applied sciences

MDPI

*Review*

# An Overview of Industrial Robots Control and Programming Approaches

**Pietro Bilancia [1,*], Juliana Schmidt [2], Roberto Raffaeli [1], Margherita Peruzzini [3] and Marcello Pellicciari [1]**

[1] Department of Sciences and Methods for Engineering, University of Modena and Reggio Emilia, 42122 Reggio Emilia, Italy
[2] InterMech MO.RE, University of Modena and Reggio Emilia, 41125 Modena, Italy
[3] Department of Engineering "Enzo Ferrari", University of Modena and Reggio Emilia, 41125 Modena, Italy
* Correspondence: pietro.bilancia@unimore.it

**Abstract:** Nowadays, manufacturing plants are required to be flexible to respond quickly to customer demands, adapting production and processes without affecting their efficiency. In this context, Industrial Robots (IRs) are a primary resource for modern factories due to their versatility which allows the execution of flexible, reconfigurable, and zero-defect manufacturing tasks. Even so, the control and programming of the commercially available IRs are limiting factors for their effective implementation, especially for dynamic production environments or when complex applications are required. These issues have stimulated the development of new technologies that support more efficient methods for robot control and programming. The goal of this research is to identify and evaluate the main approaches proposed in scientific papers and by the robotics industry in the last decades. After a critical review of the standard IR control schematic, the paper discusses the available control alternatives and summarizes their characteristics, range of applications, and remaining limitations.

**Keywords:** industrial robots; robot control; robot programming; instruction streaming; trajectory streaming; open controller



**Citation:** Bilancia, P.; Schmidt, J.; Raffaeli, R.; Peruzzini, M.; Pellicciari, M. An Overview of Industrial Robots Control and Programming Approaches. *Appl. Sci.* **2023**, *13*, 2582. https://doi.org/10.3390/app13042582

Academic Editors: Paolo Renna, Ana Martins Amaro, Paulo Nobre Balbis dos Reis and Michele Ambrico

Received: 2 February 2023
Revised: 13 February 2023
Accepted: 15 February 2023
Published: 16 February 2023



## 1. Introduction

Industry 4.0. (I4.0) has transformed almost every industrial sector, and the promotion of new digital and cyber-physical technologies has modified manufacturing plants [1]. They are now expected to produce parts at minimum cost and be able to quickly adapt their structure according to changes in production capacity and functionality [2]. Production is shifting toward the customization of products, and to cope with constant market changes, automation equipment and intelligent machines must easily reconfigure their tasks and effectively collaborate with each other to achieve the highest productivity and quality. One of the main challenges of I4.0 is the complexity of adapting to engineering changes while supporting different hardware and software platforms while maintaining adequate levels of robustness, safety, and reliability of devices during process execution [3].

Robots have been increasingly used in industry over the last 30 years, especially in the automotive and electrical/electronic fields [4,5]. In particular, starting from the early 1970s, scientific and technical improvements have contributed to their diffusion, and now, Industrial Robots (IRs) play a key role in modern smart factories, performing a wide range of tasks such as material handling, picking and placing, product packing, inspection, palletization, precise assembly, machining, and additive manufacturing operations [6]. However, despite the extremely high versatility offered by their mechanical structure, the closed architecture of IR controllers is still a limiting factor for the abovementioned challenges. In other words, IRs can be defined as "flexible machines rigidly programmed and controlled".

In the past, IRs were programmed to perform repetitive tasks in a static environment. Once tasks were defined, few code changes were made over time. New manufacturing systems, on the other hand, need to dynamically adapt their processes to meet the latest production demands, especially for mass customization, zero-defect, and quality optimization. This has given rise to greater complexity in production systems and, consequently, in the IRs' control and programming.

Most of the IRs used today are completely dependent on their manufacturers' software platforms, and robot programming is based on specific proprietary robot languages offered by each robot vendor [7]. These features make it difficult to maintain, update, or add new functionalities based on the current production needs [8]. Moreover, the lack of interoperability complicates the development of interacting multi-robot control for IRs from different manufacturers. Additionally, the integration and communication between IRs and other devices (e.g., external sensors and actuators) are restricted. The heterogeneity of platforms hampers integration, especially in dynamic environments [9,10]. As a result, it is difficult for production system components to communicate and exchange data dynamically [11], which is a barrier to collaborative work. Another limitation imposed by the current architecture of IRs is the difficulty of carrying out more complex control applications since access to the low levels of the control system is restricted.

To fill these gaps and, thus, expand robotic applications, in the last years, new technologies and control alternatives have been developed in academic and industrial fields. These allow reaching a more flexible, modular, manufacturer-independent, and open architecture that can be incorporated into different industrial plants to satisfy a wider range of production needs [12,13]. The purpose of this paper is to present an overview of the most researched and tested control and programming solutions for IRs. In particular, relevant scientific documents have been identified from the literature and compared. Along with such research works, the technologies offered by IR manufacturers have also been considered. For each control approach, the pros and cons are highlighted and discussed. Then, the current open technical issues are outlined to support future research.

The remainder of the paper is organized as follows. Section 2 briefly describes the method used to collect the documents evaluated in this research. Section 3 reports the traditional IR control schematic and programming approaches. Section 4 analyzes and compares the innovative control architectures proposed in the selected papers and discusses their applicability in the I4.0 environment, and Section 5 provides the concluding remarks.

## 2. Materials and Methods

A literature search has been carried out considering both industrial and scientific fields to obtain the results proposed in the present work. In particular, the aim of the literature analysis is to identify current issues and needs in the field of IR control and programming and to support discussions on the available tools and approaches. Therefore, different from comprehensive review studies, the state-of-the-art has been limited to scientific documents that are relevant to the industrial sector and have the potential for practical applications in industrial settings. In this sense, purely academic works that have not been tested or proven applicable to actual industrial controllers have not been considered. The widely disseminated databases Scopus, Google Scholar, and Web of Science were chosen to carry out the study. The preliminary definition of the keywords to be used in the searches was based on previously found contributions aligned with the topic. Some examples are "*(Industrial) Robot Control*", "*Industrial Robot Programming*", "*Robot Motion Control*", "*Distributed Control Architecture*", "*Reconfigurable Control*", and "*Open Robot Control*".

At the first level, about 200 papers were selected and subsequently processed. As it takes time for papers to be disseminated, the number of citations cannot be directly associated with the relevance of a paper, especially if published in the last few years. Therefore, the documents were read and evaluated according to their fit with the topic rather than the number of citations. The papers were chosen based on the authors' judgment (with particular emphasis placed on architectures that can be readily applied to industrial

assets), and their references were carefully checked. Along with the scientific literature review, commercially available industrial technologies were checked in this work, focusing on the main robot vendors. These were analyzed based on the content available online (i.e., datasheets) and the documents (i.e., advanced or specific manuals) made available by the manufacturers upon request. For each IR controller, the researched information is mainly related to the interface options, communication protocols, and settings but also to the elements that are externally accessible to the user.

## 3. Industrial Robot Control System

Automated production plants are controlled by robust control systems that require little or no human intervention. In general terms, an industrial controller sends commands to the equipment to operate a specific process and receives feedback information that allows it to monitor and determine the correct execution of such commands. Modern production systems are governed and supervised by multiple controllers, namely one or more Programmable Logic Controllers (PLCs) that coordinate the entire process providing I/O exchange, task sequencing, and action triggering, and a group of control units dedicated to single machines or sub-systems (e.g., IRs, Numerical Controls, electric axes, pneumatic/hydraulic actuators and tools, and additional external sensors) [7]. In this way, low-level control logics are still handled and solved locally for each slave controller. In the case of an IR controller, a special architecture comprising both real-time and nonreal-time modules must be implemented in order to be able to process the received requests/instructions and, in parallel, to run advanced motion control algorithms, as shown in Figure 1.

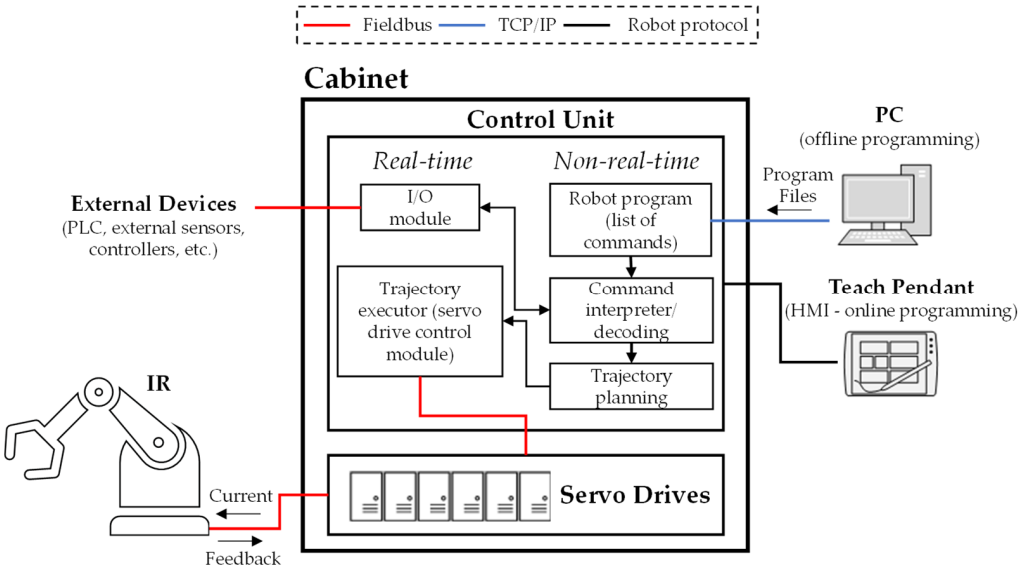



**Figure 1.** Traditional IR controller schematic.

In particular, adopting a standard IR control approach, the robot's task, namely the sequence of operations and motion to be performed within the programmed working cycle, is defined by the user via a set of instructions coded in a vendor-specific programming language and directly downloaded on the controller unit. Such instructions are progressively read, interpreted, and passed through the path planning and trajectory generation modules [14]. The path planning calculates a feasible path that connects the starting point (i.e., an initial robot configuration) to the endpoint in the end-effector space, whereas the trajectory generator combines geometric and time information and provides as an output of the inverse kinematic problem the motion law in the joint space. Then, the calculated position set-points are cyclically sent to the servo drives, which modulate and deliver the electric current to the servomotors by means of a closed-loop control based on the servomotors' position feedback (i.e., angular encoder mounted on each servomotor).

All the described steps are traditionally performed in the IR cabinet, where the servo drives and other electrical components are located. Programming the control logic and the robot task via either online or offline methods (as schematized in Figure 1) is the oldest and most widespread practice in industry. Nevertheless, such rigid approaches lead to the following practical issues (also summarized in Table 1) that strongly impact the IRs' performance and, thus, limit their application in modern, fully reconfigurable production systems.

**(1) Time-waste:** Any requested change in a robotic task/sequence/cycle requires an interruption of the plant operation, a manual modification of the robot program, and its subsequent download on the robot controller. Then, before returning the IR to full operability, extra time to reinitialize the system and perform a first movement at a very low velocity is needed by the controller [15–17], inevitably causing delays in the production system [18].

**(2) Code redundancy:** A modern production line can involve hundreds of robots, each with its own code executed at a local level. In most cases, similar tasks (i.e., procedures within the IR's working cycle) are programmed multiple times within different scripts and downloaded on each IR controller, inevitably increasing the number of robot programs that need maintenance. This condition favors the emergence of failures and increases the time required for any reconfiguration because the operators in charge of updating the code on many stand-alone controllers do not communicate [3,19].

**(3) Limitations of programming languages:** In contrast to machine tool programming (based, e.g., on G-code), the lack of a universal, manufacturer-independent programming language complicates the integration of different robot technologies within a single production plant. Indeed, the language of each robot manufacturer (e.g., KUKA, ABB, Fanuc, Stäubli) differs in complexity, syntax, and semantics, requiring specialized personnel [20]. Moreover, robot programmers are compelled to use basic commands and libraries. These cover the majority of standard programming needs, but they do not allow for performing advanced calculations and elaborate complex control strategies. Generalized offline programming tools (e.g., RoboDK, Siemens Process Simulate) use specific postprocessors to translate the 3D modeling commands and features into ready-to-use robot codes containing vendor-specific instructions. However, they still do not cover the entire function libraries of each commercial IR language and cannot replace a skilled user when more complex programming routines are needed. It should also be noted that the development of a standardized programming language is not supported by IR vendors, which typically do not provide detailed info on their low-level control schemas [21].

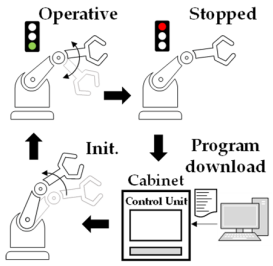
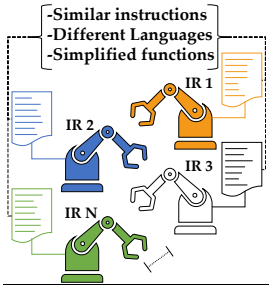
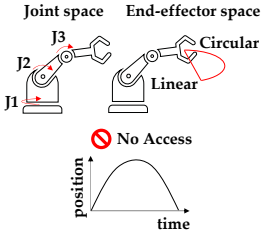
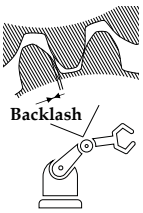
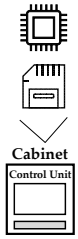
**(4) Limited motion control:** Complex robot paths (e.g., the ones required for robot machining, gluing, or precision assembly) cannot be efficiently applied via traditional programming, where the available motion instructions are in most cases limited to point-to-point (programmed in the joint space) or linear/circular movements of the robot end-effector [15–17]. In principle, one could discretize the desired non-trivial path into a set of smaller segments and proceed with the standard motion functions, although this would increment the controller's computational load and slow down the robot velocity due to the increased number of path corners. Indeed, with path cornering, the only settings available to the programmer are the maximum velocity and acceleration, while the profile type is elaborated by the controller and cannot be modified [22]. In general, the user has no access to any motion law (position vs. time and its derivatives) generated from the coded instructions.

**(5) No disturbs compensation:** Undesired dynamic effects, which seriously affect the IRs' position accuracy and, thus, preclude their use in precision manufacturing tasks, cannot be compensated utilizing the standard control and programming approaches. This is primarily due to the limited user access to any low-level motion control algorithm running in the trajectory generator (see point 4). As for the hardware, it should be noted that the use of servomotor-side angular encoders does not allow for capturing the nonlinearities introduced by the speed reducers (e.g., hysteresis and dynamic lost motion) [6,23–25]

and, therefore, to provide rich feedback information to the controller [26]. At last, the update time of the controller's real-time module is usually too high (in the order of a few milliseconds) for the correction of high-frequency disturbances originating in the reducers.

**(6) Outdated parts:** IR controllers comprise many obsolete electronic parts, starting from the main CPU (e.g., Pentium 4 single-core processor) and its related operating system (e.g., Windows XP). Moreover, the available space for data storage on the controller is quite limited (several times even 25 MB), and, therefore, the dimensions of newly downloaded items (programs and data files) must always be checked.

**Table 1.** Practical issues encountered with a traditional IR controller (numbered as in Section 3) and available solutions.

| | **Traditional IR Controller Issues** | | | | |
|---|---|---|---|---|---|
| | **Software** | | | **Hardware** | |
| **Available solutions** ↓ | **Issue (1)**   | **Issues (2) and (3)**   | **Issue (4)**   | **Issue (5)**   | **Issue (6)**   |
| External coordinator | ✔ | ✔ | ✖ | ✖ | ✖ |
| External traj. gen. | ✖ | ✖ | ✔ | ✖ | ✖ |
| Open controller | ✔ | ✔ | ✔ | ✔ | ✔ |

To overcome these gaps and, thus, meet the need for collaboration, reconfiguration, and rapid modification of IR tasks, in the last decade, advances have been made both in academic research works and by robot manufacturers with the introduction of more flexible alternatives for IR control, which will be detailed in Section 4.

## 4. Innovative Control Approaches

In this section, innovative control solutions for expanding IR capabilities are described, and examples from the literature are reported. These approaches have been conceived for easy and rapid integration in the already established production environment, i.e., by ensuring minimal intervention and replacement of pre-existing hardware and software modules.

- **External coordinator** (Figure 2a): This option makes it possible to send commands to the robot using an external device (PC or PLC), where the robot program is written in a generic language. On the robot controller, interpreter software runs cyclically to convert the received commands into robot-specific language. The translated commands are then processed, and the trajectory planning is performed on the robot controller as in the traditional approach described in Section 3.
- **External trajectory generator** (Figure 2b): The trajectory planning is performed with an external device which can be either an industrial PC equipped with a real-time board or a PLC. By implementing fast-cyclic communication, the computed motion profile and the robot feedback are, respectively, sent and received in a timely manner through the robot's I/O module. As can be noted in the figure, the non-real-time module of the robot controller is no longer involved in motion planning, although it can still be used to process other types of instructions and logical events.

- **Open controller** (Figure 2c): Since access to the lowest control levels is restricted for most IR controllers, some of the previously discussed issues (see Table 1) may lead to the partial/complete replacement of the original system with a new open system, offering the possibility of a complete tailored control design that extends the robot's operational flexibility and, thus, meets the I4.0 requirements. In particular, in addition to the original robot controller and related electronic boards, one may decide to replace drive units and servomotors to potentially achieve the best motion control performance (e.g., by employing encoders with higher resolutions).

In the following sections, the solutions offered by both IR manufacturers and academic researchers are recalled and described.

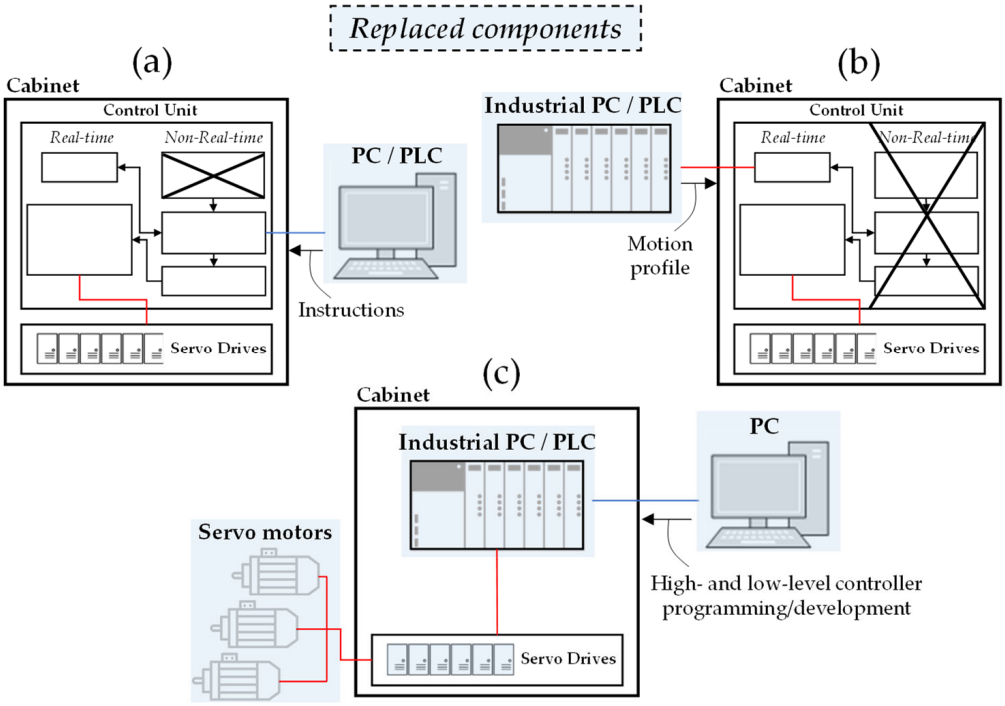



**Figure 2.** Schematic of the innovative IR control approaches. (**a**) External coordinator, (**b**) External trajectory generator, and (**c**) Open controller.

### *4.1. External Coordinator*

Approaches that adopt the stream of instructions to the robot controller offer an intuitive programming interface to the operator, which avoids any vendor-specific robot programming language. Robots are programmed in a higher-level environment (e.g., PLC editor or Python) running on an external device, and the elaborated instructions are progressively transmitted to the robot controller, where an interpreter written in its native language is implemented [20]. For this purpose, a standard PC may be used as no real-time communication with the robot is required to transmit instructions and to read/write variables, structures, and registers. Following the schematic reported in Figure 2b, in the last decade, technologies have been developed to stream data and commands between robots and external devices. These can be divided into commercial packages (already released on the market) and research-oriented packages (still under development/refinement).

The Fanuc PC Developer's Kit (PCDK) [27] falls into the first group. This is a tool that enables communication of information and instructions between an external PC and a Fanuc robot controller (R-J3 and R-J3iB). Some of the PCDK's actions are reading/writing variables and numeric registers, testing, setting, and configuring inputs and outputs, reading/writing positions, and loading/saving programs. Similar solutions offered by ABB are presented in [28,29]. These customized applications run on an external PC that communicates with the robot controller over a network by means of specific handshaking

routines. Recently, robot manufacturers have introduced new technologies that make it possible to dynamically stream a sequence of commands to the robot using an external PLC, such as mxAutomation (KUKA), motoLogix (YASKAWA), uniVAL (Stäubli), and PLC Motion Interface (Fanuc). The main benefit is that all programming tasks can now be performed through an external PLC (Figure 3), allowing robots and other devices installed in the production plant to be managed from one centralized location (i.e., using the PLC as the external coordinator). These technologies essentially consist of two main parts: (i) a server program that runs on the robot controller and waits for commands coming from the external PLC, and (ii) a coordinator program that runs on the PLC. By employing such PLC libraries provided by the robot vendors, all the original robot commands and parameters are packed into the corresponding data format and subsequently streamed via fieldbus, UDP, or TCP/IP to the server. Here, the received packets are interpreted and then executed, and specific parameters and messages are sent back to the PLC. The use of a single, centralized, robust programming environment extends manufacturing flexibility, facilitates the commissioning of big plants, and reduces their installation and maintenance costs as it lowers the number of interventions and specialized personnel.

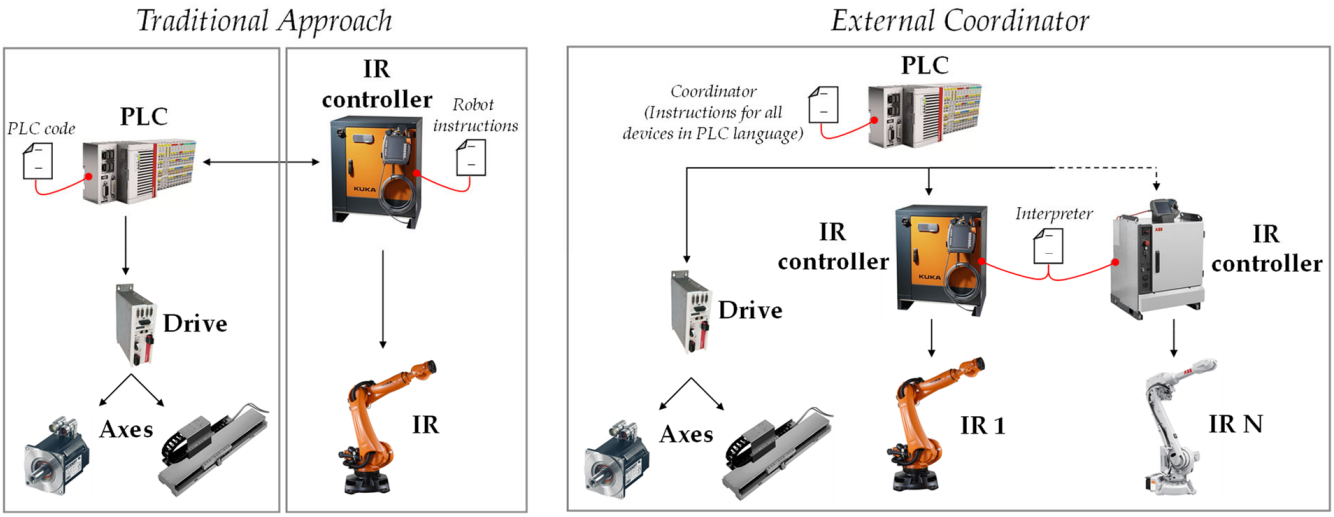



**Figure 3.** Traditional framework with multiple parallel controllers vs. approach adopting a centralized PLC coordinator to control the production plant.

KUKA mxAutomation appears in [30], where the authors present the use of visual-programming environments that allow the automatic generation of KRL (KUKA Robot Language) code, subsequently copied to the robot. This file-based workflow causes delays between file definition and execution, highlighting the need to link the external computer with the robot directly. mxAutomation is considered a potential solution since it enables the stream of all KUKA commands and feedback data. An industrial application is reported in [31], where two KUKA robots, i.e., one articulated Agilus robot and one LBR iiwa collaborative robot, are remotely controlled to work in a collaborative way to assist workers in lifting heavy loads. The authors call attention to the need to reach an optimal size of the commands' buffer.

Following the same premise, academic researchers have proposed similar solutions based on PCs, with the possibility of exploiting more programming languages. A summary of their work is reported in Table 2. for comparison. For example, Lambrecht et al. [20] proposed a proof-of-concept control layer that allows for adaptive process and motion planning. It consists of an interpreter program running on arbitrary robot controllers that translates the incoming generic commands into internal vendor-specific control commands. The interpreter was tested (with a limited set of functions) on KUKA, Comau, and Adept platforms, although its generalized structure enables its possible extension to other IRs, as also claimed by the authors. In the same direction, Deatcu et al. [32] presented a MAT-

LAB toolbox for programming KUKA and Kawasaki IRs. The data exchanged between the MATLAB program and the interpreter on the robot controller is performed through data links (i.e., serial or TCP/IP). A solution that incorporates event-based logic based on Python, specifically developed to efficiently use Adept robots in distributed environments, is discussed in [33]. The communication is via TCP/IP, and commands are communicated as string data. As a part of the open-source ROS initiative, a project named ROS-industrial provides interfaces and libraries for controlling IRs from different vendors from an external PC adopting a TCP/IP communication [34,35]. However, as clearly documented on their official website (https://rosindustrial.org/), many of these solutions are still under development and do not cover all the safety conditions that may be required to control an IR.

Adopting a different approach (i.e., not based on instruction streaming), the open-source tool JOpenShowVar presented in [36] enables the user to read and write all the native variables in a KUKA controller without the need for complementary proprietary packages. The system has a client/server architecture, with JOpenShowVar as a client in a remote PC and KUKAVARPROXY as a server in the robot controller. The communication between the client and server is via TCP/IP; therefore, real-time access to the robot's data is not guaranteed, and the system cannot be used to stream point-to-point trajectories. This can be done with the KUKA Control Toolbox (KCT) presented in [37], which includes a set of MATLAB functions for kinematics calculation, motion planning, visualization, and diagnostics of KUKA robots. Compared with the previous coordinators, the KCT preserves the aim to control an IR from a remote PC, although this tool is based on point streaming rather than instruction streaming, as in the case of the real-time trajectory generators described in Section 4.3.

The main differences between industrial and academic products can be rapidly identified in Table 2. Indeed, while scientists always try to gain as much control over the robot as possible, fully industrial solutions seek safe, robust, and easy operational interfaces. Overall, the recalled frameworks raise the level of abstraction in robot programming since the program is created and executed using an external controller (i.e., coordinator) without losing the reliability and safety normally offered by IR controllers. In contrast, they are intrinsically linked to specific manufacturers, as also notable from Table 2. Furthermore, in many of them, there is little interaction with the robot as it is only possible to access and manipulate specific commands or datasets. Another important open issue, scarcely addressed in the referenced academic works, is error management. For a more effective and robust robot control, errors must be detected during communication, interpretation, and execution of commands in the IR controller. Software strategies for their efficient recognition and subsequent handling must therefore be implemented in the future.

**Table 2.** Examples of external coordinators from literature.

| Ref. | Tool Name | Environment | IR Controller | Pros/Cons |
|---|---|---|---|---|
| [30] | mxAutomation | PLC | KUKA | All IR commands; easy integration; robust and reliable |
| [20] | N.D. | PC | KUKA/Comau/Adept | Limited number of IR commands; easy integration |
| [28] | RAP | PC (Robotstudio) | ABB | Not all IR commands; tested with one slave IR controller |
| [29] | ABB PC SDK | PC | ABB | Code editing (no command streaming); easy integration |
| [27] | Fanuc PCDK | PC | Fanuc | Remote variables editing; simple command streaming (e.g., run prog) |
| [36] | JOpenShowVar | PC | KUKA | Remote variables and structures editing (no command streaming) |
| [37] | KCT | PC (Matlab) | KUKA (RSI) | Functions for motion planning/control (no command streaming) |
| [32] | RCV Toolbox | PC (Matlab) | KUKA/Kawasaki | Limited number of IR commands; good command buffer |
| [33] | pyadept | PC (Python) | Adept | All IR commands; good decoupling between coordinator and slave |

### *4.2. External Trajectory Generator*

As outlined in Section 3, there may be situations where fine motion control is needed, such as for industrial applications requiring specific geometric paths and/or motion laws

to be imposed/monitored at the end-effector. Ready examples are robotic machining, handling of delicate objects, precise assembly, and inspection. With reference to Table 1, in such cases, an external coordinator is no longer a valid solution as it only supports the robot's native motion instructions (e.g., joint and linear/circular movements) and fully exploits the internal trajectory planning module to determine the motion profiles via proprietary (inaccessible) algorithms. A readily applicable solution, often proposed due to the little intervention required on the robotic system, is the integration of an external trajectory generator. Obviously, in order to proceed in this direction, the IR controller must first be equipped with proper real-time interfaces that allow externally guided motions. In practice, these interfaces must ensure fast and robust cyclic communication with the external device (e.g., an industrial PC or PLC, as shown in Figure 2b) in charge of the trajectory computation.

Adopting this framework, the external position data, eventually computed with a closed-loop approach based on the information received from additional sensors (position or force sensors, as shown in Figure 4), must be cyclically streamed to the robot controller at a specific rate. Since the robot motion is calculated without all the consolidated algorithms and well-tuned parameters of the original robot controller, in sensor-assisted operations, the robot can move unexpectedly due to incorrect signal definition or hardware fault (e.g., sensor malfunctioning). Consequently, proper security checks and safety procedures should be adopted to reduce the risk of danger.

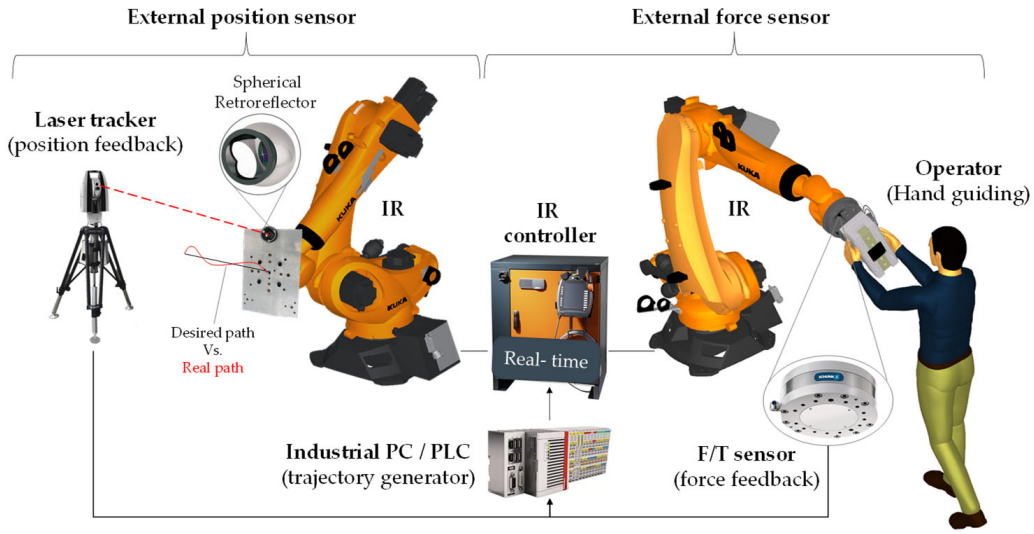



**Figure 4.** Closed-loop real-time position- and force-based guided motions. On the left, an example of precise robot positioning with laser tracker feedback; on the right, an application of hand-guiding on a high payload IR.

Some of the market-available add-on technology packages that allow real-time robot control are the KUKA Robot Sensor Interface (RSI), the ABB External Guided Motion (EGM), and the Fanuc Dynamic Path Modification (DPM). With the RSI, the data exchange can be performed either via Ethernet (UDP/IP protocol) or the I/O system of the robot controller (fieldbus). The RSI System supports two cycle rates, i.e., respectively, 12 ms and 4 ms. The position information must be sent to the RSI module within each cycle, and the received signals influence the robot's movement. To prevent possible damages and respect the servo drive's physical limitations, the RSI also monitors and eventually decreases the commanded correction value. ABB EGM offers the same functionalities but is split into two separate operation modes: (i) EGM Position Guidance and (ii) EGM Path Correction. The first is adopted to move the robot along a path consisting of position data generated externally and streamed every 4 ms (with a control lag of up to 20 ms), whereas the latter allows for real-time path corrections using sensor data. In this case, corrections must be provided in the robot's coordinate system every 48 ms. The double operation

mode is also proposed by Fanuc with the modal DPM and the inline DPM. The first is intended for applications that require continuous modification along the designed path, with a cycle rate of 8 ms and a reduced control lag. As for the inline DPM, each motion segment is adjusted only at its final destination with an offset value. From a comparison, the RSI potentially outperforms the other solutions by allowing dynamic compensations up to 250 Hz. However, the bandwidth of the established closed-loop system varies on a case-by-case basis, and more practical evaluations should also consider communication delays and control lags.

The described packages have been widely employed by many academic researchers. A summary of their contributions is reported in Table 3. A first example of an RSI application, already cited in Section 4.1, is the KCT toolbox. In this case, the trajectory generated via MATLAB is streamed to the robot without implementing any real-time feedback (i.e., sensor signal). An extended version, named RoBO-2L, is introduced in [38] to deal with older generation robot controllers. The implementation of an external sensor can be seen, for example, in [39], where a vision system is attached to the robot's end-effector to allow the robot to perform tasks without a predefined program. After a series of tests, the authors concluded that it is possible to use RSI not only for minor path corrections but also for manipulating objects in an adaptive way. Another application, still based on the use of a vision system (C-Track 780 from Creaform), is described in [40], where the authors successfully demonstrated their dynamic pose correction systems by reaching an IR pose accuracy of 0.050 mm. As for the online path compensation during robot traveling (see the left schematic of Figure 4), effective solutions implemented on KUKA, ABB, and MABI robots can be found, respectively, in [41,42], in [43] and in [44], where positioning errors are reduced by about 90%. These results become particularly important for extending the use of IRs to large-scale machining operations. To this aim, promising results have also been obtained by means of force-based compensations, as documented in [45–47]. In general, the adoption of force/torque sensors directly mounted on the robot's end-effector has been extended to many fields, ranging from automatic part handling to the execution of collaborative tasks (e.g., operator hand-guiding as depicted in Figure 4 and described in [48,49]).

**Table 3.** Examples of external trajectory generators from literature.

| Ref. | Feedback | Environment | IR Vendor | Application | Cycle Rate |
|---|---|---|---|---|---|
| [39] | Position (vision) | PLC-based | KUKA (RSI) | Adaptive pick-and-place in robotic lines | 12 ms |
| [44] | Position (tracker) | PC-based | MABI (UCI-App) | Path correction in machining | 8 ms |
| [40] | Position (vision) | PC-based | Fanuc (DPM) | Pose correction with cost-effective strategies | N.D. |
| [41] | Position (tracker) | PC-based | KUKA (RSI) | Path correction in drilling/milling | 12 ms |
| [42] | Position (tracker) | PLC-based | KUKA (RSI) | Accurate path following | 4 ms |
| [45] | Force (F/T sensor) | PC-based | ABB | Force-controlled grinding/deburring | 4 ms |
| [46] | Force (F/T sensor) | PLC-based | KUKA (RSI) | Hybrid force-controlled milling | 4 ms |
| [48] | Force (F/T sensor) | PC-based | Comau | Hand-guided robot motion | 2 ms |
| [49] | Force (F/T sensor) | PC-based | KUKA (RSI) | Hand-guided robot motion | 12 ms |

These examples prove the advantage of controlling the robot considering specific (i.e., application-dependent) needs. In fact, the use of sensors capable of guiding or protecting the robot during movement allows IRs to replace exclusive machines normally used in production lines. However, it should once again be recalled that the user becomes responsible for manipulating parameters such as velocity, acceleration, and position, as well as defining the interpolation strategies. If any detail goes unnoticed, mechanical damages may not only make further IR use unfeasible but may also pose serious risks to users [50]. Problems such as vibrations during robot motion were observed in experiments performed with non-optimized codes. Other crucial points are the limited set of functions and I/O and a major (unavoidable) economical investment for packages purchase.

### 4.3. Open Controller

Despite the positive outcomes achievable with the control solutions described in Sections 4.1 and 4.2, access to the lowest control layers remains restricted for most IRs. Therefore, some research works have proposed the replacement of the original equipment (Figure 2c) and the implementation of an open, fully programmable control system [51,52]. Important contributions in this area were reached with the European project OSACA (Open System Architecture for Control within Automation Systems), in Japan with the OSEC (Open System Environment for Controllers), and in the US with the OMAC (Open Modular Architecture Control). Many of the proposed solutions utilize a PC-based architecture (see [53–58]), the PC itself being an open, versatile, and highly customizable platform. For example, in [58], the authors proposed a low-cost control system capable of being implemented in obsolete IRs. The control architecture consisted of a standard PC and an interface based on FPGA (Field-programmable Gate Array). The PC is responsible for motion interpolation, kinematics computation, and transmitting the joint position reference and receiving the current position. The role of the FPGA is to decode feedback position data, compute the control output and commutation signals, and transmit such data to the actuators. With the same goal, in [59], the original robot controller is replaced by an embedded industrial CompactRIO controller programmed in LabVIEW.

For industrial facilities where PLCs are already utilized in other machine control applications, moving to PLC-based robotic controls (as in [60]) would be particularly convenient by leveraging (i) internal know-how of engineers and technicians (i.e., simplified training), (ii) already installed hardware (cabinets, electronic modules, cables, etc.) and software (e.g., programming editors, simulation and debugging tools, interfaces) environments, and (iii) available documentation (i.e., simplified troubleshooting and maintenance). Other important benefits include the easy integration of multiple devices within the same production environment with efficient interfaces and communication protocols. As an example, let us consider a robotic cell for material handling, where infeed/discharge conveyors, pallet dispensers, and other system components must cooperate with the IR to achieve the desired functionality. In this case, the use of a reduced number of controllers (ideally one) and boards would greatly lower the complexity of the overall production system and ease its setup and commissioning phases by limiting the effort spent on programming and interfacing the installed mechatronic devices. The most widespread platforms for developing open solutions for logic and motion control in automation, already integrated within their related commercial PLCs, are Siemens Sinumerik, Bosch IndraMotion, and Beckhoff TwinCAT, the latter being particularly flexible as it potentially turns any standard PC into an effective PLC with real-time I/O exchanges.

Naturally, the replacement of the original controllers, although interesting for the development of fully customized control algorithms aiming, e.g., at improving the performance of existing machines, may not be feasible in all industrial installations due to high costs and safety reasons. Indeed, on the one hand, substituting the original hardware requires significant expenses and time for establishing the new setup. On the other hand, it leads to the loss of safety requirements already established by the manufacturers, which are quite intricate to be reproduced in new controllers.

## 5. Conclusions

This work has explored control and programming approaches for extending IRs' potentialities and operability in the context of I4.0 production systems. The main problems with the original IR controllers are the difficulty in reconfiguring and reusing program codes and the excessively constrained control schematic, which strongly limits the development of personalized and reconfigurable procedures. These issues fueled the interest in new control and programming alternatives presented and discussed in the present research.

As a first step, a literature search accompanied by the identification of the most recent industrial technologies was carried out. From the selected material, the innovative control alternatives were evaluated and divided into three main categories, i.e., external

coordinators, external trajectory generators, and open controllers. For each category, the presented works were summarized into comparative tables based on their scope, technical features, purpose, strengths, and gaps/limitations.

It was verified that the usefulness of each alternative depends on the robot application and on the available software/hardware resources. Moreover, it was observed that further steps still need to be taken for these new solutions to be practically implemented in the industry. Industrial technologies need to take greater steps toward the required interoperability. Academic works, on the other hand, must deepen the validation phase and test the robustness of the proposed controllers under realistic scenarios coming from industry. This proves the need for close collaboration to speed up the commissioning and delivery of innovative solutions into modern production systems.

**Author Contributions:** Conceptualization, P.B.; methodology, P.B. and R.R.; formal analysis, P.B.; investigation, J.S.; writing—original draft preparation, J.S.; writing—review and editing, P.B. and M.P. (Margherita Peruzzini); coordination, M.P. (Marcelo Pellicciari). All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Zhong, R.Y.; Xu, X.; Klotz, E.; Newman, S.T. Intelligent Manufacturing in the Context of Industry 4.0: A Review. *Engineering* **2017**, *3*, 616–630. [CrossRef]
2. El Maraghy, H.A. Flexible and Reconfigurable Manufacturing Systems Paradigms. In *Flexible Services and Manufacturing Journal*; Springer: New York, NY, USA, 2006; Volume 17, pp. 261–276.
3. Sun, Y.; Gray, J.; Bulheller, K.; von Baillou, N. A Model-Driven Approach to Support Engineering Changes in Industrial Robotics Software. In *Lecture Notes in Computer Science*; (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Springer: Berlin/Heidelberg, Germany, 2012; Volume 7590, pp. 368–382.
4. Cheng, H.; Jia, R.; Li, D.; Li, H. The Rise of Robots in China. *J. Econ. Perspect.* **2019**, *33*, 71–88. [CrossRef]
5. Oztemel, E.; Gursev, S. Literature Review of Industry 4.0 and Related Technologies. *J. Intell. Manuf.* **2020**, *31*, 127–182. [CrossRef]
6. Pham, A.D.; Ahn, H.J. Rigid Precision Reducers for Machining Industrial Robots. *Int. J. Precis. Eng. Manuf.* **2021**, *22*, 1469–1486. [CrossRef]
7. Raffaeli, R.; Bilancia, P.; Neri, F.; Peruzzini, M.; Pellicciari, M. Engineering Method and Tool for the Complete Virtual Commissioning of Robotic Cells. *Appl. Sci.* **2022**, *12*, 3164. [CrossRef]
8. Estévez, E.; Sánchez-García, A.; Gámez-García, J.; Gómez-Ortega, J.; Satorres-Martínez, S. A Novel Model-Driven Approach to Support Development Cycle of Robotic Systems. *Int. J. Adv. Manuf. Technol.* **2016**, *82*, 737–751. [CrossRef]
9. Wojtynek, M.; Steil, J.J.; Wrede, S. Plug, Plan and Produce as Enabler for Easy Workcell Setup and Collaborative Robot Programming in Smart Factories. *KI—Kunstl. Intell.* **2019**, *33*, 151–161. [CrossRef]
10. Salcic, Z.; Atmojo, U.D.; Park, H.; Chen, A.T.Y.; Wang, K.I.K. Designing Dynamic and Collaborative Automation and Robotics Software Systems. *IEEE Trans. Ind. Inform.* **2019**, *15*, 540–549. [CrossRef]
11. Fischer, H.; Vulliez, M.; Laguillaumie, P.; Vulliez, P.; Gazeau, J.P. RTRobMultiAxisControl: A Framework for Real-Time Multiaxis and Multirobot Control. *IEEE Trans. Autom. Sci. Eng.* **2019**, *16*, 1205–1217. [CrossRef]
12. Arents, J.; Greitans, M. Smart Industrial Robot Control Trends, Challenges and Opportunities Within Manufacturing. *Appl. Sci.* **2022**, *12*, 937. [CrossRef]
13. Pan, Z.; Polden, J.; Larkin, N.; van Duin, S.; Norrish, J. Recent Progress on Programming Methods for Industrial Robots. *Robot. Comput. Integr. Manuf.* **2012**, *28*, 87–94. [CrossRef]
14. Siciliano, B.; Khatib, O.; Kröger, T. *Springer Handbook of Robotics*; Springer: Berlin/Heidelberg, Germany, 2008; Volume 200.
15. KUKA Deutschland GmbH System Software 8.7—Operating and Programming Instructions for System Integrators. Available online: www.kuka.com (accessed on 1 February 2023).
16. ABB Robotics Technical Reference Manual—RAPID Overview. Available online: www.abb.com/robotics (accessed on 1 February 2023).
17. FANUC Robotics FANUC Robotics SYSTEM J-30iB Controller KAREL Reference Manual. Available online: www.fanucamerica.com (accessed on 1 February 2023).

18. Llopis-Albert, C.; Rubio, F.; Valero, F. Modelling an Industrial Robot and Its Impact on Productivity. *Mathematics* **2021**, *9*, 769. [CrossRef]
19. Valente, A.; Mazzolini, M.; Carpanzano, E. An Approach to Design and Develop Reconfigurable Control Software for Highly Automated Production Systems. *Int. J. Comput. Integr. Manuf.* **2015**, *28*, 321–336. [CrossRef]
20. Lambrecht, J.; Chemnitz, M.; Kruger, J. Control Layer for Multi-Vendor Industrial Robot Interaction Providing Integration of Supervisory Process Control and Multifunctional Control Units. In Proceedings of the 2011 IEEE Conference on Technologies for Practical Robot Applications (TePRA), Woburn, MA, USA, 11–12 April 2011; pp. 115–120.
21. Chan, S.F.; Kwan, R. Post-Processing Methodologies for off-Line Robot Programming within Computer Integrated Manufacture. *J. Mater. Process. Technol.* **2003**, *139*, 8–14. [CrossRef]
22. Bigliardi, M.; Bilancia, P.; Raffaeli, R.; Peruzzini, M.; Berselli, G.; Pellicciari, M. Path Approximation Strategies for Robot Manufacturing: A Preliminary Experimental Evaluation. In Proceedings of the International Joint Conference on Mechanics, Design Engineering & Advanced Manufacturing, Ischia, Italy, 1–3 June 2022; pp. 380–389.
23. Bilancia, P.; Monari, L.; Raffaeli, R.; Peruzzini, M.; Pellicciari, M. Accurate Transmission Performance Evaluation of Servo-Mechanisms for Robots. *Robot. Comput. Integr. Manuf.* **2022**, *78*, 102400. [CrossRef]
24. Hu, Y.; Li, G.; Zhu, W.; Cui, J. An Elastic Transmission Error Compensation Method for Rotary Vector Speed Reducers Based on Error Sensitivity Analysis. *Appl. Sci.* **2020**, *10*, 481. [CrossRef]
25. Xu, H.; Shi, Z.; Yu, B.; Wang, H. Optimal Measurement Speed and Its Determination Method in the Transmission Precision Evaluation of Precision Reducers. *Appl. Sci.* **2019**, *9*, 2146. [CrossRef]
26. Mesmer, P.; Neubauer, M.; Lechler, A.; Verl, A. Robust Design of Independent Joint Control of Industrial Robots with Secondary Encoders. *Robot. Comput. Integr. Manuf.* **2022**, *73*, 102232. [CrossRef]
27. Zhang, H.; Yang, X. Interface between LabVIEW and FANUC Robot. In *Advanced Materials Research*; Trans Tech Publications Ltd.: Wollerau, Switzerland, 2012; Volume 443–444, pp. 464–470.
28. Bolmsjö, G.; Cederberg, P.; Olsson, M. Remote Control of a Standard ABB Robot System in Real Time Using the Robot Application Protocol (RAP). In Proceedings of the 33rd ISR (International Symposium on Robotics), Stockholm, Sweden, 7–11 October 2002.
29. Dalvand, M.M.; Nahavandi, S. Teleoperation of ABB Industrial Robots. *Ind. Robot.* **2014**, *41*, 286–295. [CrossRef]
30. Munz, H.; Braumann, J.; Brell-Cokcan, S. Direct Robot Control with MxAutomation: A New Approach to Simple Software Integration of Robots in Production Machinery, Automation Systems, and New Parametric Environments. In *Robotic Fabrication in Architecture, Art and Design 2016*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 440–447.
31. Institute of Electrical and Electronics Engineers. *Collaborative Control with Industrial Robots*; Institute of Electrical and Electronics Engineers: Piscataway, NJ, USA, 2017; ISBN 9781538618837.
32. Deatcu, C.; Freymann, B.; Schmidt, A.; Pawletta, T. MATLAB/Simulink Based Rapid Control Prototyping for Multivendor Robot Applications. *Simul. Notes Eur.* **2015**, *25*, 69–78. [CrossRef]
33. Semeniuta, O.; Falkman, P. Event-Driven Industrial Robot Control Architecture for the Adept V+ Platform. *PeerJ Comput. Sci.* **2019**, *5*, e207. [CrossRef] [PubMed]
34. Elkady, A.; Sobh, T. Robotics Middleware: A Comprehensive Literature Survey and Attribute-Based Bibliography. *J. Robot.* **2012**, *2012*, 1–15. [CrossRef]
35. Michieletto, S.; Tosello, E.; Romanelli, F.; Ferrara, V.; Menegatti, E. ROS-I Interface for COMAU Robots. In *Lecture Notes in Computer Science*; (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Springer: Berlin/Heidelberg, Germany, 2014; Volume 8810, pp. 243–254. [CrossRef]
36. Sanfilippo, F.; Hatledal, L.I.; Zhang, H.; Fago, M.; Pettersen, K.Y. Controlling Kuka Industrial Robots. *IEEE Robot. Autom. Mag.* **2015**, *22*, 96–109. [CrossRef]
37. Chinello, F.; Scheggi, S.; Morbidi, F.; Prattichizzo, D. KUKA Control Toolbox. *IEEE Robot. Autom. Mag.* **2011**, *18*, 69–79. [CrossRef]
38. Golz, J.; Wruetz, T.; Eickmann, D.; Biesenbach, R. RoBO-2L, a Matlab Interface for Extended Offline Programming of KUKA Industrial Robots. In Proceedings of the 2016 11th France-Japan and 9th Europe-Asia Congress on Mechatronics, MECATRONICS 2016/17th International Conference on Research and Education in Mechatronics (REM), Compiegne, France, 15–17 June 2016; pp. 64–67.
39. Rogers, L.; Vermaak, H.J. Automated Adapting Component Transfer System Using Real-Time Robot Control within a KUKA RobotSensorInterface Environment. In Proceedings of the 2017 IEEE AFRICON, Cape Town, South Africa, 18–20 September 2017; pp. 1426–1431.
40. Gharaaty, S.; Shu, T.; Joubair, A.; Xie, W.F.; Bonev, I.A. Online Pose Correction of an Industrial Robot Using an Optical Coordinate Measure Machine System. *Int. J. Adv. Robot. Syst.* **2018**, *15*, 1729881418787915. [CrossRef]
41. Wang, Z.; Zhang, R.; Keogh, P. Real-Time Laser Tracker Compensation of Robotic Drilling and Machining. *J. Manuf. Mater. Process.* **2020**, *4*, 79. [CrossRef]
42. Cvitanic, T.; Melkote, S.N. A New Method for Closed-Loop Stability Prediction in Industrial Robots. *Robot. Comput. Integr. Manuf.* **2022**, *73*, 102218. [CrossRef]
43. Szybicki, D.; Obal, P.; Kurc, K.; Gierlak, P. Programming of Industrial Robots Using a Laser Tracker. *Sensors* **2022**, *22*, 6464. [CrossRef]

44. Moeller, C.; Schmidt, H.C.; Koch, P.; Boehlmann, C.; Kothe, S.; Wollnack, J.; Hintze, W. Real Time Pose Control of an Industrial Robotic System for Machining of Large Scale Components in Aerospace Industry Using Laser Tracker System. *SAE Int. J. Aerosp.* **2017**, *10*, 100–108. [CrossRef]
45. Blomdell, A.; Bolmsjö, G.; Brogårdh, T.; Cederberg, P.; Isaksson, M.; Johansson, R.; Haage, M.; Nilsson, K.; Olsson, M.; Olsson, T.; et al. Extending an Industrial Robot Controller: Implementation and Applications of a Fast Open Sensor Interface. *IEEE Robot. Autom. Mag.* **2005**, *12*, 85–94. [CrossRef]
46. Hähn, F.; Weigold, M. Hybrid Compliance Compensation for Path Accuracy Enhancement in Robot Machining. *Prod. Eng.* **2020**, *14*, 425–433. [CrossRef]
47. Obal, P.; Gierlak, P. Egm Toolbox—Interface for Controlling Abb Robots in Simulink. *Sensors* **2021**, *21*, 7463. [CrossRef]
48. Bascetta, L.; Ferretti, G.; Magnani, G.; Rocco, P. Walk-through Programming for Robotic Manipulators Based on Admittance Control. *Robotica* **2013**, *31*, 1143–1153. [CrossRef]
49. Loske, J.; Biesenbach, R. Force-Torque Sensor Integration in Industrial Robot Control. In Proceedings of the 2014 15th International Workshop on Research and Education in Mechatronics (REM), El Gouna, Egypt, 9–11 September 2014.
50. Braumann, J.; Brell-Cokcan, S. Adaptive Robot Control New Parametric Workflows Directly from Design to KUKA Robots. *eCAADe 2015 RealTime* **2015**, *2*, 243.
51. Ford, W.E. What Is an Open Architecture Robot Controller? In Proceedings of the IEEE International Symposium on Intelligent Control, Columbus, OH, USA, 16–18 August 1994; pp. 27–32.
52. Brecher, C.; Verl, A.; Lechler, A.; Servos, M. Open Control Systems: State of the Art. *Prod. Eng.* **2010**, *4*, 247–254. [CrossRef]
53. Liandong, P.; Xinhan, H. Implementation of a PC-Based Robot Controller with Open Architecture. In Proceedings of the Proceedings—2004 IEEE International Conference on Robotics and Biomimetics (IEEE ROBIO), Shenyang, China, 22–26 August 2004; pp. 790–794.
54. Hong, K.-S.; Choi, K.-H.; Kim, J.-G.; Lee, S. A PC-Based Open Robot Control System: PC-ORC q. *Robot. Comput. Integr. Manuf.* **2001**, *17*, 355–365. [CrossRef]
55. Jokić, D.; Lubura, S.; Rajs, V.; Bodić, M.; Šiljak, H. Two Open Solutions for Industrial Robot Control: The Case of Puma 560. *Electronics* **2020**, *9*, 972. [CrossRef]
56. Roberti, F.; Soria, C.; Slawinski, E.; Mut, V.; Carelli, R. Open Software Structure for Controlling Industrial Robot Manipulators. In *Robot Manipulators Trends and Development*; InTech: Rang-Du-Fliers, France, 2010.
57. Sawada Strategy, C.; Pacific, A.; Operations, T. Open Controller Architecture OSEC-11: Architecture Overview and Prototype Systems. In Proceedings of the 1997 IEEE 6th International Conference on Emerging Technologies and Factory Automation Proceedings, EFTA'97, Los Angeles, CA, USA, 9–12 September 1997.
58. Martínez-Prado, M.A.; Rodríguez-Reséndiz, J.; Goméz-Loenzo, R.A.; Herrera-Ruiz, G.; Franco-Gasca, L.A. An FPGA-Based Open Architecture Industrial Robot Controller. *IEEE Access* **2018**, *6*, 13407–13417. [CrossRef]
59. Trujillo, J.L.A.; Pérez-Ruiz, A.; Serrezuela, R.R. Generation and Control of Basic Geometric Trajectories for a Robot Manipulator Using CompactRIO®. *J. Robot.* **2017**, *2017*, 1–11. [CrossRef]
60. IEEE Robotics and Automation Society. PLC-Based Control of a Robot Manipulator with Closed Kinematic Chain. In Proceedings of the 2009 IEEE International Conference on Robotics and Automation, Kobe, Japan, 12–17 May 2009.