrieve a 3D space position of the object, is deliberately extremely simple. Errors and uncertainties in both the camera calibration and blob position measurements reach the control system

whether it is BSP or PID, which is requested to cope with them as appropriate. As shown in the "Experiments" section, the BSP control is able to correctly manage such errors while the PID control is severely affected and is quite often unable to complete the task. Because the task consists of reaching a predefined position of the end effector in front of the red blob (a wheel), the minimized error is the estimated distance between the end effector and the red wheel.

To retrieve in real time the blob's 3D space position from the camera w.r.t. the robot end effector, at each step the image is first acquired and pre-elaborated with a median filter smoothing. Then, it is converted in the hue, saturation, and value color space, which is more suitable for the following step, consisting of retrieving all of the red pixels in the input image. The original image is suitably masked, and a new image containing only red pixels is built up, as shown in the bottom right window of Figure 1(b). From this red image, the contours are retrieved and ordered by their size. The biggest one, assumed to be the red wheel, is approximated with a curve, and its minimum enclosing circle is computed to retrieve its dimension and center, as seen by the camera. Note how this choice could be error-prone whenever another big red blob appears in front of the camera; nonetheless, in the work described here, the simple assumption of having just one big red blob in the scene was intended to keep the image processing as simple as possible. In fact, the focus of the present work is not on the computer vision algorithm but on the performance of the BSP stochastic control algorithm in comparison to a classical PID control strategy.

Using the blob position and size in the image, a mask is created to provide an indication for the blob tracking in subsequent images (i.e., the tracking is limited in the neighborhood of the image portion where the blob was in the previous step). The radius of the found red blob is augmented by 5% in the computation of the tracking mask; this is essential to avoid cutting the blob in subsequent images due to the robot's motion. Whenever the blob is not found at some step, the mask is reinitialized with the dimension of the original input image. An example of the blob tracked in different images in subsequent time steps is reported in the top right window of Figure 1(b). At this point, the 3D space position of the red wheel is computed, exploiting the a priori known parameters (camera calibration and real-wheel dimension) and the camera measurements (blob center and radius). The obtained 3D blob position w.r.t. the end effector is sent at each sample time as a feedback to the arm control system. The vision algorithm is implemented in C++ and uses the OpenCV library [16] for image processing.

Low-Level Motor Control

The low-level motor control is a driver that receives the four-joint velocity references for the arm, computes the appropriate control signals, and sends them to the Pololu Maestro board, which is in charge of physically piloting the motors. Four different models have been implemented for the low-level control (LLC) and were tested in different experimental runs. Note that the different LLC models have been refined during the testing phase; they apply a different gain to the related motor axis according to one or more thresholds on the requested joint velocity.

Control Strategies

Our main aim is to start an experimental research thread on the control of tendon-driven lightweight, low-cost arms in the context of visual-servoing. To that purpose, we have developed a platform for the reproducibility of this kind of experiment. Platform validation has been carried out by comparing, in a reproducible way, the performance of both a PID- and BSP-based controller. In the "BSP Assuming Maximum Likelihood" and "PID Control" sections, we provided details of the two control approaches.

BSP Assuming Maximum Likelihood

A simple BSP algorithm, assuming maximum likelihood of the observations and using nonlinear optimization techniques, can be effectively used to control a visual-servoed arm, as shown (even if only in simulation) in [19]. A BSP-computed trajectory for the robot end effector moves it from its current position, assumed to be the sum of a signal component and a Gaussian noise part [represented by a Gaussian probability density function (PDF)], to a final position. This last position is again expressed as a Gaussian PDF with a mean value in the desired final position and lower covariance. It is quite common for sophisticated control strategies to work very well in simulation but fail in the field. It is also typical that field experiments of novel control approaches are reported in such a way that they can neither be reproduced nor compared against other approaches. We aim to contribute to this gap mitigation. To that end, the control scheme described in [19], from a theoretical standpoint, has been adapted to the problem at hand; noise structures more elaborate than the standard "vanilla Gaussian" were tested, and the results were encouraging.

The act of randomly moving the blob in the environment was devised to inject into the experiments an unpredictable "noise" factor of human origin.

The BSP algorithm calculates the actions (i.e., the joint velocities) to move the end effector from initial position A to desired position B (both represented by Gaussian PDFs) by weighting two different objectives: the goal reaching on one side and the covariance reduction in the mid- and endpoints on the other. Such control actions reduce the number of uncertainties on the trajectory and end-effector positions. This is equivalent to stating that the controller actually performs information-gathering actions while controlling the arm movement.

The proposed procedure is summarized in Algorithm 1. The BSP algorithm requires as inputs the initial belief state mean m_0 and the final goal m_{goal}, both representing the mean of the Gaussian PDFs modeling our beliefs on the initial and final positions (p_0 and p_{goal}), and returns the sequence $u_{1:s}$ of the control actions, where s is the number of discrete steps executed by BSP at each iteration.

The algorithm initializes all of the variables to proper values via the function `InitBSP` (line 1). The BSP strategy is then applied until the norm of the Cartesian error e_t for the end effector decreases below a predefined value thr_2 (line 2). For each step (line 4), a plan is calculated via the `createPlan` function (line 3). The `for` loop starts at `iCnt` to keep the correct previous actions in case a replanning occurs at a certain point in the loop without discarding the entire computed plan. Thus, $(\bar{u}_{1:s}, \bar{m}_{1:s})$ are obtained, which are the sequences of the needed control actions to reach the goal and related belief states, respectively. From this plan, the optimal control action u_t is computed through a linear quadratic regulator step (line 6). The distance z_t between the goal and the end effector is estimated (line 7), and, finally, the next belief state m_{t+1} is propagated through an extended Kalman filter on line 9. Note that z_t is computed by the vision algorithm and ξ represents the noise affecting the measurements. In case the norm of the distance between the planned and the actual belief state ($\bar{m}_t$ and m_t, respectively) is above a predefined threshold thr_1 (line 10), the current plan is discarded (line 18) because it is not suitable for correctly driving the end effector toward the goal. In this occurrence, a new plan is computed, and the procedure starts again. Note that variables *rplCnt* and *iCnt* are counters in charge of correctly managing the replanning phase (lines 11–15).

If, on the other hand, the control plan is able to suitably decrease the distance from the goal, the computed control action u_t is sent (line 19) to the arm LLC that applies the requested velocities to the motors. The process is iterated until the final goal is reached. Index t is not explicitly incremented in the algorithm. This is because the BSP time step is not fixed but dynamically depends on the optimization. For this reason, for time-series reconstruction and storage it is necessary to use an external synchronization mechanism: during the real-time experiments, time is queried to the CPU and stored, allowing for reconstruction of the time evolution of all the variables.

Algorithm 1: The BSP Algorithm.

Input: m_0, m_{goal}
Output: $u_{1:s}$

```
1:  initBSP ();
2:  while ‖e_t‖ > thr_2 do
3:      (ū_{1:s}, m̄_{1:s}) ← CreatePlan(m_t, m_goal);
4:      for i = iCnt to s do
5:          t = GetSystemTime();
6:          u_t ← LQR(ū_t, m̄_t, m_t);
7:          z_t ← MeasureBlob3Dposition();
8:          e_t ← m_goal − z_t + ξ;
9:          m_{t+1} ← EKF(m_t, u_t, z_t);
10:         if ‖m̄_t − m_t‖ > thr_1 then
11:             rplCnt ← rplCnt + 1;
12:             if rplCnt%5 == 0 then
13:                 iCnt ← 1;
14:             else
15:                 iCnt ← i;
16:             break;
17:         DriveEE (u_t);
```

Algorithm 2: The PID Algorithm.

Input: p_0, p_{goal}
Output: u_t

```
1:  initPID ();
2:  while ‖e_t‖ > thr_2 do
3:      t = GetSystemTime();
4:      dt = t − t';
5:      z_t ← MeasureBlob3Dposition();
6:      e_t ← p_goal − z_t + ξ;
7:      e_t∫ = e_t∫ + e_t dt;
8:      u_t = k_p e_t + k_i e_t∫ + k_d (e_t − e_t')/dt;
9:      t' = t;
10: DriveEE (u_t);
```

PID Control

The BSP controller has been compared to a standard PID with fine-tuned gains k_p, k_i, and k_d. In particular, the H_2Arm's implemented controller has a significant proportional component, a smaller integral one, and a small derivative component, which are introduced to avoid potential overshooting. During field applications, even in research activities, PID controllers are often chosen for reasons that have not been theoretically explored in depth thus far. This inspired us to compare BSP against PID. We believe that the findings are interesting and, more importantly, that our platform and reproducible results will allow an for objective comparison of different methods by other researchers. Some preliminary tests (besides those reported in the following and belonging to the three main test campaigns) were performed to tune the PID gains to improve the system's overall performance without degrading its stability.

The PID control loop is shown in Algorithm 2. It requires as inputs the initial and desired blob 3D position p_0 and p_{goal}, respectively, and outputs control action u_t. Here, p_0 and p_{goal} represent the actual 3D position and not a PDF mean, as in the BSP case. After a first

initialization phase (line 1) in which all of the variables are correctly instantiated, the control loop computes, at each step, the current system time t (line 3) and the elapsed time dt (line 4) and receives the measured 3D blob position from the vision-based algorithm (line 5). Note that t' is initialized, along with many other variables, in the `initPID` function. Then, the residual Cartesian error e_t is computed as the difference between the reference p_{goal} and the measured 3D position z_t (eventually, with the injection of the additional noise ξ) in line 6. Finally, the error integral $e_t\int$ is computed (line 7). The control action u_t to be sent to the LLC via the `DriveEE` function (line 11) is obtained through the classical PID expression $u_t = k_p e_t + k_i \int_0^t e_\tau d\tau + k_d \dot{e}_t$ (line 8). Other details of PID implementation (correspondence among control parameters, the number of successful and total runs, different test sets, the LLC model, and the parameter set) are omitted here for the sake of brevity. They can be found in the supplemental information.

Experiments

We performed an extensive experimentation with the goal of 1) validating our hardware/software platform, thus enabling the reproduction of our results; 2) testing the BSP and PID controllers on a specific, challenging real-world task; 3) comparing their results; and 4) studying the features of our simple manipulator.

The same "LLC" and "Vision Algorithm" blocks in Figure 2 were used for the BSP and PID experimentation to ensure that the observed differences in the performance depended on the controllers only.

Table 1. The main results of both PID and BSP control algorithms in the three performed test campaigns.

—	—	PID						BSP					
Test Campaign	**Category**	**Good**	**Total**	**%**	T_M (s)	T_m (s)	$\bar{T}$ (s)	**Good**	**Total**	**%**	T_M (s)	T_m (s)	$\bar{T}$ (s)
First	Fixed blob, no noise	4	12	**33.3**	88	3.2	**31.3**	8	11	**72.7**	385.6	10.8	**96.6**
	Moving blob, no noise	5	6	**83.3**	137.7	9.2	**48.8**	5	5	**100**	159.9	44.1	**97.2**
	Moving blob, two-axis noise, $\mathbf{w} = 0.05$	13	30	**43.3**	312.7	1.9	**40.4**	10	10	**100**	125.7	27.5	**86.6**
	Moving blob, three-axis noise, $\mathbf{w} = 0.025$	3	11	**27.3**	49.6	13	**23.1**	15	15	**100**	340.9	34	**128**
	Moving blob, three-axis noise, $\mathbf{w} = 0.05$	1	12	**8.3**	47.9	13.4	**28.5**	21	22	**95.5**	442.3	54.6	**191.8**
Second (optimized system)	Fixed blob, no noise	12	15	**80**	83.2	3.5	**26.2**	17	19	**89.5**	144.2	7.1	**46.7**
	Moving blob, three-axis noise, $\mathbf{w} = 0.025$	2	6	**33.3**	41.9	12.9	**22.7**	6	6	**100**	238	66.1	**131.3**
	Moving blob, three-axis noise, $\mathbf{w} = 0.035$	3	6	**50**	37.3	6.7	**23.7**	6	6	**100**	224	22.1	**94.2**
Third (uncalibrated system)	Fixed blob, no noise	0	5	**0**	58.3	9.8	**28.3**	4	5	**80**	56.1	15.1	**34**
	Moving blob, three-axis noise, $\mathbf{w} = 0.035$	1	7	**14.3**	54.5	5.3	**25.1**	5	5	**100**	129.8	39.0	**87.3**
TOTAL	all categories, all campaigns	44	110	**40**	312.7	1.9	**31.5**	97	104	**93.3**	442.3	7.1	**109.6**

$\bar{T}$, T_M, and T_m stand for the experiments' mean, maximum, and minimum time duration, respectively, while w is related to the additional injected noise. The time statistics presented here do not distinguish between failed and successful runs. For detailed data, refer to [14].

We performed more than 200 experimental runs subdivided into three different test campaigns—each of which was in a different condition—for both the BSP and PID experiments:

- *Fixed blob, no additional noise*: The blob was kept still in the environment, and no additional noise was injected into the system.
- *Moving blob, no additional noise*: The blob was randomly moved by hand in the environment during the task completion, and no additional noise was injected into the system.
- *Moving blob, two-axis noise*: The blob was randomly moved by hand in the environment during the task completion, and an additional noise was injected into the system feedback. Noise was added only to the x, y components of the estimation (w.r.t. frame $< c >$ in Figure 4) while the estimated distance (along the z-axis of frame $< c >$) from the blob was kept unaffected.
- *Moving blob, three-axis noise*: The blob was randomly moved by hand in the environment during the task completion, and an additional noise was injected into the system on all of the three feedback linear components x, y, and z (again, w.r.t. frame $< c >$ in Figure 4).

The first test campaign was performed with an initial calibration of the system parameters retrieved through a small number of preliminary runs (probably comparable to the number of tests usually executed in an average robotics research work). The second test campaign was based on the system parameter optimization coming from the first campaign. However, during the third test campaign, a system parameter uncalibration was deliberately caused to test the respective robustness of the compared control systems. To be fair and to introduce an equal uncalibration for both BSP and PID, we "mistuned" the LLC. As a consequence, the PID and BSP control systems faced the same issues in comparable conditions. In the second and third campaigns, only the more interesting experimental categories resulting from the first campaign were executed and investigated. The obtained results are reported in Table 1.

The act of randomly moving the blob in the environment was devised to inject into the experiments an unpredictable "noise" factor of human origin. This kind of disturbance is typically challenging for many robotics applications as it is difficult to model. Furthermore, because the long-term target is the application of the BSP technique for the control of a manipulator mounted on a floating platform within a marine environment, the choice of the additional noise to be injected in the system was done accordingly. The Pierson–Moskowitz spectrum [20], an empirical description of the ocean energy distribution consolidated in the literature, was exploited for noise generation. We find it interesting that the BSP controller showed a comparable robustness to added noise with nonuniform frequency characteristics.

The results of the first test campaign are reported in the top part of Table 1. A total number of 63 and 71 experimental runs were conducted for the BSP and PID control systems, respectively, distributed among five different categories. The BSP control was, on average, much more reliable than the PID control, even in not particularly challenging conditions.

The results of the second test campaign can be found in the middle part of Table 1. They basically confirm the greater robustness of BSP w.r.t. the PID controller. In particular, if we focus only on the experimental runs conducted after parameter optimization and with additional noise, the BSP and PID control schemes were able to complete the tasks in 100% and 41.7% of the cases, respectively.

The results of the third experimental campaign are reported in the bottom part of Table 1. They show how much the PID control scheme can be impacted by even a minimal "miscalibration" of the system LLC.

The PID's ability to complete the task dropped from 80% to 0% for the optimized and uncalibrated systems, respectively, in the "fixed blob, no noise" case. In the "moving blob, three-axis noise" case, the same probability dropped from 50% to 14.3% for the optimized and uncalibrated systems, respectively. BSP's ability to complete the task in the "fixed blob, no noise" case decreased from the 89.5% to the 80%, while it did not change in the "moving blob, three-axis noise" case (100%).

Table 2. A comparison between BSP and PID control results in terms of mean, maximum, and minimum execution time $\bar{T}$, T_M, and T_m respectively, obtained within the third performed test campaign.

	Fixed Blob, No Noise			Moving Blob, Three-Axis Noise w = 0.035		
BSP control—uncalibrated system						
	T_M (s)	T_m (s)	$\bar{T}(s)$	$T_M(s)$	$T_m(s)$	$\bar{T}(s)$
Successful Experiment	56.1	15.1	**37.6**	129.8	39	**87.3**
Failed Experiment	19.6	19.6	**19.6**	–	–	–
PID control—uncalibrated system						
	T_M (s)	T_m (s)	$\bar{T}(s)$	$T_M(s)$	$T_m(s)$	$\bar{T}(s)$
Successful Experiment	–	–	–	22.3	22.3	**22.3**
Failed Experiment	58.3	9.8	**28.3**	54.5	5.3	**25.6**

The time performance is reported for the BSP and PID control schemes for both their successful and failed runs. For each category, whenever no data were available (i.e., no experimental runs belonged to this category), a dash is drawn. When only one experiment belonged to the corresponding category, the same value is reported for all $\bar{T}$, T_M, and T_m because these values are equal and coincide with only the experiments' time duration.

Note that, in the "uncalibrated" experiments, the already tuned k_p, k_i, and k_d gains of the PID controller were maintained. This means that further sensitivity and misbehavior could be provoked in the PID control scheme; for the BSP scheme, the only possible miscalibration source is related to the manipulator LLC (i.e., the robot's hardware). The third test campaign illustrates the greater robustness of BSP w.r.t. to the PID. In particular, when considering the more realistic case of additional noise injection, the BSP controller was able to complete the task in 100% of the cases compared to the 14.3% success rate of the PID.

The BSP controller successfully completed 97 experimental runs out of a total of 104, while the PID controller completed the task in 44 cases out of 110. The experiments showed that the distribution of the final Cartesian error norm is quite similar whether the BSP or PID controller is used. The PID controller was definitely faster than the BSP (refer to Tables 1 and 2); in contrast, the BSP technique was more reliable in task completion. Moreover, the BSP controllers showed better performance when the test runs were affected by noise.

The PID controller was able to complete the task in the third test campaign in only one case out of 12 tests, considering both the tested categories. For this reason, Figure 5 reports the final error norm and time duration of the successful BSP runs after system uncalibration

The results showed that, on the H_2Arm, the BSP algorithm was systematically more robust because it almost always allowed for task completion.

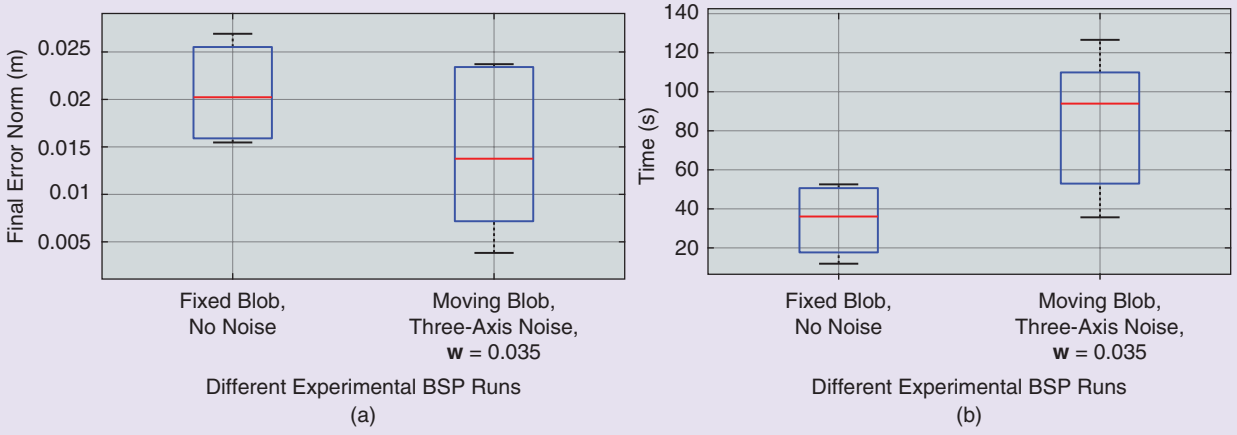


Figure 5. Boxplot diagrams showing (a) the final error norm and (b) the time duration for all successful BSP experiments of the third test campaign after the system's parameter uncalibration.

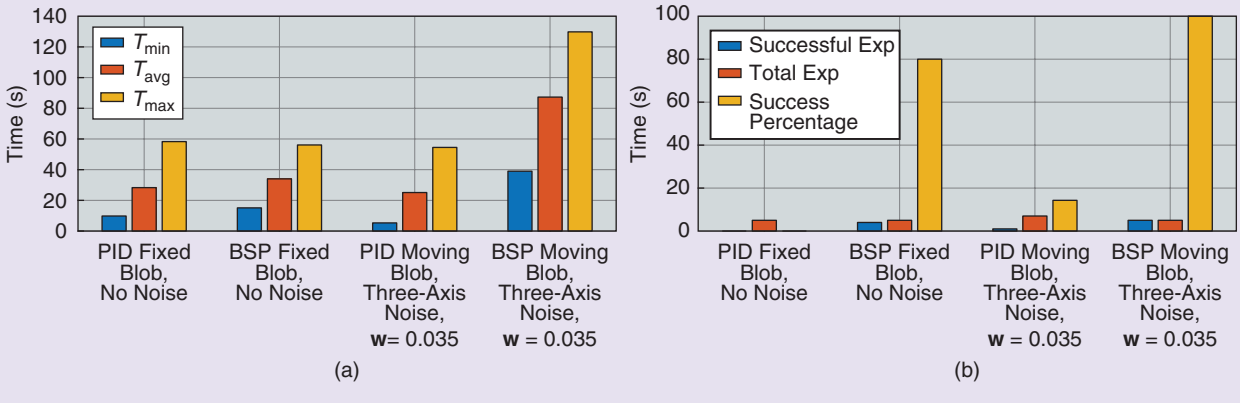


Figure 6. The third test campaign performance reported for (a) the execution time (minimum, mean, and maximum) and (b) the successful and failed runs. Exp: experiments.

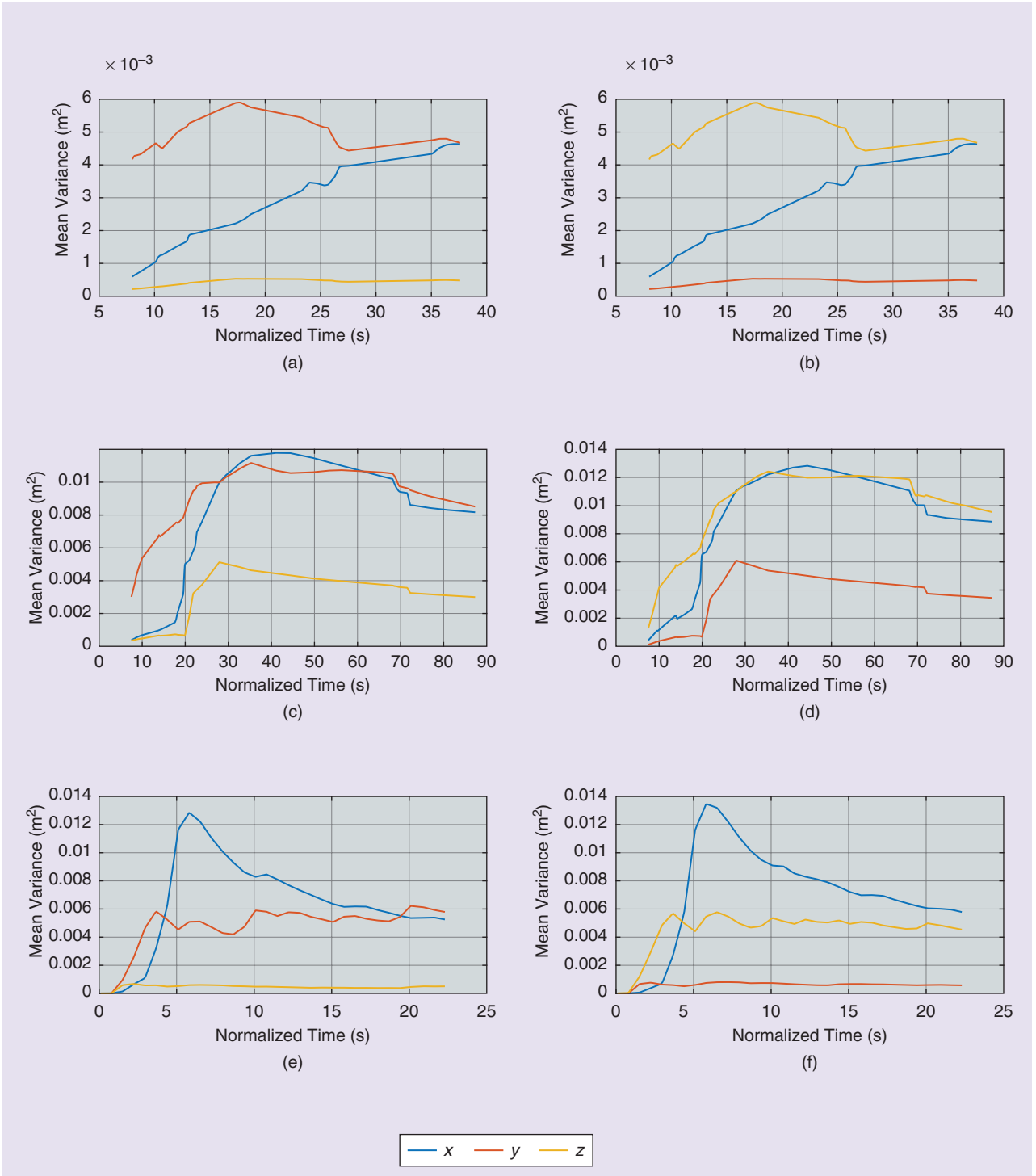


Figure 7. The time evolution of mean posterior variance of the blob position measurement (left column) and of the end-effector Cartesian error (right column) throughout successful experiments in the different categories of the third test campaign. No PID run in the "fixed blob, no noise" case was successful. As expected, the graphs in the right column are very similar to those in the left column. (a) and (b) BSP with fixed blob, each with no noise. (c) and (d) BSP moving blob, each with three-axis noise, $\mathbf{w} = 0.035$. (e) and (f) PIDs with moving blobs, each with three-axis noise, $\mathbf{w} = 0.035$. [Just one experiment in each of (e) and (f) was successful.]

only. Table 2 summarizes the execution-time statistics for both PID and BSP control in the third test campaign, considering the two different categories and remarking successful and failed runs. The same data reported in Tables 1 and 2 are presented in bar diagrams in Figure 6. Moreover, in Figure 7 the time evolution of the mean posterior variance of both the blob position measurement and end-effector Cartesian error is reported for the third test campaign. A more detailed discussion of the results can be found in the supplemental information.

Conclusions and Future Work

We developed a platform for reproducible research and objective comparisons of different methods for a cheap, low-accuracy, lightweight robotic arm. Additionally, we performed an extensive comparison of two very different technical approaches for the execution of the same simple task. The results showed that, on the H_2Arm, the BSP algorithm was systematically more robust because it almost always allowed for task completion. On the contrary, the PID completed the task in only a subset of cases, but, in general, it converged more quickly. Our research also showed that the qualitative conclusions of studies like ours depend on the number of executed tests: we may have cherry-picked the successful PID tests and concluded that, on the H_2Arm, a simple PID implementation worked properly. It is important to note that the qualitative and quantitative results of the comparison may change if some refinements are introduced, for example, a more sophisticated BSP approach or a control observer suitably integrated in the PID controller. After establishing an objective ground for comparison, this research topic appears very interesting for the entire robotics community. The reproducibility of our results allows for their replication by other research groups and the implementation and comparison of other control strategies on the same platform.

Our future work will focus on the application of deep (reinforcement) learning and other control strategies on the same hardware platform. Furthermore, we are interested in bringing some changes to the hardware platform and assessing their impact on the implemented control strategies. We also plan platform improvements to obtain better computing efficiency; the first step toward this goal consists of porting the BSP optimization from MATLAB to C++. We hope that other researchers will build on our work due to the reproducibility of our results. We are aware that ours is just a small, yet necessary step toward making reproducible research a mainstream practice in our field.

References

[1] F. Bonsignorio, "A new kind of article for reproducible research in intelligent robotics," *IEEE Robot. Autom. Mag.*, vol. 24, no. 3, pp. 178–182, 2017. doi: 10.1109/MRA.2017.2722918.

[2] F. Bonsignorio and A. P. D. Pobil, "Toward replicable and measurable robotics research," *IEEE Robot. Autom. Mag.*, vol. 22, no. 3, pp. 32–35, 2015. doi: 10.1109/MRA.2015.2452073.

[3] A. Tsanni, "African scientists leverage open hardware," *Nature*, vol. 582, no. 7810, p. 138, 2020. doi: 10.1038/d41586-020-01606-z.

[4] J. J. Craig, *Introduction to Robotics: Mechanics and Control*, 3rd ed. London: Pearson, 2003.

[5] O. Khatib and B. Siciliano, *Springer Handbook of Robotics*, 2nd ed. New York: Springer-Verlag, 2016.

[6] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*. 2nd ed. New York: Springer-Verlag, 2017.

[7] F. Nagata and K. Watanabe, *Controller Design for Industrial Robots and Machine Tools*, 1st ed. Sawston: Woodhead Publishing, 2013.

[8] T. Brogårdh, "Present and future robot control development: An industrial perspective," *Annu. Rev. Control*, vol. 31, no. 1, pp. 69–79, 2007. doi: 10.1016/j.arcontrol.2007.01.002.

[9] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. New York: Springer-Verlag, 2009.

[10] J. L. S. Rincon and S. Carpin, "Time-constrained exploration using topo-semantic spatial models: A reproducible approach to measurable robotics," *IEEE Robot. Autom. Mag.*, vol. 26, no. 3, pp. 78–87, 2019. doi: 10.1109/MRA.2019.2923452.

[11] "Robotics and the art of science," *Nature Mach. Intell.*, vol. 1, p. 259, June 2019. doi: 10.1038/s42256-019-0066-8.

[12] F. Bonsignorio, J. Hallam, and A. del Pobil, "Good Experimental Methodology: Gem Guidelines," Euron Gem Sig Rep., 2008. [Online]. Available: http://www.heronrobots.com/EuronGEMSig/downloads/GemSigGuidelinesBeta.pdf

[13] "RAM information for authors," IEEE Robotics & Automation Society, New York. Accessed on: Aug. 5, 2020. [Online]. Available: https://www.ieee-ras.org/publications/ram/information-for-authors-ram

[14] E. Zereik, F. Bonsignorio, and A. Odetti, "H2Arm BSPvsPID repository," IEEE-DataPort. Accessed on: Aug. 5, 2020. [Online]. Available: https://ieee-dataport.org/open-access/h2arm-bsp-vs-pid-experiments

[15] A. Dollar, F. Mondada, A. Rodriguez, and G. Metta, "Open-source and widely disseminated robot hardware [from the guest editors]," *IEEE Robot. Autom. Mag.*, vol. 24, no. 1, pp. 30–31, 2017. doi: 10.1109/MRA.2016.2646068.

[16] "Computer Vision Community," OpenCV library. 2019. Accessed on: Aug. 5, 2020. [Online]. Available: https://opencv.org/

[17] M. Stoelen, F. Bonsignorio, and A. Cangelosi, "Co–exploring actuator antagonism and bio–inspired control in a printable robot arm," in *Proc. Int. Conf. Simulation Adaptive Behavior (SAB 2016)*, 2016, pp. 244–255. doi: 10.1007/978-3-319-43488-9_22.

[18] M. Stoelen, "GummiArm repository," Github, 2019. [Online]. Available: https://github.com/mstoelen/GummiArm

[19] R. Platt, R. Tedrake, L. Kaelbling, and T. Lozano-Perez, "Belief space planning assuming maximum likelihood observations," in *Proc. Robotics: Science and Systems*, Zaragoza, June 2010. doi: 10.15607/RSS.2010.VI.037.

[20] W. J. Pierson and L. Moskowitz, "A proposed spectral form for fully developed wind seas based on the similarity theory of S. A. Kitaigorodskii," *J. Geophys. Res.*, vol. 69, no. 24, pp. 5181–5190, 1964. doi: 10.1029/JZ069i024p05181.

Fabio Bonsignorio, Heron Robots, Genoa, Italy. Email: fabio.bonsignorio@heronrobots.com.

Enrica Zereik, Italian National Research Council, Institute of Marine Engineering, Genoa, Italy. Email: enrica.zereik@cnr.it.