High-Velocity Walk-Through Programming for Industrial Applications: A Safety-Oriented Approach

Simone di Napoli[1], Mattia Bertuletti[1], Mattia Gambazza[1],
Matteo Ragaglia[1], Cesare Fantuzzi[2] and Federica Ferraguti[2]

[1]*Gaiotto Automation (SACMI Group), Via Toscana 1, Piacenza PC 29122, Italy*

[2]*DISMI, University of Modena and Reggio Emilia, Department of Sciences and Methods for Engineering, Via Amendola 2, Pad. Morselli - 42122 Reggio Emilia, Italy*

Keywords: Physical Human-Robot Interaction, Cooperative Robotics, Admittance Control, Walk-Through Programming.

Abstract: Traditionally, industrial robots are programmed by highly specialized workers that either directly write code in platform-specific languages, or use dedicated hardware (teach-pendant) to move the robot through the desired via-points. In the last years, new strategies to manually move the robot through the human input had been introduced. During the human-robot interaction, the most limitation of this kind of use of the robot is the velocity reduction of the machine. Taken into account the introduction of sensor-based walk-through programming approaches as the ideal solution to reduce programming complexity and time, this paper proposes a safety architecture for walk-through programming of industrial manipulators specifically designed in order to reach high velocities while guaranteeing the operator's safety. The proposed solution is validated on an industrial manipulator.

1 INTRODUCTION

In the recent years the paradigm for robot usage has radically changed, moving from the idea of complete workspace segregation (achieved through physical barriers) to a scenario in which robots and humans share the same workspace and even collaborate side by side (Michalos et al., 2021). In this context, robots are becoming key elements in increasing productivity, since continuous and fruitful physical human-robot interaction (pHRI) can definitely help to achieve the higher production flexibility in order to cope with limited volumes and rapidly changing product requirements. Nevertheless, widespread adoption of robotic technologies is still undermined by certain well-known factors, among which the inherently complex and time-consuming nature of robot programming surely plays a crucial role (Pan et al., 2012).

In order to supply productivity and flexibility advantages in pHRI, robot programming strategies usually defined as "walk-through programming" (also referenced as "lead-through programming", or "manual guidance") have been proposed (Ang et al., 2000), with practical applications ranging from spraying (Ferretti et al., 2009) to welding (Ang et al., 1999). "walk-through programming" strategies are characterized by the fact that the human operator manually moves the robot directly interacting with the robot end-effector. Pararelly, robot records its trajectory and, after the teaching phase, this can be reproduced.

From the control point of view, the first applications of this programming paradigm were completely passive and relied on backdrivable actuators. Mechanical compensation was typically provided, by means of either hydraulic or pneumatic cylinders, in order to help the operator lift and move the robot. An example of this technology is represented by the Gaiotto GA-2000 manipulator (Gaiotto Automation Spa,).

With the introduction of "manual guidance" strategies, an even greater relevance is attributed to robot safety standards, which have been updated to address co-working scenarios as walk-through programming. In particular, industrial settings need to comply with strict safety requirements, given by the standards ISO 10218-1 (ISO, 2011a) and ISO 10218-2 (ISO, 2011b) and the most recent technical specification ISO/TS 15066 (cit, 2015). The ISO/TS 15066 provides the Hand-guiding interaction method where the human could exploit the manually movements to teach a machine on how to correctly complete his

di Napoli, S., Bertuletti, M., Gambazza, M., Ragaglia, M., Fantuzzi, C. and Ferraguti, F.
High-Velocity Walk-Through Programming for Industrial Applications: A Safety-Oriented Approach.
DOI: 10.5220/0012250900003543
In *Proceedings of the 20th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2023) - Volume 1*, pages 503-510
ISBN: 978-989-758-670-5; ISSN: 2184-2809

work. Unfortunatly, the standards severely limits the robot performances during the human interaction; the velocity is controlled at an appropriately reduced speed. Indeed, for many industrial application, even during the teaching phase, the high velocity of the process is important to see a real result. Therefore, for these cases, the hand-guiding method should be too liming.

1.1 Related Works

Fro a purely functional point of view, walk-through programming architectures rely on two key elements: a sensor system and an admittance (or impedance) control algorithm (Villani and De Schutter, 2008). The sensor system is responsible for measuring the interaction forces/torques exerted by the human operator on the manipulator, while the control algorithm ensures that the robot responds to the operator's input accordingly.

Assuming that an architecture like this is available, another critical issue that still needs to be addressed is the unavoidable trade-off between safety and performance. In particular, (cit, 2015) specifies restrictions regarding the transferred energy, the biomechanical limits and the robot velocity allowed during transient and quasi-static contact. Among these requirements, the most relevant one with respect to the presented implementation of walk-through programming applied to industrial robots that need to execute continuous trajectories at very high velocities is the Cartesian velocity limit of 250 mm/s.

Indeed, though this limitation may be suitable for scenarios in which only via-points need to be memorized (Ragaglia et al., 2016), it may prevent walk-through programming to be used when continuous trajectories performed at high-velocity need to be recorded. For instance, spraying robots cannot be manually guided at low Cartesian velocities, since their motion needs to be synchronized with the spraying system set-points that cannot be kinematically scaled with respect to time.

In principle, solutions based on tele-operation (Tafazoli et al., 2002) (Tanzini et al., 2016) could be considered as a safer alternative to walk-through programming, since they do not require pHRI. Nevertheless, the lack of direct interaction needs to be compensated by adding tele-presence features such as haptic feedback (Jacinto-Villegas et al., 2017) and remote vision (Tripicchio et al., 2017), thus leading to significant equipment cost increases that can be justified only when working in highly dangerous contexts such as building demolition, decommissioning of nuclear power plants, disaster recovery, etc.

1.2 Contribution Statement

As already pointed out, safety regulations establish requirements that may prevent walk-through programming to be used when continuous trajectories performed at high-velocities need to be recorded. However, in industrial applications such as spraying, the robot programming has to be recorded at high-velocities without any kinematic scaling. Consequently, a control architecture which satisfies the safety regulations but allows the execution of trajectories at high-velocities needs to be developed.

To this regard, this work introduces a novel safety control architecture for walk-through programming that combines traditional safety checks with advanced monitoring functionalities of both the robot and the human operator in order to ensure their safety when recording high-velocity continuous trajectories. For the sake of completeness, a more detailed explanation of the proposed solution is given in (Ferraguti et al., 2023).

2 SAFETY LOGIC DESIGN

Walk-through programming is without a doubt one of the clearest examples of physical human-robot interaction, it consists of two different phases:

- **Teaching Phase:** the human operator physically guides the robot end-effector to teach the trajectory to be executed, while the robot controller records all the significant poses of the trajectory itself;
- **Execution Phase:** the robot plays the continuous trajectory back.

To this regard, this work focuses on the development of a strategy that enables human operators to record continuous trajectories performed at high velocity both at the joint and at the Cartesian space level. Consequently, the fact that mechanical hazards are the most significant ones to be taken into account from the safety point of view comes at no surprise. More specifically, among the various potential consequences of the several mechanical hazards listed by (ISO, 2011a), we took into consideration two main scenarios: human-robot impact, and entanglement/trapping of a generic operator's body part.

In order to obtain an effective trade-off between safety and productivity, in this paper we propose to design a control architecture that tackles safety issues by following a two-fold approach:

- **Basic Safety Functions:** upper bounds on both joint and Cartesian space dynamic quantities are established and the safety controller checks if these bounds are violated during the teaching phase;

- **Dynamic Human-Robot Monitoring:** the safety controller monitors in real-time the relative distance and the relative velocity of the robot with respect to the human operator. Threshold values are established in order to identify situations characterized by a high risk of impact and/or entanglement.

In both cases the safety controller checks a logic condition. If both conditions are true, the programming phase can proceed, while if at least one condition is evaluated as false, an emergency stop is issued by the safety controller and the manipulator is stopped.

2.1 Basic Safety Functions

As already mentioned, basic safety functions can be implemented in order to force an emergency stop whenever a specific dynamic quantity (at both joint or Cartesian space level) exceeds a prescribed threshold value. In this work three different basic safety functions have been considered, each one consisting in checking a specific dynamic quantity against a maximum allowed positive value, defined as follows:

- maximum allowed Cartesian Linear Speed at TCP $\left\|\dot{x}_p^{UB}\right\| \in \mathbb{R}$, where x_p and $\dot{x}_p$ correspond to the positional portion of x and $\dot{x}$, respectively;
- maximum allowed Joint Velocities $\dot{q}^{UB} \in \mathbb{R}^m$;
- maximum allowed Joint Accelerations $\ddot{q}^{UB} \in \mathbb{R}^m$.

As far as velocities are concerned, the TCP linear speed threshold depends on the specific task and it corresponds to the maximum speed needed to correctly record the continuous trajectory that will be directly converted into a robot program.

Moving to joint velocity bounds, suitable threshold values can be established by taking into account a reference configuration of the manipulator $\overline{q} \in \mathbb{R}^m$ and by computing, for each joint, the angular velocity $\dot{q}_k^{UB}$, with $k = 1, \ldots, m$, that would result in the maximum allowed TCP linear speed $\dot{x}_p^{UB}$ (which is always positive) via multiplication by the geometric Jacobian matrix $J(\cdot)$.

$$\dot{q}_k^{UB} \leftarrow \min\left(\dot{q}_k^{Max}, \frac{\dot{x}_p^{UB}}{\sqrt{\sum_{w=1}^{3}\left(J\left(\overline{q}\right)\right)_{w,k}^2}}\right) \tag{1}$$

Saturation with respect to absolute joint velocity limits ($\dot{q}_k^{Max}$) must be taken into account to guarantee that the safety check is effectively enforced. In this way, this safety function acts as a redundant check with respect to TCP speed monitoring.

Finally, once joint velocity bounds have been defined, the corresponding accelerations bounds can be computed on the basis of the robot stopping time Δt. Once again, saturation with respect to absolute joint acceleration limits ($\ddot{q}_k^{Max}$) must be taken into account to guarantee that the safety check is effectively enforced.

$$\ddot{q}_k^{UB} \leftarrow \min\left(\ddot{q}_k^{Max}, \frac{\dot{q}_k^{UB}}{\Delta t}\right) \tag{2}$$

For the sake of completeness, joint velocities and accelerations are checked against the discussed maximum values by means of their absolute value, so that either positive and negative values are properly checked. Consequently, the comprehensive logic condition computed by the basic safety functions can be expressed as follows:

$$\begin{aligned} \|\dot{x}_p\| &\leq \left\|\dot{x}_p^{UB}\right\| && \wedge \\ |\dot{q}_k| &\leq \dot{q}_k^{UB} \quad \forall\, k = 1, \ldots, m && \wedge \\ |\ddot{q}_k| &\leq \ddot{q}_k^{UB} \quad \forall\, k = 1, \ldots, m && \end{aligned} \tag{3}$$

2.2 Dynamic Human-Robot Monitoring

As introduced before, in addition to the basic safety functions, the proposed safety controller is endowed with the capability to monitor in real-time the relative distance and the relative velocity of the robot with respect to the human operator. This way, the controller can identify situations characterized by a high risk of impact/clamping and issue an emergency stop accordingly.

In order to perform these functionalities, a strategy to model the space occupancy of both the operator and the robot is needed. To this regard, previous contributions in the field of safe human-robot interaction (Ragaglia et al., 2015) have already explored the application of capsule-based geometry models to the problem of modeling the space occupancy of complex objects, proving that this approach can be extremely efficient in approximating structured geometries. For the sake of clarity, a capsule consists in the convex hull of a sphere of given radius (also named "clearance"), which is translated along a segment. For instance, Fig. 1 shows two distinct capsules. The first one on the left is defined by points $P_{1,0}$, $P_{1,1}$ and radius r_1, while the second one on the right is defined by $P_{2,0}$, $P_{2,1}$ and radius r_2. As exemplified in the figure, the minimum distance between the two capsules d can be obtained by computing the minimum distance d_1

between the segments $\overline{P_{1,0}\,P_{1,1}}$ and $\overline{P_{2,0}\,P_{2,1}}$ (as thoroughly explained in (Ericson, 2005)), and then subtracting the two clearances r_1 and r_2. Clearly, whenever $d \leq 0$, the capsules are colliding.

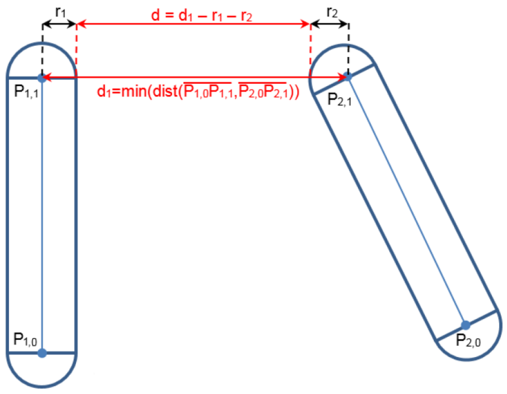


Figure 1: Geometry representation of two separate capsules and of the minimum distance between them (Ferraguti et al., 2023).

In this work, capsules are used to model the space occupancy of the robot and of the human operator, as it is shown in Fig. 2. As far as the robot is concerned, a separate capsule is defined for each link and clearances are assigned on the basis of the links' geometry. Moving to the human operator, their space occupancy can be modeled by means of a single vertically oriented capsule, whose radius and height can be parameterized according to the anthropometric features of the specific operator. In addition, the operator's capsule is rigidly attached to the robot's end-effector according to the geometry of the handle that allows to move the robot.

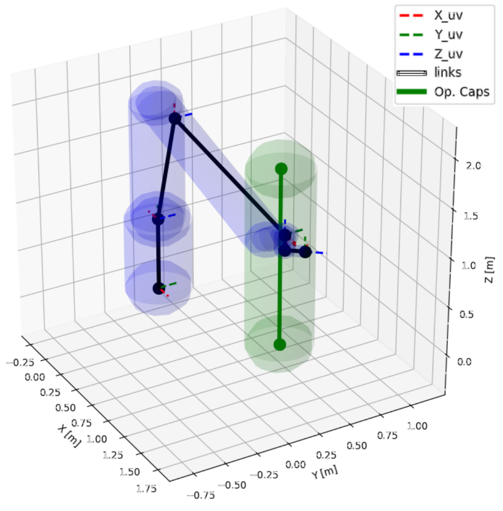


Figure 2: Example of operator's capsule and capsule-based geometry model of a 6 DoF offset wrist manipulator.

In principle, a more complex model could be taken into account for the human operator, by considering all the pairs composed by a robot link and a human link that can be computed dynamically as proposed in (Ferraguti et al., 2020). To this aim a tracking system that allows to detect and monitor the current position of each link of the human operator is required to be integrated, by following the strategies proposed in (Ragaglia et al., 2018). However, the main reason behind our choice lies in the fact that we are interested in preventing collisions by ensuring that there is a sufficient separation distance between the robot and the operator, rather than precisely identifying the operator's body part involved in a collision. In addition, given the nature of the task, some body parts (i.e. hands, wrists, and possibly forearms) would always result in collision (or at least in close proximity) with the robot, thus making this level of detail not necessary.

During the execution of the teaching phase, the safety controller updates in real-time the position of all the capsules, computes the relative distance between each robot capsule and the operator's one (like it is shown in Fig. 1), and checks that the minimum relative distance remains above a parameter threshold. By setting this threshold equal to the distance that the robot can cover at the maximum allowed linear speed $\dot{x}_p^{UB}$ during the time needed to enforce an emergency stop, the safety controller is able to prevent collisions between the robot and the operator.

Having covered the strategy that allows to estimate the relative distance between the robot and the operator, let us focus on the monitoring of the relative velocity between them. The usage of relative human-robot velocity in combination with relative human-robot distance for safety purposes has been proposed as a safety metric by several contributions in the field of safe human-robot interaction, like for instance (Ragaglia et al., 2014). In this work, relative human-robot velocity is monitored in order to identify situations where at least a single robot link is moving towards the operator with enough speed, that an emergency stop may not be able to prevent a collision. To this purpose, a reasonable choice for the threshold value is represented by the maximum allowed linear speed $\dot{x}_p^{UB}$.

Once again, the capsule-based geometry models of both the manipulator and of the operator represent the starting point. More specifically, the relative velocity between the robot and the operator is defined as the maximum relative velocity between the robot capsules' end-points and the operator's capsule.

Let us consider Fig. 3, where the solid grey bar represents the i-th robot link, surrounded by the corresponding capsule $robCap_i$. On the other hand, the operator capsule $opCap$ is pictured in the same way around its axis, passing through $opCap.P$. For both capsules clearances $robCap_i.r$ and $opCap.r$ are also highlighted. By means of kinematic calculations, the linear velocities vel_{P_0} and vel_{P_1} can be easily computed, given both joint angles and joint velocities. Then, by projecting velocity vel_{P_0} (vel_{P_1}) along the direction that connects point $robCap_i.P_0$ ($robCap_i.P_1$) to the axis of the operator capsule, we can compute the relative velocity of that specific point with respect to the operator. It is worth mentioning that the result of this projection is a signed velocity value, that is positive if the link end-point is moving towards the operator, and negative when it is moving away from the operator.

It is also worth mentioning that, as stated in (Ragaglia et al., 2015), linear velocity varies linearly between the link end-points. As a result, either point $robCap_i.P_0$ or $robCap_i.P_1$ is necessarily the link point characterized by the maximum linear velocity. Since the projection with respect to the operator's capsule consists in a linear combination of the linear velocity coordinates, once again we can state that either point $robCap_i.P_0$ or $robCap_i.P_1$ is necessarily the link point characterized by the maximum relative velocity with respect to the operator, thus making it not necessary to check intermediate points.

Finally, please note that point $robCap_i.P_1$ of the i-th link corresponds to point $robCap_i.P_0$ of the $(i+1)$-th link. In addition, since the robot base (which typically corresponds to point $robCap_i.P_0$ of the first robot link) is normally still, the maximum relative velocity can be obtained by projecting the linear velocities of each end-point $robCap_i.P_1$ of the robot links.

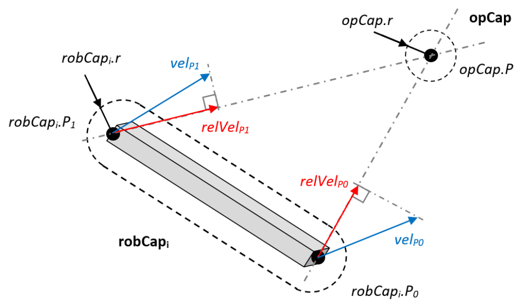


Figure 3: Geometric representation of the relative velocity between the end-points of a generic robot link (solid grey bar) and the operator's capsule.

As a result, the comprehensive logic condition checked by the dynamic human-robot monitoring block can be expressed as follows:

$$relDistOk = 1 \ \wedge \ relVelOk = 1 \tag{4}$$

3 IMPLEMENTATION

In order to validate the proposed control architecture, we implemented the high-speed walk-through programming strategy described in the previous sections in a practical industrial use case. In particular, the strategy has been implemented on the Gaiotto GA-OL manipulator shown in Fig. 4, which is a 6 DoF industrial robot designed for spraying applications (Gaiotto Automation Spa,). Typically, the GA-OL robot is endowed with an auxiliary turning table, on top of which the object to be sprayed is loaded and then rotated in order to help the robot execute the spraying programs.

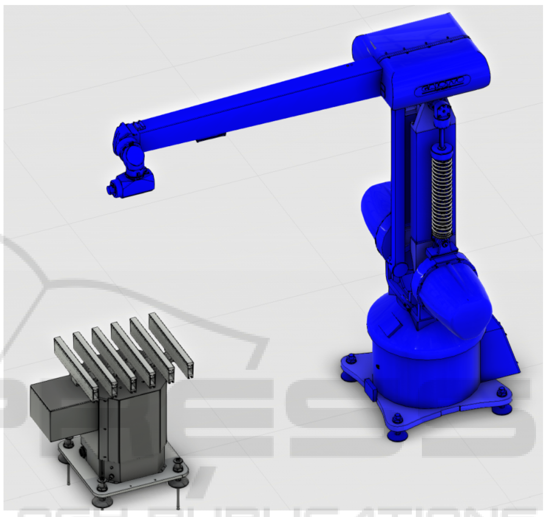


Figure 4: Gaiotto GA-OL manipulator (on the right, pictured in blue) and auxiliary turning table (on the left, pictured in grey) (Ferraguti et al., 2023).

The implementation of the safety logic lies between the safety and functional environments, as depicted in Fig. 5. More specifically, the basic safety functions (see subsection 2.1) are implemented within the "Safety Logic" domain. On the other hand, the dynamic human-robot monitoring functionalities (see subsection 2.2) are executed within the "Functional Logic" domain, which is further divided into two distinct environments: the "C++ Environment" and the "IEC Environment". The first one comprises a series of libraries written in C/C++ that implement the functional counterpart of the safety controller proposed in this paper, while the latter hosts components written in either Structured Text or FBD, like for instance the aforementioned dynamic human-robot monitoring functionalities. The communication between the functional domain and the safety domain is realized by means of "virtual digital IOs", i.e. boolean variables that are exchanged between the two domains via shared memory.

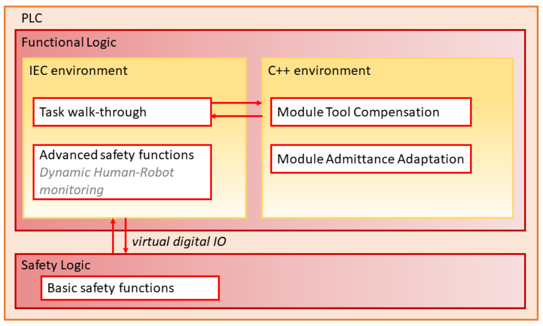


Figure 5: General control architecture (Ferraguti et al., 2023).

The Safety Logic collects all the information needed to compute the Safe Torque Off (STO) signal, that cuts-off the electric power to the axis motors (thus preventing them to develop torque) whenever it is de-activated by the Safety Logic. As far as parameters are concerned, the following values have been used:

- $||\dot{x}_p^{UB}|| = 1.30\ m/s$ - maximum linear speed at TCP, due to spraying task requirements[1];
- $\dot{q}^{Max} = [200, 200, 200, 300, 300, 300]\ deg/s$ - maximum GA-OL joint velocities;
- $\ddot{q}^{Max} = [450, 450, 450, 450, 450, 450]\ deg/s^2$ - maximum GA-OL joint accelerations;
- $robRad = [0.25, 0.20, 0.10, 0.075, 0.05, 0.05, 0.075]\ m$ - robot and tool capsules radii;
- $opRelP = [0.000, 0.000, 0.250]\ m$ - operator capsule position with respect to GA-OL end-effector. Relative orientation is considered null;
- $opRad = 0.200\ m$ - operator capsule radius;
- $opH = 1.80\ m$ - operator capsule height;
- $stopT = 0.35\ s$ - robot stopping time;
- $minDistTh = 0.455\ m$ - minimum allowed distance between operator and robot capsules. It is obtained by multiplying $stopT$ by $\dot{x}_p^{UB}$;
- $maxVelTh = 1.30\ m/s$ - maximum allowed relative velocity between operator and robot capsules. Given $stopT$ and $minDistTh$, it can be set equal to $\dot{x}_p^{UB}$.

The operator capsule radius and height have been empirically selected as the medium value between the real characteristics of the testers involved in the experimental campaign.

[1]The value of the velocity limit has been selected as the maximum value of the TCP reached by the robot in 40 different programs executed on a GA-2000 manipulator, which is equipped with mechanical compensation systems for passive walk-through programming.

4 EXPERIMENTAL RESULTS

An extensive experimental evaluation phase has been performed in order to demonstrate the effectiveness of the proposed architecture, whose results are hereby presented and discussed. For the sake of completeness, experiments involved several testers selected among Gaiotto employees.

First, let us show how the violation of the conditions defined by the basic safety functions results in disabling the STO and stopping the robot. Figure 6(a) shows that as soon as the TCP linear velocity limit ($\dot{x}_p^{UB} = 1.30 m/s$) is exceeded, the STO signal (properly scaled) is disabled and the robot stops. In addition, the magnification shown in Fig. 6(b) also proves that the robot completely stops within the declared stopping time equal to $0.35\,s$.

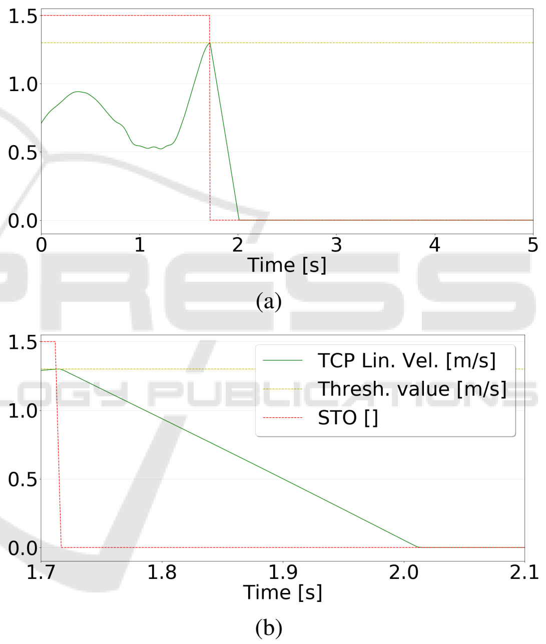
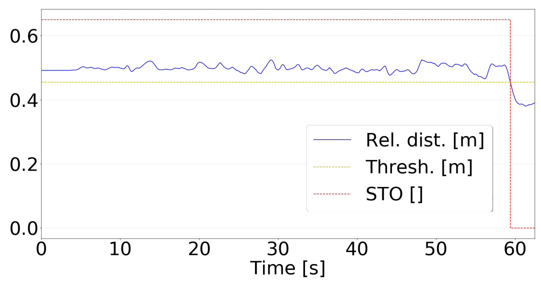
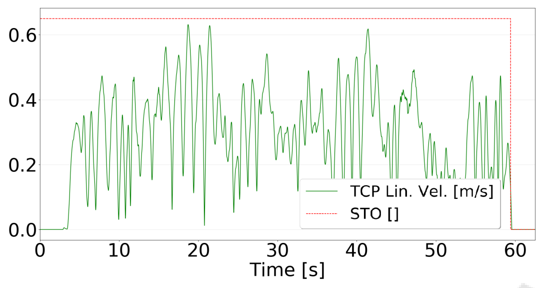


Figure 6: Violation of TCP linear velocity limit and corresponding variation of scaled STO signal, with magnification between $1.70\,s$ and $2.10\,s$.

On the other hand, Figures 7 and 8 shows violations related to the dynamic human-robot monitoring algorithms. Figure 7(a) shows how the violation of the minimum relative distance condition triggers the disabling of the STO by the safety controller, while in Fig. 7(b) TCP linear velocity is depicted. The very same behaviour characterizes the safety controller response to a relative velocity violation, as it is shown in Fig. 8.

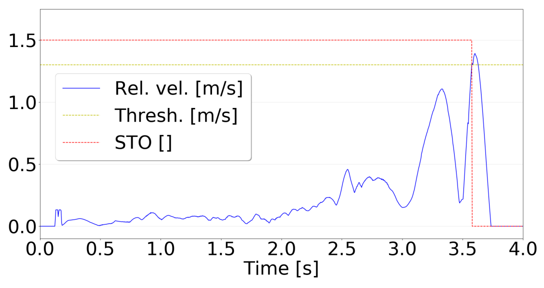


(a)

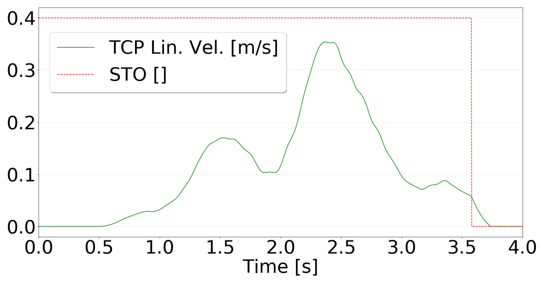


(b)

Figure 7: Violation of the relative human-robot minimum distance with TCP linear velocity and corresponding variation of scaled STO signal, with magnification between 59.25 s and 59.75 s.



(a)



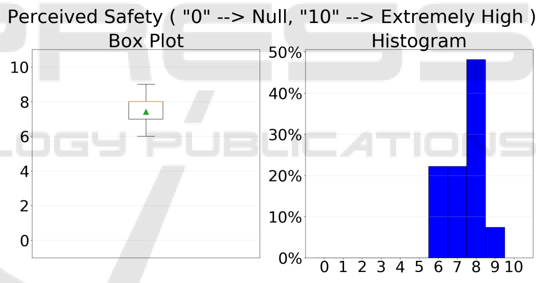
(b)

Figure 8: Violation of the relative human-robot maximum velocity with TCP linear velocity and corresponding variation of scaled STO signal, with magnification between 3.50 s and 3.80 s.

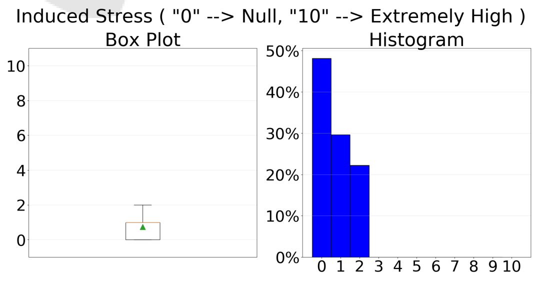
4.1 Usability Evaluation

In order to properly evaluate the usability of the proposed walk-through programming architecture in actual industrial scenarios, the selected testers have been asked to answer a questionnaire right after the tests. More in detail, testers were asked to rate several indicators, by assigning each one a score ranging from "0" (meaning either "null" or "extremely negative"), to "10" (meaning either "extremely high" or "extremely positive").

Figure 9 shows the results corresponding to two distinct indicators: perceived level of safety, and induced stress. More specifically, a very high overall score has emerged with respect to perceived safety 9(a). Please also notice that the low score regarding to induced stress 9(b) corresponds to a positive evaluation by the testers. Finally, we can state that the proposed architecture represents a promising solution not only from the theoretical point of view, but also from the application perspective.



(a)



(b)

Figure 9: Evaluation of general indicators. For each indicator, the box-whiskers plot on the left shows the approximated distribution of the rates assigned by the testers, by highlighting the median (solid orange line) and the mean (green triangle). Then, the histogram on the right displays the normalized frequency of the assigned rates.

5 CONCLUSIONS & FUTURE DEVELOPMENTS

In this paper, the authors propose an innovative control architecture for walk-through programming of an industrial manipulator. The proposed solution aims at allowing human operators to program industrial manipulators by directly teaching high-speed trajectories, while guaranteeing the operator's safety. A dedicated safety controller has been developed to monitor the kinematic configuration of the manipulator and stop its motion whenever the risk of a collision with the human operator becomes too high.

Given the results of the experimental validation, the authors can state that the proposed architecture successfully achieves a fruitful trade-off between safety and productivity, by guaranteeing the operators' safety and the possibility to directly record high-velocity trajectories.

As far as future developments are concerned, the authors foresee to integrate a tracking system to detect and monitor in real-time the current position of each link of the human operator, in order to obtain very accurate occupancy volumes.

ACKNOWLEDGMENT

This work is partly supported by Regione Emilia-Romagna under the agreement "Innovazione della value proposition di Gaiotto Automation 2021-2025" (grant number / CUP: E32C21001090009).

REFERENCES

(2015). Iso/ts 15066 - robots and robotic devices – collaborative robots.

Ang, M. H., Lin, W., and Lim, S. (1999). A walk-through programmed robot for welding in shipyards. *Industrial Robot: An International Journal*, 26(5):377–388.

Ang, M. H., Wei, L., and Lim Ser Yong (2000). An industrial application of control of dynamic behavior of robots-a walk-through programmed welding robot. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 3, pages 2352–2357 vol.3.

Ericson, C. (2005). *Real-Time Collision Detection*. Elsevier.

Ferraguti, F., Bertuletti, M., Gambazza, M., and Ragaglia, M. (2023). High-velocity walk-through programming for industrial applications. *Robotics and Computer-Integrated Manufacturing*, 81:102505.

Ferraguti, F., Talignani Landi, C., Costi, S., Bonfè, M., Farsoni, S., Secchi, C., and Fantuzzi, C. (2020). Safety barrier functions and multi-camera tracking for human-robot shared environment. *Robotics and Autonomous Systems*, 124:103388.

Ferretti, G., Magnani, G., and Rocco, P. (2009). Assigning virtual tool dynamics to an industrial robot through an admittance controller. In *International Conference on Advanced Robotics (ICAR)*, pages 1–6.

Gaiotto Automation Spa. Gaiotto website. https://www.gaiotto.com/. Last checked on Sept 06, 2023.

ISO (2011a). Iso-2011-10218-1 - robots and robotic devices - safety requirements for industrial robots - part 1: Robots.

ISO (2011b). Iso-2011-10218-2 - robots and robotic devices - safety requirements for industrial robots - part 2: Robot systems and integration.

Jacinto-Villegas, J. M., Satler, M., Filippeschi, A., Bergamasco, M., Ragaglia, M., Argiolas, A., Niccolini, M., and Avizzano, C. A. (2017). A novel wearable haptic controller for teleoperating robotic platforms. *IEEE Robotics and Automation Letters (IEEERAL)*, 2(4):2072–2079.

Michalos, G., Karagiannis, P., Dimitropulos, N., Andronas, D., and Makris, S. (2021). *Human Robot Collaboration in Industrial Environments*.

Pan, Z., Polden, J., Larkin, N., Van Duin, S., and Norrish, J. (2012). Recent progress on programming methods for industrial robots. *Robotics and Computer-Integrated Manufacturing*, 28(2):87–94.

Ragaglia, M., Bascetta, L., Rocco, P., and Zanchettin, A. M. (2014). Integration of perception, control and injury knowledge for safe human-robot interaction. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1196–1202.

Ragaglia, M., Zanchettin, A. M., Bascetta, L., and Rocco, P. (2016). Accurate sensorless lead-through programming for lightweight robots in structured environments. *Robotics and Computer-Integrated Manufacturing (RCIM)*, 39:9 – 21.

Ragaglia, M., Zanchettin, A. M., and Rocco, P. (2015). Safety-aware trajectory scaling for human-robot collaboration with prediction of human occupancy. In *International Conference on Advanced Robotics (ICAR)*, pages 85–90.

Ragaglia, M., Zanchettin, A. M., and Rocco, P. (2018). Trajectory generation algorithm for safe human-robot collaboration based on multiple depth sensor measurements. *Mechatronics (MECH)*, 55:267–281.

Tafazoli, S., Salcudean, S. E., Hashtrudi-Zaad, K., and Lawrence, P. D. (2002). Impedance control of a teleoperated excavator. *IEEE Transactions on Control Systems Technology (IEEECST)*, 10(3):355–367.

Tanzini, M., Jacinto-Villegas, J. M., Filippeschi, A., Niccolin, M., and Ragaglia, M. (2016). New interaction metaphors to control a hydraulic working machine's arm. In *IEEE Symposium of Safety and rescue Robotics (SSRR)*.

Tripicchio, P., Ruffaldi, E., Gasparello, P. S., Eguchi, S., Kusuno, J., Kitano, K., Yamada, M., Argiolas, A., Niccolini, M., Ragaglia, M., and Avizzano, C. A. (2017). A stereo-panoramic telepresence system for construction machines. *Procedia Manufacturing (PROCMAN)*, 11:1552 – 1559.

Villani, L. and De Schutter, J. (2008). *Force Control*, pages 161–185. Springer Berlin Heidelberg, Berlin, Heidelberg.