A ROS-based kinematic calibration tool for serial robots

Caroline Pascal, Olivier Doaré, Alexandre Chapoutot

A ROS-based kinematic calibration tool for serial robots*

Caroline Pascal[1], Olivier Doaré[2], Alexandre Chapoutot[1]

***Abstract*— The use of serial robots for industrial and research purposes is often limited by a flawed positioning accuracy, caused by the differences between the robot nominal model, and the real one. Such an issue can be solved by means of kinematic calibration, which is usually a tedious and intricate task. In this paper, we propose a complete kinematic calibration procedure relying on established geometric modeling, measurements design and parameters identification methods, as well as multiple integration tools, to provide a high adaptability and a simplified handling. The overall process was bundled up in a ROS-based modular and user-friendly package, whose main objective is to offer a smooth and fully integrated framework for the kinematic calibration of serial robots. Our solution was successfully tested using a motion tracking device, and allowed to increase the overall positioning accuracy of two different serial robots by 75% in a matter of hours.**

I. INTRODUCTION

In the near future, serial robots are to become an essential asset in the development of an ageing industry, by offering autonomous and reproducible means of manufacturing and production control [1–3]. In particular, time-consuming metrology operations, aiming to ensure the mechanical compliance and manufacturing quality of produced parts, are perfect candidates for automation using robotic arms [4, 5]. Yet, considering the demanding precision requirements of current metrology applications, the flawed absolute positioning accuracy of serial robots is an unfortunate issue.

Fortunately, this issue has been the focus of an extensive research work in the last decades, and was mainly answered by the kinematic calibration of robots. The main idea behind kinematic calibration is to estimate the values of the geometric parameters involved in the kinematic modeling of serial robots, with respect to experimental observations [6].

A kinematic calibration procedure is usually split in three steps: (1) the kinematic modeling of the serial robot, (2) the choice of the configurations in which the observations are to be performed and their practical implementation, and (3) the identification of the parameters exhibited in (1) according to the observations done in (2). To ensure the efficiency of the results, an additional verification step is also performed, and consists in a direct comparison between the calibrated kinematic model output, and wisely chosen observations. Finally, the identified geometric parameters are either directly overwritten in the robot controller (correction), or passed to a post-processing adaptive tool (compensation) [7]. Nevertheless, as most efforts have been focused on improving each calibration tasks separately [1], the effective roll-out of calibration still suffers from the lack of its complete and smooth integration into an easily and effectively usable tool.

Related work: The directly embedded and sensor-less calibration routines offered by robots manufacturers, usually only allows for a brand dependant, partial [8, 9] or restricted [10] identification. More complete and generic solutions are provided by proprietary external software, such as RoboDK [11], which handles all steps from the gathering of experimental data to the identification of the robot kinematic parameters. Yet, the details of the implementation remain opaque, with no clear insight about the hardware and software interfaces, or the actual parameters identification process. Several open-access solutions were also developed, but none of them propose an actual smooth and full integration of the calibration procedure. The software GECARO [12] effectively deals with the design of observations, and the identification of kinematic parameters, but is completely unrelated from both the robot, and the experimental data gathering parts. The integration of the robot as an integral part of the whole process is tackled by ROSY [7], featuring a simulation model to be adjusted to the real mechanism, and COMET [2], offering a catalog of predefined robot models. Yet, both of them are facing integration issues regarding the heterogeneity of proprietary programming languages and communication protocols. Concerning the practical embedding of the experimental observations, the resort to an external measuring device, in an opened-loop way, has drawn much attention. Various authors [3, 13, 14] proposed solutions highlighting the connections between the measuring device, the robot and the main calibration process, without taking advantage of their potential for a smooth and complete integration of all calibration steps. In parallel, Ginani and Motta [15] ensure the interchangeability of the measuring device, provided that the broadcast data satisfy a given interface, but with no clear sign of collaboration. Additionally, most of these solutions are not easily accessible in a directly usable form, and offer small algorithmic adaptability, in regards with quickly evolving techniques. Looking towards a smoother hardware-software interface and increased modularity, the Robotic Operating System (ROS) middleware [16], holds promising prospects. Several free access packages dedicated to the kinematic calibration of serial robots have indeed already been developed [17, 18]. Yet, they mainly rely on the use of monocular cameras, and rather focus on the challenging

*This work was partially supported by Carnot TSN, Carnot Mines and the CIEDS project APRO.

[1]Caroline Pascal and Alexandre Chapoutot are with U2IS, ENSTA Paris, Institut Polytechnique de Paris, 828 boulevard des Maréchaux, 91120 Palaiseau, France caroline.pascal.2020@ensta-paris.fr alexandre.chapoutot@ensta-paris.fr

[2]Olivier Doaré is with UME, ENSTA Paris, Institut Polytechnique de Paris, 828 boulevard des Maréchaux, 91120 Palaiseau, France olivier.doare@ensta-paris.fr

hand-eye calibration issue, than on the integration aspects triggered by the calibration of serial robots.

Problem statement: Eventually, kinematic calibration procedures, which remain crucial to ensure serial robots accuracy, have become a burdening task, suffering from (1) an unequal integration of the fundamental calibration steps, (2) discontinuous hardware-software interfaces, and (3) impractical algorithmic adaptability.

Contributions: We present a novel and complete kinematic calibration procedure, which allows to automatically calibrate a serial robot, from its kinematic modeling to the verification of the final identified model. This tool, based on the ROS middleware, allows for a comprehensive and smooth integration of the calibrated serial robot and measuring device, with the only requirements that these devices are compatible with ROS. Finally, having in mind that each step of the kinematic calibration process is likely to be improved separately, the proposed architecture is very modular, hence facilitating the integration of new algorithms. The project is available at `https://gitlab.ensta.fr/pascal.2020/robot_arm_calibration`.

Paper organization: In Section II, we detail the steps contained in our kinematic calibration procedure, and the challenging hardware-software interfaces are discussed in Section III. Section IV presents an experimental validation of the proposed approach, followed by discussions about future work and conclusion in Section V.

II. KINEMATIC CALIBRATION PROCEDURE

In this section, the theoretical aspects as well as the algorithmic implementation of all four steps contained in our kinematic calibration procedure are detailed (*c.f.*, Figure 1).

A. Step 1: Geometric modeling of serial robots

The modeling of the robot kinematic chain, *i.e.*, the relationship between the robot state and the position and orientation of the end effector, is a crucial step, as it conditions the parameters to be identified [6]. In practice, defining a faithful model may become a complex task, depending on the diversity and complexity of the modeled phenomena: flexibilities, gear-induced drift, thermal effects, etc. Yet, discrepancies caused by manufacturing and assembling defects appear to have the most determining impact on the robot accuracy [15, 19, 20] and benefit from a simplified and often sufficient geometric modeling.

In addition to the compliance with the robot true behaviour, calibrated models must meet continuity, completeness, and minimal requirements, in order to ensure a reliable parameters identification [21]. Efforts have been made in order to streamline the generation of such models, based on usual representations, such as the widespread Denavit-Hartenberg parameters [20–22]. Following the systematic method presented in [23], we developed a fully automated Python routine, able to extract the analytic geometric model of a serial robot based on its Denavit-Hartenberg parametrisation, or directly using an URDF file.

Taking into account the geometric parameters related to the robot base, end effector, links and joints, the pose of the end effector against a fixed reference frame is given by

$$T(Q,\pi) = T_{Base}(\pi_{Base}) \cdot [T_{Joint_1}(q_1, \pi_{J_1}) \cdot T_{Link_1}(\pi_{L_1})] \cdot [T_{Joint_N}(q_N, \pi_{J_N}) \cdot T_{Link_N}(\pi_{L_N})] \cdot T_{EE}(\pi_{EE}) \quad (1)$$

where each T-indexed matrix defines the corresponding local homogeneous transformation matrix, and $Q = (q_k)_{k=1,\dots,N}$ describes the N joints angles.

For each part of the robot, the corresponding π-indexed vector contains the relevant identifiable geometric parameters, which will either be translational or rotational depending on the robot layout (see [23] for more details). All vectors are then concatenated into a single vector π of size N_π, such that π_0 denotes the nominal values of the robot geometric parameters.

Based on this analytic formulation, our routine also computes the so-called identification Jacobian matrix:

$$J_\pi = \left(\frac{\partial T(Q,\pi)}{\partial \pi_i} \right)_{i=1,\dots,N_\pi} . \quad (2)$$

Both the robot model and identification Jacobian matrix are compiled into callable Python functions, and stored in a dedicated file for further use.

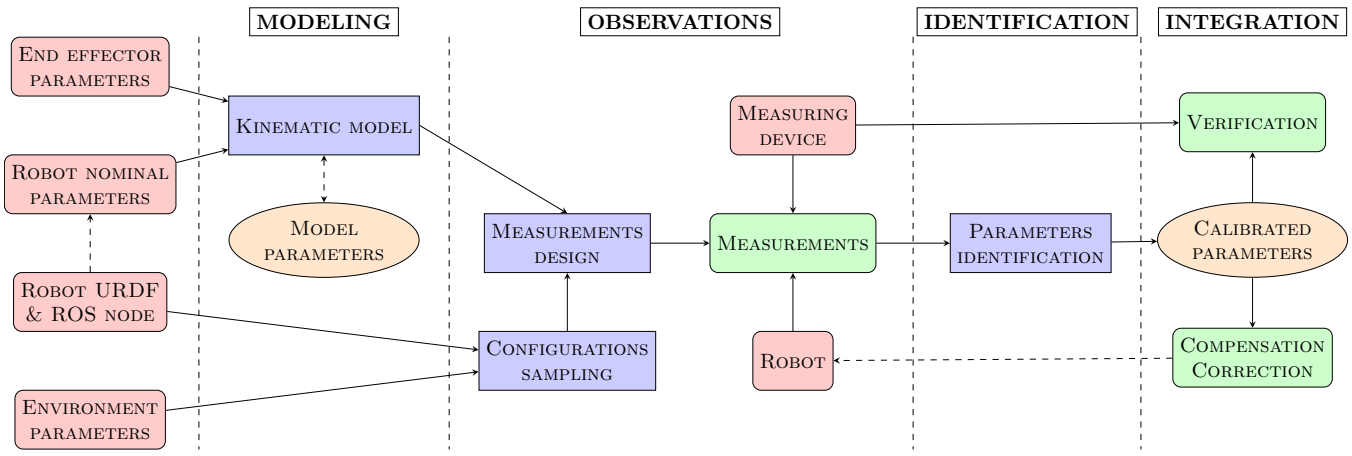


Fig. 1: Description of the overall kinematic calibration procedure

B. Step 2: Experimental observations design and execution

For the previously defined model to accurately describe the behaviour of the actual robot, its parameters must be identified regarding to experimental observations of the robot. Such observations can either be performed in a closed-loop way, where the robot end effector is linked to the robot base [2], or geometrically constrained [12], or in an open-looped way, using an external measuring device. Even though the latter implies a more complex and expensive setup, its usual use and simpler implementation makes it our default choice.

Open-loop measurements consist in moving the robot into several configurations, in which the state of the robot is measured. In the case of a geometrical approach, both the joints angles and the pose (*i.e.*, position and orientation) of the end effector are monitored.

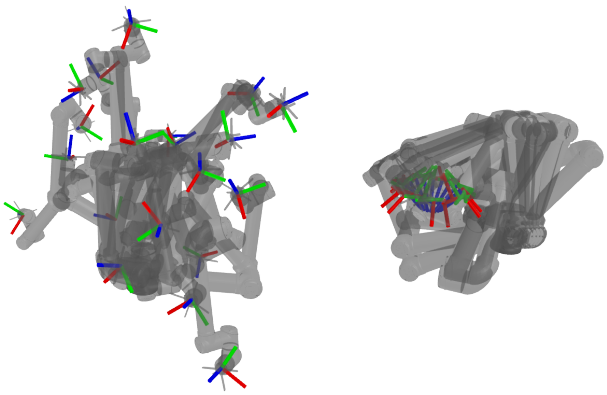
Most basic calibration procedures rely on randomly chosen configurations. It has been shown that an insightful choice of measurements configurations may improve both the robustness and accuracy of the calibration [6, 23, 24]. The aim is to pick a set of configurations maximizing a given metric, often related to the identifiability of the parameters, and computed using the identification Jacobian matrix. The search for such configurations can either be done continuously over the whole actuators angular range [23], or discretely, after a sampling of the configurations [24]. Despite being more restrictive, the discrete method provides a precious advantage for collisions avoidance, as physical interactions between serial robots and complex environments are not easily embedded into continuous optimization problems. On the other hand, each discretely sampled configuration can be filtered based on its reachability, thus avoiding eventual collisions.

Following this reasoning, we developed a simulation based sampling tool, detailed in Section III, taking as inputs both the robot and the environment collision properties, and returning a set of sampled and reachable configurations. This tool also takes as input an optional subspace restriction, in order to narrow down the sampling into an area of interest, or related to the projected task [25]. The obtained sampled configurations are then stored in a YAML configuration file, and made available for further use. A partial sampling example is pictured in Figure 2.

Once the configurations sampled, usual combinatorial methods such as genetic algorithm, simulated annealing, or greedy exchange algorithm [24] can be used to extract the optimal configurations. In our case, all cited methods and usual observable metrics were implemented using a generic and simple interface, thought for further additions and developments. The practical handling of the measurements will be further detailed in Section III.

C. Step 3: Kinematic parameters identification

As previously stated, the identification step aims to find the best parameters estimation regarding to gathered experimental data. In practice, this problem is solved by finding the vector minimizing the Euclidean distance between the $N_{\text{meas.}}$ measured end effector poses, and their modeled counterparts:



(a) Unrestricted sampling (b) Spatially restricted sampling

Fig. 2: Partial configuration sampling example using an UR10e serial robot

$$\pi^* = \arg\min_{\pi} \sum_{i=1}^{N_{\text{meas.}}} ||T(Q_i, \pi) - T_{\text{measured}}(Q_i)||^2 \quad . \tag{3}$$

Where $\{Q_i\}_{i=1,\dots,N_{\text{meas.}}}$ denote the $N_{\text{meas.}}$ robot configurations considered for the measurements.

However, such a full-pose formulation is often impractical, as it brings out homogeneity issues between linear and angular components, and requires the use of a measuring device able to provide both position and orientation readings. In order to avoid these issues, we opted for a partial-pose approach [23], where only the position of $N_{EE} \geq 3$ reference end effector points is measured, *i.e.*:

$$\pi^* = \arg\min_{\pi} \sum_{i=1}^{N_{\text{meas.}}} \sum_{j=1}^{N_{EE}} \left|\left| P^j(Q_i, \pi) - P^j_{\text{measured}}(Q_i) \right|\right|^2 \tag{4}$$

where $\left\{P^j\right\}_{j=1,\dots,N_{EE}}$ denote the position of each of the N_{EE} end effector points, in a common and fixed reference frame, often set by the measuring device.

In this context, Equation (1) and Equation (2) need to be rewritten for each end effector point, and truncated only to retain the position outputs. The final model is then obtained by the concatenation of the N_{EE} partial models and identification Jacobian matrices.

It should still be noted that, eventhough partial-pose measurements provide a clearer definition of the identification problem, and a higher number of equations for the same number of robot configurations, they conversely introduce an additionnal set of parameters (*i.e.* the position of each end effector point), and requires a larger number of measurements to ensure a certain level of accuracy [26].

The optimal parameters estimation is then obtained using SVD based methods [14], the gradient descent method [7] or the widespread Gauss-Newton [6, 12] and Levenberg-Marquardt [7, 15] algorithms. As for the previous step, most cited methods were implemented, with intended room for further improvements.

D. Step 4: Verification, correction and compensation

To round off the calibration procedure, the identified parameters estimates still have to be verified, and integrated to the use of the robot.

The verification step often relies on an additional round of measurements, using either random, or task specific configurations. In the same vein as the measurements design step, our implementation allows verification configurations to be generated in a reachability aware way, or manually specified by the user. Using the metrics introduced in Equation (3) and Equation (4), the quality of the estimation and the resulting accuracy can then be assessed.

Finally, the identified parameters are combined in a YAML configuration file and translated into an URDF file to ensure a smooth and facilitated use.

III. HANDLING OF HARDWARE-SOFTWARE INTERFACES

As precised in the introduction, most calibration software suffer from an incomplete, or rough integration of hardware interactions with the measuring device, the studied robot, and their surrounding environment. In this heterogeneous context, the Robotic Operating System (ROS) offers a promising alternative, providing support for low-level device control and message-passing between processes at a hardware abstraction level guaranteeing a high modularity. In this regard, this section will go through the tools and abstraction methods used to ensure the smoothness and versatility of the hardware-software interfaces (*c.f.*, Figure 3).

A. Robot interface - Simulation and control

The most crucial, yet challenging, interface applies to the calibrated robot, as the off-line programming of serial robots still mostly relies on proprietary and laboriously compatible software [2]. Yet, this burdening issue has been successfully tackled by Moveit [27], an open source and user-friendly robotics manipulation platform, allowing for a simplified integration of serial robots in ROS. This tool offers numerous useful features, out of which three major ones were blended in our calibration module:

- A well stocked and accessible catalog of fully integrated robots, including complete URDF descriptions, out of which our modeling routine can build identifiable geometric models;
- A collision aware simulation tool, providing *a priori* insight on the reachability of the sampled configurations;
- A high level motion planning and control API, allowing the development and execution of generic targets reaching routines, and therefore crucial for the completion of the measurements step.

It should be noted that to be fully compatible with Moveit, a robot must provide some specific interfaces, including a joint level controller interface, able to process angular commands inputs, and a monitoring service, providing information about the current robot state, *e.g.*, angular joint values, speed, etc. If both interfaces are necessary for Moveit to be fully operable, the output of the latter is all the more crucial for the measurements steps.

Thereby, provided that the studied robot is set up to work with ROS and Moveit, its hardware integration in our calibration package is ensured.

B. External environment interface - Collisions

As highlighted in the previous section, a specificity of our approach is a preliminary spatially constrained and collision aware sampling of the measurements robot configurations. These collisions can either be caused by the robot itself, when a link collides with another one, or by its external environment, when a link collides with surrounding objects. The first ones are directly handled by Moveit during the simulation step, but the second ones need an additional step of modeling and embedding of the robot environment. The modeling part is handled using simple geometrical primitives (*e.g.*, spheres, cylinders and cubes) stored in an user provided YAML configuration file, and embedded to Moveit simulated environment on start-up (*c.f.* Figure 4). As collision avoidance only require a coarse over-approximation of colliding objects, this methods only requires a fast and approximate assessment of the surroundings, and provides a simple yet efficient solution.

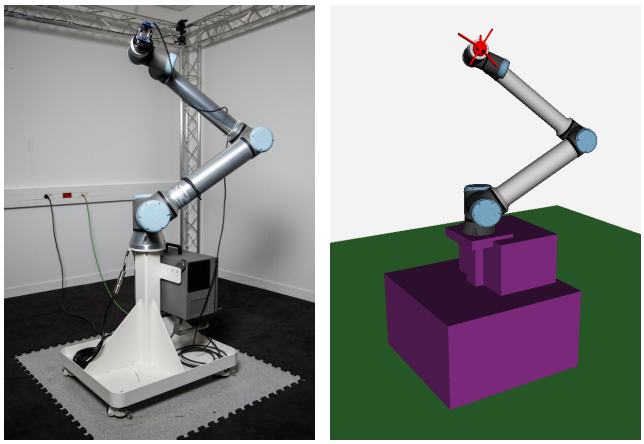
C. Measure device interface - Synchronisation and recording

Together with an easily controllable robot, a well interfaced measuring device is key to a smooth and error free calibration. This requirement has been met in our software through a virtual ROS service, allowing a properly synchronized communication between the robot, the main calibration procedure, and the measuring device. Concretely, when the robot reaches a targeted configuration, its state shift triggers a blocking call to the ROS service. On the server side, a completely hardware generic routine is executed and the measured data gathered into a dedicated CSV file, along with the monitored robot state. Once the measurement is either performed, or encountered a failure, the corresponding status is sent back, and the ability to start the next motion given back to the robot.

Concerning the hardware interface in itself, the only requirement is that the measured data is published on an accessible ROS topic, whose name is to be filled in to our package configuration parameters. It should also be noted that, although the measurement service currently only handles position measurements, any ROS compatible sensor can be readily added in a few lines of code. In this way, all measurements, regardless of their source, are performed at the same time and are fully synchronized with the robot motion, hence reducing the risk of errors.

IV. EXPERIMENTAL RESULTS

To assess the actual efficiency, ease of use and adaptability of the proposed solution, this section details its practical application to the calibration of two ROS-compatible serial robots: a 6-axis UR10e and a 7-axis Franka Emika Panda.



(a) Real work cell (b) Modeled work cell

Fig. 4: Modeling example of the collisions objects laying in a work cell

Both robots geometric models were directly derived from the URDF description provided by their ROS packages.

Calibration measurements were performed using both the robots controllers state monitoring outputs, and an *OptiTrack* motion tracking tool, equipped with 6 *Prime 13* cameras. Even though no information is provided concerning the actuators encoders, which are assumed to be perfectly accurate, the tracking system allows us to measure the absolute position of infrared reflective spherical marker with an accuracy of $\pm$ 0.2mm. In order to implement the partial-pose measurements approach, a custom structure holding 7 markers in a three dimensional and asymmetrical layout (*c.f.* Figure 5), was designed and thought to be fixed on the serial robots flange.

Each marker then describing an end effector point, their positions relative to the robot flange is added to the robots models, based on CAD data. It should be noted that, although only 3 end effector points are theoretically necessary, a higher number of markers brings redundancy in the measurements, and robustness to the parameters identification,

Fig. 5: The 7 markers measurement tool used on an UR10e

while keeping the number of identified parameters relatively low. In addition, a planar pattern of 7 markers was also created, and inserted between the robots bases and support to provide a rough estimation of the robot location in the *OptiTrack* reference frame. Although this estimation is not necessary [23], it remains a simple and inexpensive tool to remove the measuring tool reference frame dependency. Regarding the sampling step, the angular range of each robot actuator was evenly sampled by dividing it into 4 regular intervals, and retaining the 5 resulting endpoints. From there, 6^5 (UR10e) and 7^5 (F.E. Panda) angular configurations were constructed by combination and filtered against self and external collisions. Following the usual rule of thumb, twice as many measurement configurations as parameters to identify were randomly extracted on one hand, and wisely picked using the recommended O1-optimality index [24] on the other hand. This index is given by the determinant of the product of the identification Jacobian matrix by its transpose:

$$o_1 = \sqrt{\det\left(J_\pi^T J_\pi\right)} \ .$$

The discrete optimization was performed separately using the simulated annealing, and greedy exchange algorithms, with a limited number of iterations set to 150. Following the selected configurations, all positions measurements were carried out using the *mocap_optitrack* ROS package. For the geometric parameters identification step, the recorded

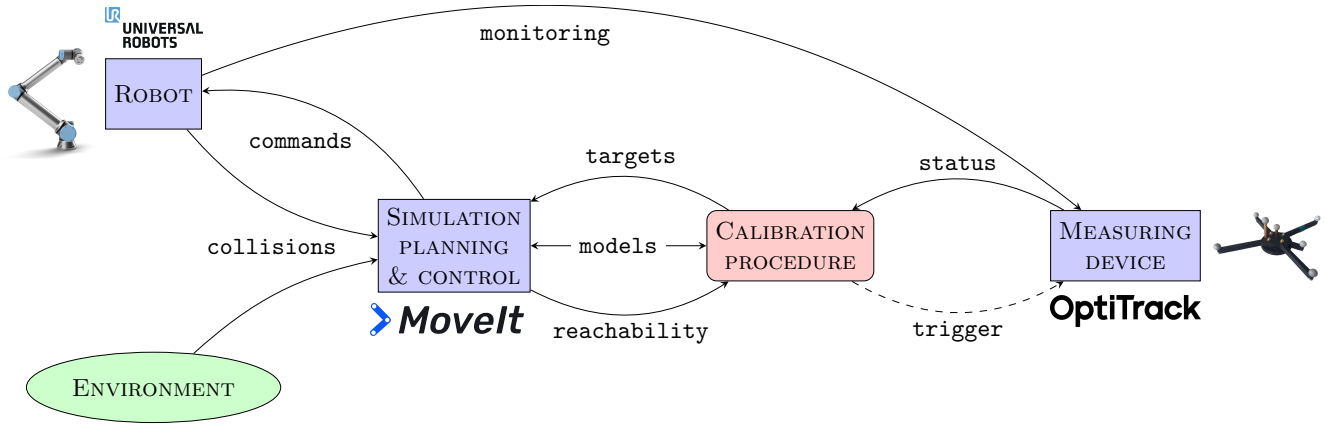


Fig. 3: Description of the hardware-software interfaces. Squares represent ROS nodes, and connecting arrows represent data exchanged over topics and services. Ellipse stands for a configuration file containing a description of surrounding obstacles.

data and previously computed identification Jacobian matrix were exploited using, disjointly, the gradient descent and Gauss-Newton algorithms, with a stopping criterion defined by a relative improvement lower than 10^{-8}. The finally obtained results, produced during the verification step, are summed up in Table I, along with the monitored duration of each calibration step. The identified parameters validity was assessed using 20 collision free configurations randomly picked over all actuators angular ranges. Details of the obtained accuracy outcomes are pictured on Figure 6 and Figure 7.

TABLE I: Summary of the main results obtained during the calibration of the studied robots

	UR10e	**F. E. Panda**
Number of axes	6	7
Total number of parameters	45	49
Number of robot specific parameters[1]	18	22
Overall calibration duration (hours)	7.03	4.38
Analytical modeling (min)	45	55
Measurements (hours)	6.0	3.1
Identification (min)	17	22
Initial average accuracy (mm)	14.4	7.66
Best final average accuracy (mm)	2.03	1.63
Improvement rate	85.9%	78.7%

[1] *e.g.* excluding the robot base transformation (6 parameters) and the measurement tool end effector points (21 parameters).

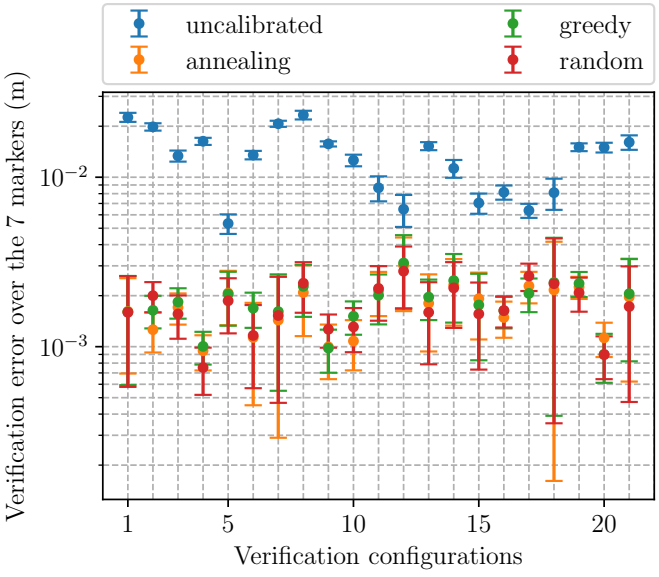


Fig. 6: Average positioning errors over the 7 markers obtained with and without calibration - UR10e (logarithmic vertical scale). *Results obtained with Gauss-Newton and gradient descent algorithms coincide.*

For both robots, our calibration procedure led to an increase in positioning accuracy of at least 75% over the 20 verification configurations, and 7 end effector points.

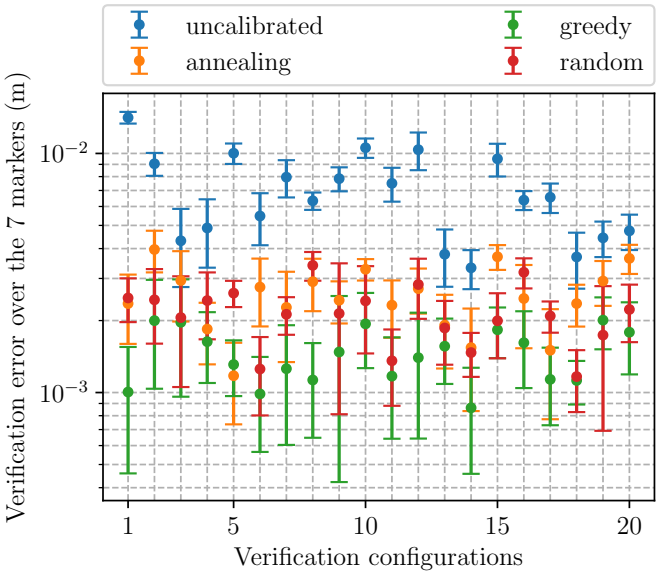


Fig. 7: Average positioning errors over the 7 markers obtained with and without calibration - Franka Emika Panda (logarithmic vertical scale). *Results obtained with Gauss-Newton and gradient descent algorithms coincide.*

However, the positioning errors dispersion on Figure 6 and Figure 7 show inconsistencies both between the verification configurations and end effector points, which bring to attention possible flaws in the motion tracking system readings (light variation, robot obstruction, etc.), and likely improvements in the choice of measurements configurations.

Comparing the two robots, the Franka Emika Panda broadly showed a lower increase in accuracy, presumably caused by a lack of consideration for the actuators flexibilities, which are a known feature of this flexible 7-axis cobot. On a side note, the positioning repeatability of these robots was not assessed in this study, as their expected values lie below the measuring device sensibility. This conclusion might also be extended to the UR10e as well, as the best results still remain above the minimal sensibility allowed by the measuring device.

Globally, no clear trend stands out concerning the impact of the identifiability index, or the combinatorial optimization algorithm. For instance, randomly picked measurements configurations for the Franka Emika Panda lead to a better accuracy than the ones selected by the simulated annealing algorithm. Concerning the parameters identification, both the gradient descent and the Gauss-Newton algorithm led to the same parameters estimations, but the latter broadly had a faster convergence pace, and is recommended for further use.

Thanks to its genericity and completeness, the proposed software allowed for both robot calibrations to run smoothly, with almost no interruptions, and in an autonomous way.

As highlighted in Table I, in both cases, the calibration results were obtained in a matter of hours, including all steps from the geometric modeling of the robot, to the recovery of the identified parameters. The major difference between the

two robots lies during the measurements step, and is caused by an extended reachability testing and motion duration for the UR10e. This robot is indeed more bulky (1.3m against 0.85m for the Franka Emika Panda), and less mobile (6 against 7 axes for the Franka Emika Panda), which makes the motion planning task and the motion itself more complicated.

V. FUTURE WORKS & CONCLUSIONS

Closing this paper, we presented a complete kinematic calibration procedure for serial robots, including their geometric modeling, the insightful choice of observations configurations upstream positioning measurements campaigns, and the identification of experience conditioned model parameters. This procedure was fully and smoothly integrated into an open access ROS package, whose main intent is to leave as much room as possible for further algorithmic improvements and testings. Benefiting from the middleware strength, this package provides generic and ergonomic hardware interfaces, allowing a simplified set up of any ROS compatible robot, or position sensor, while ensuring a time efficient implementation of the calibration itself. This package was successfully evaluated on two serial robots, and helped to increase their absolute positioning accuracy by 75% in a matter of hours.

Yet, in the continuation of the results presented in the previous section, several relevant leads should still be investigated to improve both the proposed procedure and package. Considering the calibration procedure itself, significant interest should be paid to the kinematic modeling of serial robots, which can significantly benefit from the embedding of more complex phenomena, such as flexibilities [3, 26] or thermal and mechanisms related effects [13]. On another aspect, one may focus on the optimality metric used to select the measurements configurations, as most of them rather focus on the identifiability of the kinematic parameters, rather than on the targeted accuracy itself [23]. On the practical application side, some refinement work might help improve the time complexity of the implemented steps. In particular, the analytical modeling step might be simplified into a numeric, yet approximate, equivalent counterpart, and the reachability testing highly accelerated using high-level parallelism. Regarding the hardware, additional work might be carried out to include full-pose measurements, and their inherent homogeneity issues.

ACKNOWLEDGMENT

Author warmly thanks Thibault Toralba and Clément Yver for their implication building tools needed for experiments.

REFERENCES

[1] S. N. Chiwande and S. S. Ohol, "Comparative need analysis of industrial robot calibration methodologies," in *Proc. IOP Conf. Ser.: Mater. Sci. Eng.*, vol. 1012, Jan. 2021, p. 012009.

[2] U. Schneider, M. Drust, M. Ansaloni, C. Lehmann, M. Pellicciari, F. Leali, J. W. Gunnink, and A. Verl, "Improving robotic machining accuracy through experimental error investigation and modular compensation," *Int. J. Adv. Manuf. Technol.*, vol. 85, no. 1, pp. 3–15, July 2016.

[3] M. H. To, "A framework for flexible integration in robotics and its applications for calibration and error compensation," Ph.D. dissertation, School of Eng., Cranfield Univ., June 2012.

[4] P. Margerit, T. Gobin, A. Lebée, and J.-F. Caron, "The robotized laser doppler vibrometer: On the use of an industrial robot arm to perform 3D full-field velocity measurements," *Opt. Lasers Eng.*, vol. 137, p. 106363, Feb. 2021.

[5] N. Knezevic, M. Bjelic, and K. Jovanovic, "Automated Sound Intensity Measurement With Robot And Intensity Probe," *Int. J. Elect. Eng. Comput.*, vol. 2, pp. 20–28, June 2018.

[6] J. Hollerbach, W. Khalil, and M. Gautier, "Model Identification," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Berlin, Heidelberg: Springer, 2008, pp. 321–344.

[7] L. Beyer and J. Wulfsberg, "Practical robot calibration with ROSY," *Robotica*, vol. 22, no. 5, pp. 505–512, Oct. 2004.

[8] "Calibration," in *Product manual IRB 1200*. ABB, Oct. 2022.

[9] "MotoCal V EG." [Online]. Available: https://www.automate.org/products/yaskawa-america/motocal-v-eg

[10] *Kinematic Calibration Manual for e-Series*. Universal Robots A/S, 2021.

[11] "Simulator for industrial robots and offline programming - RoboDK." [Online]. Available: https://robodk.com/

[12] W. Khalil and P. Lemoine, "A software package for the calibration of robots," in *Proc. CESA'96 IMACS*, July 1996, p. 131.

[13] M. Vocetka, Z. Bobovský, J. Babjak, J. Suder, S. Grushko, J. Mlotek, V. Krys, and M. Hagara, "Influence of Drift on Robot Repeatability and Its Compensation," *Appl. Sci.*, vol. 11, no. 22, p. 10813, Jan. 2021.

[14] H. Stone and A. Sanderson, "A prototype arm signature identification system," in *Proc. IEEE Int. Conf. Robot. Automat.*, vol. 4, Mar. 1987, pp. 175–182.

[15] L. S. Ginani and J. M. S. T. Motta, "Theoretical and practical aspects of robot calibration with experimental verification," *J. Bras. Soc. Mech. Sci. Eng.*, vol. 33, pp. 15–21, Mar. 2011.

[16] M. Quigley, "ROS: an open-source Robot Operating System," in *Proc. IEEE Int. Conf. Robot. Automat.*, Jan. 2009.

[17] J. Meyer, "robot_cal_tools." [Online]. Available: https://github.com/Jmeyer1292/robot_cal_tools

[18] M. Ferguson, "robot_calibration." [Online]. Available: https://github.com/mikeferguson/robot_calibration

[19] S. Kana, J. Gurnani, V. Ramanathan, S. H. Turlapati, M. Z. Ariffin, and D. Campolo, "Fast Kinematic Re-Calibration for Industrial Robot Arms," *Sensors J.*, vol. 22, no. 6, p. 2295, Mar. 2022.

[20] W. Khalil and E. Dombre, "Chapter 11 - Geometric calibration of robots," in *Modeling, Identification and Control of Robots*. Oxford: Butterworth-Heinemann, Jan. 2002, pp. 257–289.

[21] K. Schröer, S. L. Albright, and M. Grethlein, "Complete, minimal and model-continuous kinematic models for robot calibration," *Robot. Comput.-Integr. Manuf.*, vol. 13, no. 1, pp. 73–85, Mar. 1997.

[22] A. Pashkevich, "Computer-aided generation of complete irreducible models for robotic manipulators," in *Proc. 3rd Int. Conf. Model. Simul.*, Jan. 2001, pp. 293–298.

[23] Y. Wu, A. Klimchik, S. Caro, B. Furet, and A. Pashkevich, "Geometric calibration of industrial robots using enhanced partial pose measurements and design of experiments," *Robot. Comput.-Integr. Manuf.*, vol. 35, pp. 151–168, Oct. 2015.

[24] D. Daney, Y. Papegay, and B. Madeline, "Choosing Measurement Poses for Robot Calibration with the Local Convergence Method and Tabu Search," *Int. J. Robot. Res.*, vol. 24, no. 6, pp. 501–518, June 2005.

[25] L. Lattanzi, C. Cristalli, D. Massa, S. Boria, P. Lépine, and M. Pellicciari, "Geometrical calibration of a 6-axis robotic arm for high accuracy manufacturing task," *Int. J. Adv. Manuf. Technol.*, vol. 111, no. 7, pp. 1813–1829, Dec. 2020.

[26] A. Klimchik, Y. Wu, S. Caro, B. Furet, and A. Pashkevich, "Geometric and elastostatic calibration of robotic manipulator using partial pose measurements," *Adv. Robot.*, vol. 28, Oct. 2014.

[27] D. T. Coleman, I. A. Sucan, S. Chitta, and N. Correll, "Reducing the Barrier to Entry of Complex Robotic Software: a MoveIt! Case Study," *J. Softw. Eng. Robot.*, Jan. 2014.