applied sciences

Review

Task Complexity and the Skills Dilemma in the Programming and Control of Collaborative Robots for Manufacturing

Peter George, Chi-Tsun Cheng *, Toh Yen Pang and Katrina Neville

School of Engineering, STEM College, RMIT University, Melbourne, VIC 3000, Australia
* Correspondence: ben.cheng@rmit.edu.au

Abstract: While traditional industrial robots participate in repetitive manufacturing processes from behind caged safety enclosures, collaborative robots (cobots) offer a highly flexible and human-interactive solution to manufacturing automation. Rather than operating from within cages, safety features such as force and proximity sensors and programmed protection zones allow cobots to work safely, close to human workers. Cobots can be configured to either stop or slow their motion if they come in contact with a human or obstacle or enter a protection zone, which may be a high pedestrian traffic area. In this way, a task can be divided into sub-processes allocated to the cobot or the human based on suitability, capability or human preference. The flexible nature of the cobot makes it ideal for low-volume, 'just-in-time' manufacturing; however, this requires frequent reprogramming of the cobot to adapt to the dynamic processes. This paper reviews relevant cobot programming and control methods currently used in the manufacturing industry and alternative solutions proposed in the literature published from 2018 to 2023. The paper aims to (1) study the features and characteristics of existing cobot programming and control methods and those proposed in the literature, (2) compare the complexity of the task that the cobot is to perform with the skills needed to program it, (3) determine who is the ideal person to perform the programming role, and (4) assess whether the cobot programming and control methods are suited to that person's skillset or if another solution is needed. The study is presented as a guide for potential adopters of cobots for manufacturing and a reference for further research.

Keywords: cobot; collaborative robot; programming; control; skills; task complexity; teach pendant; manufacturing

Citation: George, P.; Cheng, C.-T.; Pang, T.Y.; Neville, K. Task Complexity and the Skills Dilemma in the Programming and Control of Collaborative Robots for Manufacturing. *Appl. Sci.* **2023**, *13*, 4635. https://doi.org/10.3390/app13074635

Academic Editors: Zoran Jurković, David Ištoković and Janez Gotlih

Received: 3 March 2023
Revised: 4 April 2023
Accepted: 4 April 2023
Published: 6 April 2023

1. Introduction

Cobots are an integral part of modern manufacturing, destined to become the leading form of robotics technology in the future [1]. While most mechanical equipment currently used in manufacturing is mono-functional, cobots can assist with a wide variety of tasks, including some standard operations listed in Table 1, along with many more custom applications. As systems incorporate self-learning capabilities of artificial intelligence and auto-correction [2] into cobot operations, the subsequent autonomous behaviour can provide unprecedented collaborative assistance to human workers [3,4]. Evolving from traditional industrial robots [5], cobots have additional programming requirements to feed a collaborative functionality, as shown in Table 2.

Table 1. Typical cobot tasks (Adapted from [6]).

Cobot Task	General Application Description
Welding	Joining of metal parts, typically with a MIG welding (automatic welding wire feed) process
Machining	Precision surfacing process involving a milling or cutting tool

Table 1. *Cont.*

Cobot Task	General Application Description
Deburring	Removal of waste material from casting or machining processes with an abrasive tool
Polishing	Treatment to remove surface irregularities or attain a lustre to coated or machined surfaces
Spray Painting	Part coating, applied to protect or otherwise enhance surface appearance
Sorting	Practical categorisation of unsorted parts for kitting, classification or other organisational operation
Pick & Place	Moving components from a starting point to an endpoint for assembly or other processes
Stacking	Moving finished products from a production line to a pallet or other storage location
Machine Tending	Inserting billets or parts into a milling machine, lathe, etc. and retrieving machined or processed components
Inspection/Measurement	Analysis or quality assurance process where parts and other sub-assemblies or components are measured with sensors or by other means to ensure they are within an acceptable tolerance range

Table 2. Comparison between industrial robot and cobot capabilities [7–9].

Feature	Traditional Industrial Robot	Collaborative Robot
Engagement with humans	Segregated. Operates within a protective barrier, away from humans	Interactive. Operates collaboratively with humans
Safety near humans	Not safe. Must work separately	Safe to work with humans
Environmental awareness	Cannot dynamically adapt behaviour	Can adapt to changes in the environment
Programming flexibility and complexity	Fixed, rudimentary use case programs. Typically reprogrammed infrequently	Flexible, customised use case programs. Potentially reprogrammed frequently
Ease of implementation	Arduous. Requires fixed infrastructure	Fast set-up and easy deployment
Operational behaviour	Fast and repetitious, mainly task focused	Slow and varied. Focus on task and environment
Footprint/Portability	Large footprint, fixed location	Small footprint, mobile
Purchase cost	Relatively expensive to purchase	Relatively inexpensive to purchase
Profitability	Needs medium to large-volume production	Profitable at low-volume production
Investment prospect	Slow return on investment	Fast return on investment

Traditional industrial robots were designed to perform simple, repetitious tasks (refer to Table 2), typically operating within human protective enclosures [10]. These tasks often form part of a mass production process, where the robot is engaged in the same task for long periods [7]. As such, industrial robots are typically programmed infrequently. In contrast, cobots were designed primarily to assist humans with, among other things, low-volume, customised production [2,11]. Due to the constantly changing task requirements, cobots are typically reprogrammed on a more regular basis [12] as a consequence of their operational flexibility.

Identifying the need for a simplified cobot programming method, Dmytriyev et al. [12] proposed the implementation of a flowchart-based programming environment. El Zaatari et al. [13] found that to increase cobot autonomy, the complexity of the industrial application and the worker's knowledge of the task should be considered when choosing programming features. Schou et al. [14] presented a task-level programming solution using a skill-based system, with the instruction of cobots by operators with no previous robotic experience. The skills, in this context, refer to generic controls related to cobot capabilities rather than operator skills. To narrow the skills gap between robotics experts and workers who lack programming expertise, a system using expert frames, which focus on specific cobot operational aspects, was proposed. One of these frames relates to program quality and identifies syntax errors, unused code and missing parameters or code segments. An interactive interface allows operators to visualise a cobot operation and modify behaviour in response to feedback data received [15].

Understanding the contributing elements would be beneficial to systematically analyse the effectiveness of existing cobot programming and control methods. However, reviews on foundation data, including existing and proposed cobot programming and control methods in the literature, their intended users and the types of programmable cobot tasks, appear deficient. Furthermore, research on the relationship between cobot programming skills requirements and task complexity is lacking. These are the research gap questions this study intends to address:

1. What are the common programming and control methods for cobots in the existing market and literature?
2. What bearing does the complexity of a cobot task have on the skills required to program the cobot to perform that task?
3. How effective are cobot programming and control methods in the existing market and literature? Who is the most appropriate worker to implement the methods, and have the methods been suitably developed for that worker?

2. Technical Readiness for Cobot Deployment

Industry perspectives showed that the most significant cobot adoption hurdle is a deficiency among their staff in the knowledge required to program and interact with a cobot [14,16]. Within the knowledge gap, understanding general cobot technologies, application methods, and practical programming is prominent [17]. Employees with prior knowledge of robot programming are considered a key asset in robotic task allocation, ostensibly due to the wide variety of tasks cobots are capable of performing, such as those listed in Table 3. However, the programming of these tasks was more complex than practitioners expected [18]. This apparent misconception about the ease of cobot programming may well be generated from the marketing promises presented by the cobot vendors in general. This premise will be analysed in Section 5 of this paper. Augmenting the programming skills problem, a fear of programming and technology in general among potential operators has been identified as a possible barrier to cobot adoption [19].

Although there is a significant role for cobots to play in low-volume manufacturing [2,11,20], the time spent programming cobots for small batch runs is often not economically viable [21]. This suggests that developing a simpler and more efficient programming method would significantly increase cobot utilisation, especially within the varied tasks associated with low-volume manufacturing operations.

Traditional industrial robots have typically been programmed by dedicated engineers, who may be located off-site and have limited operational knowledge of the task being performed by the robot [14]. Due to the smaller job runs and requirement for faster and more frequent reprogramming of cobots, it may be more efficient to shift the focus from the skills required to *program* a cobot to the knowledge of the *task* a cobot is carrying out [7]. There seems to be a disparity in programming a cobot to perform a particular task. In most practical scenarios, it is often the person who knows the task well who is not competent with programming, and the person who is competent with programming does not know the task well [22]. A possible solution is that the person with knowledge of the task could also program the cobot [23,24]. Operators with task expertise provide substantial improvements in the precision and efficiency of cobot operations when compared to non-experts [22]. An aim of adaptive task-sharing design principles is that cobot programming should be part of the workers' and assembly planners' duties [25]. Substituting a program engineer for a worker skilled with the task being conducted but without cobot technical skills would require a simplified programming tool.

Table 3. Cobot tasks and related operations (Adapted from [6]).

Cobot Task Category	Related Cobot Operations
Assembly	Screwdriving, Part Insertion, Pick and Place
Dispensing	Gluing, Sealing, Lubricating, Painting, Coating, Dipping
Finishing	Sanding, Polishing
Material Handling	Packaging, Palletising, Bin Picking, Kitting
Material Removal	Grinding, Deburring, Trimming, Milling, Routing, Drilling
Welding	Mig, Soldering
Quality Inspection	Testing, Inspecting, Measuring
Machine Tending	CNC, Injection Moulding, Automated Machining

3. Existing Cobot Programming and Control Methodologies

Depending on the complexity of the task to be undertaken by the cobot, a range of proprietary and third-party solutions are available. For more straightforward cobot tasks, such as the first and second tasks listed in Table 4, teach pendant (also known as *Lead through programming*), teach or program by demonstration, and offline/simulation programming are commonly available options [23,26]. Figure 1 shows the teach pendants produced by ABB and Universal Robots. A variety of other programming options are provided by cobot vendors, as detailed in Table 5.

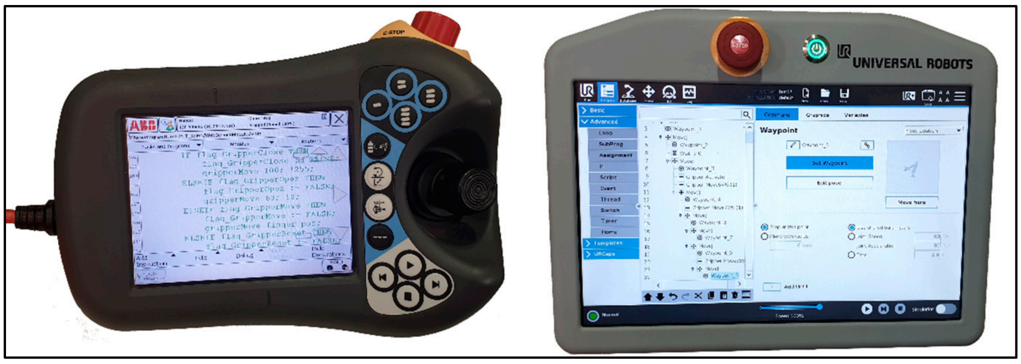


Figure 1. Teach pendants from ABB (**left**) and Universal Robots (**right**) [Image by authors].

For more complex tasks, such as the last two levels in Table 4, there are proprietary scripts, graphical user interface (GUI) programming applications, and the option of programming in a traditional programming language such as Python, C, C++, C# or Java, or a dedicated robotics language such as Robot Operating System (ROS). ABB's RAPID [27] and Universal Robot's URScript [28] are two typical examples of this category, which require a relatively higher level of programming skill. The collaborative nature of a cobot allows it to perform a greater range of tasks compared with an industrial robot, as shown in Table 2. Many of the programming methods developed for industrial robots could restrict the range of operations and be less adaptable if applied directly to a cobot [29]. Because of this programming rigidity, legacy programming methods may be less intuitive than those developed specifically for cobots.

3.1. Programming of Industrial and Collaborative Robots

Traditional industrial robots have fulfilled manufacturers' automation needs since their introduction in the 1950s [5]. Relieving human workers from repetitious and laborious tasks, these robots have been segregated from humans for safety reasons. Resolutely devoted to its assigned task, the industrial robot blindly follows assigned orders, unable to accommodate sudden changes in circumstances. Requiring relatively infrequent reprogramming, tasks assigned to industrial robots are typically simple, repetitive operations designed to be performed from a fixed location [29]. While some industrial robots are mounted on sliding rails or mobile platforms to increase their mobility, most are stationary because of their bulk and requirement to be housed within protective cages [30]. Their ability to operate at high speed allows them to excel at high-volume manufacturing in the mass production era [31].

A cobot is designed to perform precision tasks at relatively slower speeds while demonstrating high flexibility and interactiveness and allowing a much more diverse range of operational tasks than industrial robots [7]. In summary, both industrial and collaborative robots are practical but have different roles to play in the industry. Directly comparing attributes between the two, such as speed, without considering these different roles [32] can potentially result in misguided conclusions. If applied to cobots, similar programming techniques for industrial robots may not make sufficient allowance for the differences between the two, which are comparably outlined in Table 2.

As modern technology progresses, many manufactured products become more complex, as do their manufacturing processes [33]. Today's intricate manufacturing assemblies and processes are generally beyond the scope of industrial robots, especially where part of the workload must be shared with human workers. With a more compact and streamlined form and the addition of sensors and other safety features, cobots can operate outside protective cages and work safely and collaboratively with humans. This allows a level of automated task flexibility and functional expansion, opening up new opportunities for product diversity in manufacturing [34]. As a result, a much wider variety of low-volume, on-demand product manufacturing is now possible, which is the embodiment of agile manufacturing [7,35]. Accompanying this more flexible and adaptive capability, however, is the need for more frequent reprogramming of cobots [7].

3.2. Cobot Task Complexity

Cobots are one of the most utilised machines within manufacturing automation, capable of implementing a broad range of application tasks [11]. The level of complexity among tasks may vary according to the type of task and the object or workforce being manipulated. For example, a sanding task on an object with a flat surface may be considered less complex than if the object was of a complicated design. Similarly, screwdriving, parts insertion and assembly may be deemed complex tasks due to the fine movements and precision required. Still, the level of complexity may vary depending on the configuration, degree of precision and other specific requirements of that task. Universal Robots has defined a common range of cobot task categories [6], outlined in Table 3. Some examples of tasks for each category are also provided in the table.

Cobots are expected to autonomously manage task complexity, especially as it increases beyond human capacity during collaborative operations [13]. The execution of less complex tasks is often considered an automated process, while for tasks of higher complexities, there is an expectation that they will be conducted autonomously by robots [36]. Furthermore, task complexity can be reduced by sharing the task among multiple cobots [37] or by task reduction, cooperatively solved by multiple robots [38]. For instance, some researchers have explored different ways of coordinating multiple cobots' movements and actions [39], while others have focused on developing new algorithms to improve communication and cooperation among cobots [40]. This paper focuses on the complexity of the task overall to be undertaken by a cobot or cobots with respect to the programming effort required. While multiple-robot systems (MRS) can reduce the overall complexity of a task, their need for collaboration and accurate coordination [41] can add significantly to the programming workload. Further compounding the technical complexity of MRS is the need to select a suitable task allocation strategy that is robust, scalable and can be optimised for a specific task [42]. A cluster-based approach, centred around the location of tasks relative to team members, allocates tasks by training a binary classifier to nominate one of two task allocation mechanisms through an auction bidding process [43]. Another model, using k-means clustering, works to solve the balanced multi-robot task allocation problem by minimising travel distance while optimising utilisation, which relates to task completion time [44]. Intended for use within a Cyber-Physical System such as a warehouse management system, task execution is asynchronously assigned to multiple robots in an ordering mechanism to allocate interdependent Human–Robot Collaboration (HRC) tasks [45]. The model uses mutual exclusion to allocate tasks, dynamically promoting system accuracy and robustness. A *Fuzzy Logic System* can coordinate multiple robots to fulfil a common task. Robots, trained using a *Genetic Fuzzy System*, derive a functional strategy to jointly execute a nominated task [46]. MRS that rely on communication mechanisms, such as decentralised planning algorithms, need complex programs to manage alignment coordination between interacting agents accurately [47].

Adapted from the Universal Robots task definitions, an empirical list of task complexity levels, with associated indicative category examples, is presented in Table 4. The levels have been selected based on the amount of robotic and programming effort required to perform a typical task at the proposed complexity level rather than on the computational resources consumed by a robot during the execution of the task [38]. The degrees of effort have been determined empirically based on a range of programming tests conducted primarily with a UR5e, a cobot representative of the Universal Robots eSeries cobot range [48]. Robotic input refers to the range of manipulator and end-effector actions per task, including:

- Type of joint movement (joint or linear motion)
- Number of movements
- Number of waypoints
- Number of end-effector engagements
- Degree of positional precision required
- Sensor input required (force, proximity, gyroscopic, etc.)
- Ease of end-effector to grip (owing to the surface or shape of parts, etc.)
- Payload handling

Table 4. Task complexity levels.

Task Complexity Level	Types of Associated Tasks
1 (Low)	Simple Gluing, Sealing, Dipping, Sanding, Polishing
2 (Low–medium)	Pick and Place, Lubricating, Painting, Coating, Injection Moulding
3 (Medium)	Material Handling, Simple Assembly, Grinding, Deburring, Trimming, Drilling, Screwdriving
4 (Medium–high)	Parts Insertion, CNC, Automated Machining
5 (High)	Complex Assembly, Precise Milling, Routing, Quality Inspection

Programming input refers to the quantity and complexity of the programming elements required per task, as discussed in Section 5.3. Within the *types of associated tasks*, each task can vary significantly in complexity (see Table 4). For example, while a sanding task conducted on a single flat surface could be considered a task with low complexity, the precision sanding of a spherical object may well place the task at a higher level. It is, therefore, impossible to be definitive with the assignment of the input levels, nor is it the intention within the scope of this study. Instead, the objective is to assign input levels based on the task complexity levels presented, with the *types of associated tasks* being indicative of those levels.

3.3. Task Allocation in Human–Robot Collaboration

A vital step in the optimisation of Human–Robot Collaboration (HRC) is the assignment of tasks to individual collaborators (agents) to maximise the efficient utilisation of the cobot [49]. Within a manufacturing operation, sub-processes can be instinctively allocated to the cobot or the human, based on considerations such as suitability, capability, or human preference. To increase efficiency in a collaborative production environment, a systematic resolution process must be applied to determine which tasks should be conducted by the cobot and which tasks would be best left to a human [50,51]. The individual skillsets of both the cobot and human involved are considered. Tasks are dynamically assigned in a skill-based HRC system, which provides a graphical programming interface and pre-programmed macros to simplify the cobot programming operation [52]. A task allocation model, which maps task characteristics to agent capability, was proposed by [53] and exploited human operators' adaptability and cognitive prowess, along with cobots' efficiency, accuracy, and consistency. Designed to manage tasks allocated to a human and two robots in a heavy part handling HRC assembly operation, a solver based on the *Genetic Algorithm* [54] is used to optimise both operation time and selection based on agent capability. Considering human contentment in HRC, a two-staged capability-based task allocation process was proposed [55]. In the first stage, task elements that align with human capabilities are identified to assess whether humans can perform these safely or are automated to preserve human safety. The remaining tasks are then allocated according to suitability in the second stage based on agent capability, time efficiency, cost, and quality outcomes. Allocation of tasks in HRC assembly operations is based on the classification of the task complexity level and assigned to humans or robots in a complexity-based task allocation method [56]. Agent skills, along with environmental aspects such as component properties, presentation and feeding, are considered in the selection process, which affects efficient handling in an assembly environment. A hierarchical HRC framework proposed by [57] separates task allocation into abstraction and allocation layers in an assembly environment. The abstraction layer defines the planning phase of the collaborative assembly, considering the specific attributes of each element, including human and robot agents and the real-time communication capabilities of each, while the allocation layer manages the skills implementation requirements for the task.

An action scheduling framework using Artificial Intelligence (AI) contains a scheduling algorithm that generates task allocations, which was proposed by [58]. The system considers the sequence of operations, tools required, resource availability, positioning and time efficiency in its optimisation process. Within this human-centric framework, there is a focus on the safety of human operators, and operations are blocked if prerequisite operations have not been completed. Task completion time rewards motivate a *Markov game* model developed by [59], using deep multi-agent reinforcement learning to determine the optimal cobot task allocation schedule. This model evaluates cobot and human agent availabilities and determines the optimal task scheduling policy for the operation within a chessboard structure that minimises all assembly task completion times as its objective.

3.4. Leading Cobot Vendors' Programming and Control Methodologies

To provide readers with an overview of the landscape of existing cobot programming and control methodologies, Table 5 lists seven leading cobot vendors and details of their programming and control solutions. A selection of generic programming applications, which provide an alternative to many proprietary developments, are also included.

Table 5. Programming methods for leading cobot vendors.

Cobot Manufacturer	Programming Interface/Type	Programming Method	Intended User Level	Intended Task Complexity Limit	Source Language/Related Language (If Known)
Universal Robots [28,60,61]	PolyScope	GUI/Hand guide	Unskilled	Low	Proprietary (URP)
	URScript	Textual (Script)	Skilled	High	Python, C++, C#, VB, Java
	URSim	GUI/Simulation	Unskilled	Low	Proprietary (URP)
ABB [62–64]	Wizard	GUI/Hand guide	Unskilled	Low	Block-based RAPID
	FlexPendant	GUI/Simulation	Semi-skilled	Medium	
	RAPID	Textual	Skilled	High	Visual Basic
	RobotStudio	GUI/Simulation	Semi-skilled	Medium to High	Proprietary (RAPID)
Fanuc [65]	TP (Teach Pendant)	GUI/Hand guide	Unskilled	Low	Proprietary (TP)
	Karel	Textual	Skilled	High	Pascal
	Roboguide	GUI/Simulation	Semi-skilled	Medium to High	
Festo [66]	BionicCobot	GUI	Unskilled	Low to Medium	ROS
	Festo Robotic Suite	GUI	Semi-skilled	Medium	Python/ROS
	RoboCIM	GUI/Simulation	Semi-skilled	Medium	Proprietary
Kuka [67]	iiOKA OS	GUI/Hand guide	Unskilled	Low	Linux kernel
	KUKA Work.Visual	GUI/Textual	Skilled	High	
	KRL (KUKA Robot Language)	GUI/Textual	Skilled	High	Pascal
	Kuka Sunrise	Textual	Skilled	High	Java
	Kuka Sim/SimPro	GUI/Simulation	Semi-skilled	Medium to High	Proprietary
Yaskawa [68–70]	Direct Teach (DT)	Hand guide	Unskilled	Low	Proprietary
	Smart Pendant	GUI/Hand guide	Unskilled	Low	Proprietary
	INFORM II	GUI/Textual	Skilled	High	C
Omron [71]	ACE (Automatic Control Environment)	GUI/Hand guide	Unskilled	Low	C#
	eV+	Textual	Skilled	High	MS-DOS/Unix Script
	ACE Emulation Mode	GUI/Simulation	Unskilled	Low	C#
Generic cobot programming applications (RoboDK, ArtiMinds, Wandelbot, Pickit, Robomaster, G-Code, ROS, Traditional languages)	RoboDK GUI [72]	GUI/Hand guide	Unskilled	Low	RDK
	RoboDK API [72]	Textual	Skilled	Medium–High	Python (default), C, C++, . . .
	ArtiMinds RPS [73]	Modular	Semi-skilled	Medium–High	Proprietary (RPS)
	ArtiMinds RPS [73]	Modular	Semi-skilled	Medium–High	Proprietary (RPS)
	Wandelbot Tracepen [74]	Input device/App	Unskilled	Medium–High	Proprietary
	Pickit robot vision system [75]	GUI/3D vision	Semi-skilled	Medium	
	Robotmaster [76]	GUI/Simulation	Semi-skilled	Medium–High	
	G-Code [77]	(CAD/CAM)/Textual	Skilled (CNC)	Medium–High	G-Code
	ROS/ROS2 [78]	Textual	Skilled	High	C++/Python
	Traditional prog languages: Python, C, C++, C#, Java, [79]	Textual	Skilled	High	Proprietary

The *'Intended User Level'* and *'Intended Task Complexity Limit'* fields in Table 5 are broadly based on cobot vendors' statements, including those in Section 4.1 of this paper, and practical reviews of the various applications. The aforementioned fields will be the subject of the comparative analysis presented in Section 6. In addition, *Intended User Level* references the general skill level definitions stated in [80], with unskilled work requiring little to no independent judgement or previous experience to perform simple tasks; semi-skilled may require close attention and coordination abilities for tasks with some complexity, with decisions made by others; and skilled work, which requires judgement and decision-making abilities while performing more complex tasks.

3.5. Controlling Cobots without Formal Programming

Recognising the need for frequent reprogramming and flexible, collaborative operation, cobot vendors compete to find the most intuitive and efficient method of controlling their cobots.

3.5.1. Existing Cobot Control Methods

A common easy-to-navigate control method offered by cobot vendors is 'Guiding' (sometimes referred to as the Teach method/Easy programming/Basic programming), which involves human manipulation to 'teach' the cobot the required sequence of operation. Typically, user interfaces combine a pendant, tablet or PC-based display with the user positionally hand guiding the cobot, as shown in Figure 2.

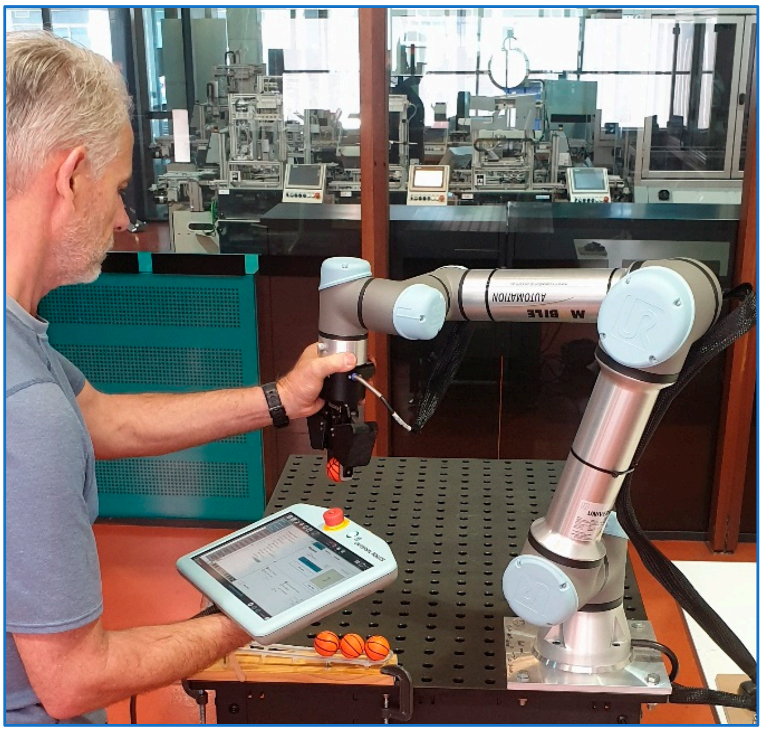


Figure 2. Programming UR5e with the teach pendant and hand guiding by a human operator.

3.5.2. Proposed Cobot Control Methods in Recent Literature

The benefits of assigning cobots to small-batch agile production [81] can be eroded by the requirement for more frequent reprogramming [7], especially if there is a need to continuously hire skilled programmers [12]. However, by simplifying programming and control methods sufficiently, workplace operators, rather than dedicated programmers, could intuitively program the cobots [82]. In this case, there would be greater potential for increased cobot utilisation since the programming role would be open to a broader range of workers.

Skill Based System

Since cobots are designed to interact closely with human workers, who have extensive knowledge of the task being undertaken, programming the cobots by the same workers would seem to be a logical part of the collaboration. With an emphasis on the task rather than the program, a Skill Based System (SBS) allows a robotic novice worker to use a graphical interface to map skills relating to a task, which equates to individual robotic actions. As shown in Figure 3, these skill entities are used as parameters to initiate the kinaesthetic configuration of a cobot's joints [14]. For some prospective programmers, however, the first step will be to overcome their fears of programming and technology and accept cobots as collaborative partners rather than their future replacement [19].

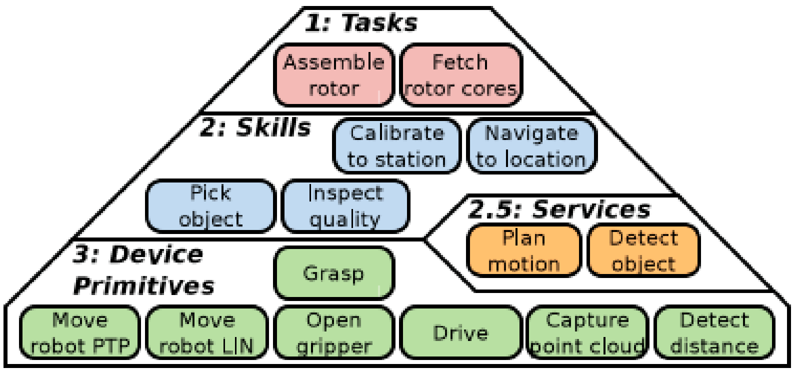
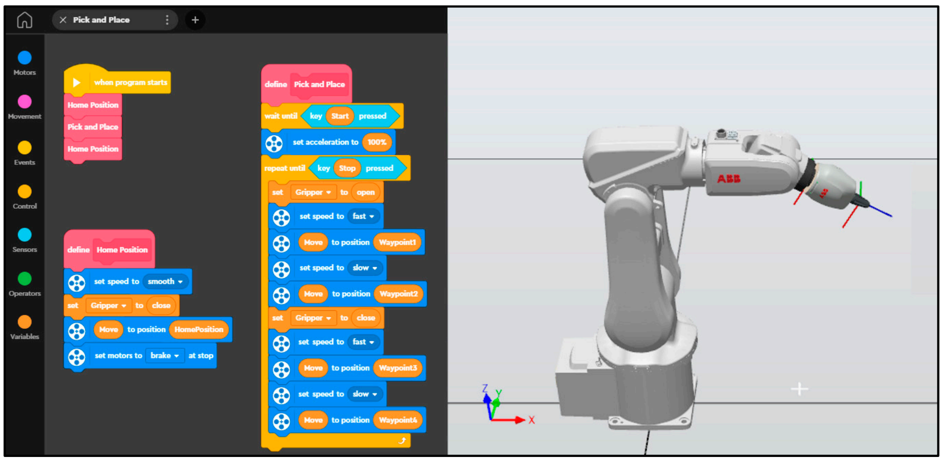


Figure 3. An example of a skills-based system with 3 abstraction layers is used to configure cobot joint settings [14] (by permission).

Block-Based Interface

One approach in the quest for a more comprehensible programming experience is a block-based interface, where pre-programmed modules can be selected and placed in a desired sequence by novice users, as shown in Figure 4 [83]. This type of drag-and-drop interface has two key advantages over text-based programs. Firstly, the blocks, written in simple language, are designed to be intuitively arranged by a novice while providing visual cues as the program develops. This allows the user to focus on the task's workflow rather than the program's syntax, which relates to the second advantage. Each block can be considered a pre-programmed function, which is syntactically correct and, along with the other blocks in a sequence, collectively compilable [84].



```
when program starts
Home Position
Pick and Place
Home Position

define Home Position
set speed to smooth
set Gripper to close
Move to position HomePosition
set motors to brake at stop

define Pick and Place
wait until key Start pressed
set acceleration to 100%
repeat until key Stop pressed
    set Gripper to open
    set speed to fast
    Move to position Waypoint1
    set speed to slow
    Move to position Waypoint2
    set Gripper to close
    set speed to fast
    Move to position Waypoint3
    set speed to slow
    Move to position Waypoint4
```

Figure 4. Block-based programming editor with a virtual industrial robot (Adapted from [83]).

Layered Block-Based Interface

A layered approach to block-based programming provides users with different skill levels and the flexibility to carry out programming tasks with a complexity level that matches their ability. From workers with little technical skill generating basic assembly workflows to those experienced in programming and robotics undertaking more complex programming tasks, a three-layered approach caters for a broad range of user abilities and task complexities in a single system (refer to Table 6) [24].

Table 6. A three-layer block-based cobot programming model [24] (by permission).

Programming Layer	Roles and expertise	Required Training	Support Techniques
Layer 1: Basic assembly workflows (robot movements and tool actuation)	Assembly workers and laypersons with some assembly experience	Some technical training (e.g., professional school)	Mounting and assembly devices, multimodal teach-in
Layer 2: Block-based programs (task blocks, variables and control structures)	Industrial engineers with computational thinking abilities and technical intuition	Formal technical training and a programming course	CAD-modelling, 3D printing, laser cutting
Layer 3: Advanced functionality (databases, connectivity, etc.)	Software engineers with advanced programming skills	Formal software engineering training	Internet/intranet, databases, cloud, Manufacturing Execution System (MES)

Chat-Assisted Block-Based Interface

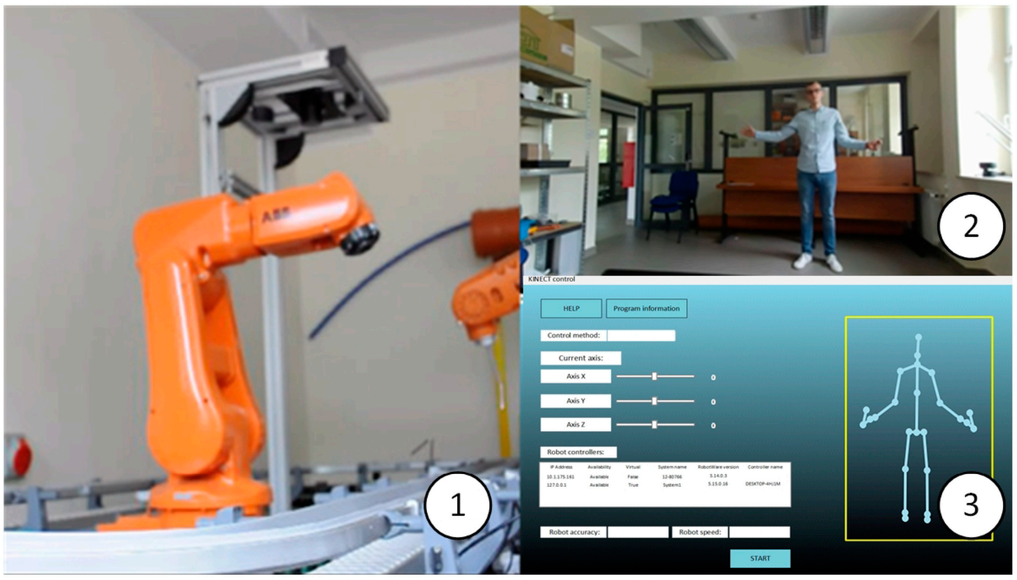
By enhancing block-based programming with natural language chat functionalities, non-technical users can more intuitively assign programmed tasks to cobots. Furthermore, this flexible hybrid approach can allow users to implement more complex tasks more easily than block-based programming alone while building user acceptance of the technology and confidence with cobot programming over time [85].

Artificial Intelligence

Artificial Intelligence (AI) can also assist a programmer with specific language code suggestions offered in real-time, generated through the Natural Language Processing (NLP) functionality of OpenAI Codex [86]. However, programming expertise would be required for each suggestion as part of the accept/reject/edit decision-making process. Developing autonomous cobots motivates many researchers, and they are exploring specific areas of AI that are more suited to efficient cobot management. For instance, *Reinforcement Learning* has been effectively used to train robots to conduct complex tasks; while modifying their behaviour in response to changes in their environments, which they continuously monitor. In contrast, Deep Learning (DL) could restrict dynamic response behaviour due to time-inefficient data processing requirements [87]. According to [87], DL may not be suitable for differentiating objects of similar appearance, which may be the case with some electronic components used in assembly processes. In addition, DL is a complex area of AI that requires significant experience and skills to implement [88], a notion that opposes the objective of the current study. To improve cobot adaptivity and intelligence in HRC, an AI-based 3D perception system, which incorporates *RGB* colour scan cameras to generate 3D representations of a cobot's operational environment, contains an anti-collision function and allows a human operator to control cobot movement with natural gestures [89]. Furthermore, a path planning architecture establishes a predetermined path for *Unmanned Aerial Vehicle* robots, using the *Hungarian algorithm* to optimise cost efficiencies. A 3D occupancy map of the cobot's operational environment is generated to ensure a collision-free path [90].

Voice and Gesture Control

A more instinctive cobot control measure that alleviates the need to formally program allows a user to guide a cobot through a task using voice commands and physical gestures. An image and sound processor comprises a camera, a depth sensor (for human movement tracking) and a set of microphones that capture the relevant human speech and gesture inputs. These are converted in the software program into corresponding coordinates for the cobot joints [91]. Figure 5 shows the testing station for the speech and gesture system, depicting the operator, graphical interface and ABB IRB120 robot.



IP Address	Availability	Virtual	System name	RobotWare version	Controller name
10.1.175.161	Available	False	12-80766	5.14.0.3	
127.0.0.1	Available	True	System1	5.15.0.16	DESKTOP-4HJ1M

Figure 5. A testing platform for cobot voice and gesture control system, proposed by [91], 1. ABB IRB120 robot, 2. the operator, 3. graphical interface (by permission).

A validation process is used between gestures in a task-managed system to prevent undesired sequences from executing. In their work, users can decide to return to a previous save point of the program by way of a predefined human gesture [92]. In these systems, operators would need to be suitably trained on the range of valid speech and gesture commands.

Virtual Reality Systems

Moving further away from traditional programming and closer to a gaming platform, there is an immersive cobot control method. Controlling a physical cobot from a virtual environment is a concept that replaces a standard cobot interface console with a virtual reality (VR) system, which is connected to the cobot's controller mechanism. From within the virtual environment, a human can control a virtual cobot (a virtual representation of a physical cobot), with the inputs stored in a dedicated database so that the cobot's trajectories can be visualised in real-time or reproduced from stored values [93]. In addition, cobot control includes safety precautions in human–cobot interaction, which can be achieved in a VR environment. Cobot behaviour is dynamically modified to anticipate human movement using a motion tracker [94].

Augmented Reality Systems

Combining the virtual environment with the real world, augmented reality (AR) systems can provide a more realistic human experience than VR systems. AR systems allow a faster and more user-friendly approach for humans to interact with cobots, with users preferring an AR system over traditional teaching pendants or console options [95]. Users with no previous programming experience can perform a range of cobot motion tasks through an AR interface. However, as the complexity of the cobot task increases, so does the difficulty experienced by the user. For example, pick and place tasks are more easily conducted than welding tasks. Overall, the AR experience positively affects the novice user's confidence to accurately and safely carry out the assigned tasks [96].

4. The Cobot Programming and Control Sales Pitch

With the global cobot market expected to exceed USD 9 billion by 2024 [9], competition is fierce among cobot vendors as they vie for dominance in this lucrative industry. One of the cobot vendors' prime objectives is to simplify the human–cobot interaction method.

Furthermore, to alleviate the need for robotic or formal programming skills, cobot vendors attempt to present users with the most intuitive interface to control the cobot.

4.1. Cobot Vendors' Perspectives on Their Programming Applications

In this section, a selection of claims and statements in the leading cobot vendors' product description material has been quoted on the program and control attributes of their cobot interfaces aimed at untrained users (refer to Table 7). The claims made by the cobot vendors will be used to estimate programming effectiveness and intuitiveness from the perspective of the intended user.

Table 7. Cobot vendors' marketing claims.

1. Universal Robots (PolyScope) [60]
"Use your process expertise and PolyScope's graphical interface to create a robust automation system. No Code? No Problem." "PolyScope connects operators to robots for efficient and productive automation. You don't need coding experience to automate your processes." "Build programs by selecting nodes from a menu and placing them in order of operation. Each node represents an instruction for the robot and its parameters can be configured."
2. ABB (Wizard) [62]
"Wizard easy programming—An easy and intuitive way to program cobots and Industrial robots." "Program your robot application within minutes! Wizard is an easy graphical programming interface for ABB Cobots." "With Wizard, anyone and everyone can program their robot application." "Only a few minutes after the installation you will be able to operate your robot. With Wizard's easy drag and drop blockly based programming software, no specialized training or programming skills are required."
3. Fanuc (TP) [65]
"But the CRX's #1 "Ease of Use" benefit is its Simple Drag and Drop Programming. 30 years ago, robot programming was a high-level structured language like Fortran or C++. An engineer or maintenance technician went to school to learn how to learn the programming language. Both are very powerful, but not easy to use. With the all new FANUC CRX Tablet Teach Pendant–simple programming becomes reality. Easily program and teach points with the CRX Tablet Teach Pendant. The drag and drop interface for lead-through teaching and simple programming is easy with no prior robotic experience needed."
4. Festo (Festo Robotic Suite for BionicCobot) [66]
"Programming a robot is child's play" "The BionicCobot is operated intuitively via a graphic user interface developed in-house." "Commissioning and programming are intuitive, quick and easy with the "Robotic Suite" software." "The developers were focusing specifically on making the Robotic Suite, the actual heart of the cobot, as simple to operate as possible so there is no need for prior programming knowledge." "When it comes to programming, most of us probably think about complicated lines of code with lots of abbreviations, brackets and other symbols. But programming a robot can actually be very easy, as is shown by the software that Festo developed for its pneumatic lightweight robot, the BionicCobot."
5. Kuka (iiQKA OS) [67]
"iiQKA allows you to put together your individual automation package, without any prior knowledge or programming experience." "The intuitive graphical interface allows for fully autonomous control of the system without any programming knowledge." "Designed for quick start-up with little to no expertise, iiQKA offers incredible speed to integrate robots."

Table 7. *Cont.*

6. Yaskawa (DT) [68,69]
"Quick and easy programming." "Simple and intuitive operator control, short learning curve." "Ideal for users who need to carry out frequent reprogramming and thus appreciate simple operator control." "Ideal for novice robot users, this pendant simplifies INFORM programming for easy-to-understand operation and fast implementation of the robot system." "The perfect entry into programming. Simply move the robot flange by hand, record the motion points and operate the gripper actuation by pressing the respective DT buttons. The code is automatically generated in the background on your pendant."
7. Omcron (ACE) [71]
"The Automation Control Environment (ACE) software allows you to build applications, such as Pack Manager packaging applications, which can be basic pick-and-place cells or complex cells with multiple cameras, conveyors, and robots. You can create and configure these cells without having to write any programming code. For applications that require greater control, you can override the default V+ program code and make changes as needed."

4.2. Epitomising the Marketing Message

Regarding user interaction with their products, the general message from cobot vendors is that neither robotics nor programming experience is required, at least for the less complicated cobot tasks. However, the complexity level is not explicitly referred to in their marketing blurbs. Furthermore, many instructional videos released by cobot vendors [97,98] tend to emphasise the simplicity of their user interfaces, while limitations, in terms of task complexity, have generally been concealed. Marketing cobots as "simple to program" can lead to the assumption that skilled programmers are no longer required [20]. This assumption makes no allowance for the complexity of the task to be programmed, nor does it consider the variation in the skillset required to do so. Despite cobot vendors' assurances that unskilled personnel can program their cobots, others have a different perspective. A leading cobot tool manufacturer views cobot programming as a job for an engineer with mechanical, electronics, electrical and programming skills and an understanding of combined system functionality and associated theory [99]. Manufacturers in the industry, who operated cobots, revealed that of the total time employees spent with their cobots, automation engineers accounted for 48%. Approximately half of the automation engineers' cobot duties were dedicated to programming [100]. In Section 6, the cobot vendors' claims will be compared with the observed skills required to program a cobot, to establish the validity of their claims and the effectiveness and suitability of the programs for their intended purposes and at their intended user levels.

5. The Underlying Skillset

There are three key elements involved in the process of cobot task allocation [101].

I. The cobot and its environment
II. Cobot programming steps
III. Cobot task implementation

For a human tasked with implementing the process, these elements correspond to the knowledge and skills that must be acquired or provided extrinsically. Regarding cobots, a conceptual knowledge of the hardware and technical skills in cobot operation and problem-solving would be necessary, along with knowledge of the assembly line or other cobot environments. To program cobots, knowledge of the programming application is essential, along with skills in programming within the context and capabilities of a specific cobot. Finally, knowledge of the task the cobot is to perform, optimisation characteristics, and the skills to carry out the steps of the task would typically be required [101]. An aim of adaptive task-sharing design principles is that cobot programming should be part of the workers' and assembly planners' duties [25]. Substituting a program engineer for a worker skilled with the task being conducted, but without cobot technical skills, would require a simplified programming tool.

5.1. Fundamental Technical Aspects of Cobot Control

To highlight the complexities potentially experienced by the user of a typical cobot programming interface, robotics and programming concepts and keywords are presented. Table 8 relates to the robotic (hardware) elements, while Table 9 contains the programming (software) elements. The reference source for both tables was the Universal Robots *PolyScope* graphical interface [60], used to program a UR *e*Series cobot [102]. The list is not intended to be exhaustive but rather a selection of representative fields more relevant to the programming than the setting up of a cobot.

It is acknowledged that in Tables 8 and 9, not every listed entity would need to be configured for each program. However, it could be argued that in support of sound judgement, a knowledge of the purpose and functionality of each would be required to determine when they *should* or *should not* be configured. For example, there is a view in the cobot tool manufacturing industry that cobot programmers should have adequate knowledge of advanced programming functions [99], such as those listed in Table 9.

5.2. Robotics (Hardware)

A selection of robotic control elements, along with corresponding robotic entities, is listed in Table 8.

Table 8. Robotic control elements.

Robotic Element	Specific Robotic Entity
General	Fundamental kinematics and dynamics principles
Limits	Safety limits, including power, momentum, stopping time and distance, tool speed and force and elbow speed and force limits
	Joint position range limits and maximum speeds
Orientation and positioning	Tool Centre Point (TCP), Tool Offset, Tool Position and Tool Rotational Vector settings (all represented as three-dimensional cartesian coordinate frames)
	Relationship between base and tool coordinate frames
	Direction for linear movement (expressed as positive or negative cartesian coordinates or direction vector)
	Waypoints (with options of fixed, relative or variable position)
Joints	Joint positions (in degrees) for the base, shoulder, elbow and three wrist joints (pitch, yaw and roll).
	Linear, non-linear and circular joint movements
	Joint speed and acceleration values
Communication	I/O Signals (Digital, Analog, Tool, Configurable, Boolean Register, Integer Register, Float Register)

5.3. Programming (Software)

A selection of programming elements, along with corresponding cobot programming entities, are displayed in Table 9.

5.4. Skills Required to Program and Control a Cobot

An empirical list of the skills required to program and control a cobot has been created to analyse the relationship between these skills and the complexity of the tasks they are used to conduct. Descriptions have been provided for each pair to clarify the assignment of skill values and levels. TACOM [103], a *ta*sk *com*plexity measure, was used to guide the selection of cobot task complexity categories. The metric is based on the quantity of information an operator must process about the task, the number of actions and logical sequence a task contains, current knowledge of the task and available cognitive resources for decision-making. TACOM calculates task complexity with respect to the task's procedural steps. The information density and composition that define a task and the number of actions and order of operation required to execute it affect the complexity level of a task. A certain amount of system knowledge is required to carry out an action and to understand the complications of the task. The capacity of an operator, in terms of the precision and cognitive effort, during the execution of a task, along with the specific resources required by the task, contribute to the performance level of a human operator [104].

Table 9. Cobot programming requirements.

Programming Element	Specific Cobot Programming Entity
General syntax	Program structure and sequencing
Data	Constants, variables, variable assignment
Conditionals/selections	If, ElseIf, Else, Until, Switch statements, Boolean
Program loops/iterations	For, While, Do-While
Functions/procedures	Thread, Subroutine call
Control points	Event, Wait, Set, Halt, Timer

Table 10 outlines a proposed user skill level paradigm. Each skill level designation has a range from a low to a high skill level listed, followed by a suffix, which indicates the skill type (R = Robotics, P = Programming). For example, '1 R' indicates a skill level on the low end of basic robotics knowledge, while '3 R' refers to the high end of the same level. There is a slightly broader range (7–10) in the higher levels of both skill types to accommodate the greater range of advanced technical concepts.

Table 10. User skill levels for the programming and control of cobots.

Skill Level Designation	Description of Skill Level
0: Unskilled	No knowledge of robotic operation or programming
1–3 R: Basic Robotics	Basic knowledge of robotic concepts, such as the difference between collaborative and industrial robots, the joint structure of a manipulator (robotic arm) and basic understanding of end effectors such as a gripper or suction cup
4–6 R: Mid Robotics	Familiar with robotic movement and functionality, including linear and non-linear joint movement, consequences of joint speed and acceleration settings (collision prevention) and coordinate frames
7–10 R: High Robotics	In-depth knowledge of robotics, with practical skills in cobot installation, tool configuration with respect to coordinate frames, precise joint configuration, I/O signals, sensors and configuration of safety elements such as protection zones
1–3 P: Basic Programming	Basic knowledge of programming concepts, such as data types, data inputs, computations and outputs
4–6 P: Mid Programming	Familiar with basic programming techniques involving common elements such as variables, loops and conditionals
7–10 P: High Programming	Competent in structured programming, using functions, different loop and conditional types, switch statements and classes

Although cobot programming methods range from those that are easier to use, for less complex tasks, to those requiring higher skills for more complex tasks, there is some common ground between them. Following some targeted programming comparisons, it was found that some tasks of medium complexity could be programmed at the higher end of a Teach Pendant's capability, for example, or at the lower end of a script-based method, as indicated with the Venn diagram in Figure 6. The choice of method would be subject to the skill level or discretion of the person programming the task. This further complicates mapping task complexity to the corresponding required skill level for tasks within that intersecting zone.

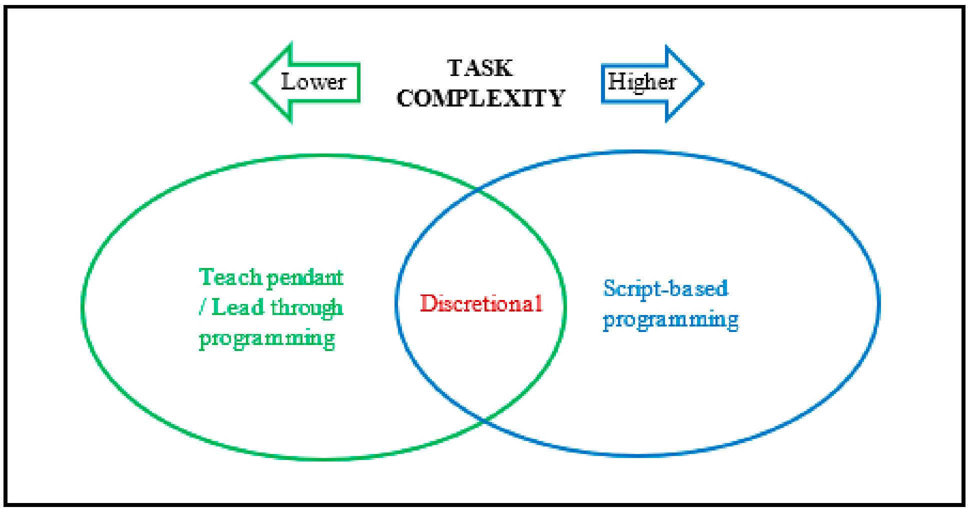


Figure 6. Cobot programming methods according to task complexity.

The extent of the skills and knowledge required will vary based on the complexity of the task being programmed. In addition, the broad range of approaches available to program a specific task can further complicate the job. For example, the kinematic flexibility of a cobot, while allowing great freedom in tool orientation and position, also increases the programming complexity [93]. Considering the robotics and programming skills required, Figure 7 shows a graphical representation of the skills required to program a cobot by different methods based on task complexity.

From a skill requirement perspective, the transition from programming a cobot with a teach pendant to write a script-based program to do so is significant. Using the Universal Robots *URScript* or ABB *RAPID* programming methods as examples, a programmer must

1. Establish a connection to the cobot controller from a remote console. Some knowledge of computer networks would be required to communicate with a host over a socket connection.
2. Compose a syntactically correct control program. At least a moderate level of skill in programming, with an understanding of program structure and syntax, would be required. For this study, *URScript* test programs were written in *Python*, so an understanding of the relevant formal language for the client program is also necessary.
3. Choose:

- Individual cobot joint positional coordinates and orientation parameters
- Motor speed and acceleration settings
- Tool selection
- Delay timing
- Other aspects depend on the complexity of the task being programmed.

Sound knowledge of robotic functionality, particularly about coordinate frames and joint movement, would be mandatory.

5.5. Skills Versus Task Complexity Testing Methodology and Results

A UR5e cobot was programmed to simulate representative tasks from the associated task categories outlined in Table 4. Four tasks with complexity levels ranging from 1 to 8 were programmed with the *Teach Pendant*, while two tasks with complexity levels ranging from 7 to 10 were programmed in *Python* through *URScript*, for comparison. The range of technical skills utilised during the configuration of each programmed task was recorded. These related to the robotic and programming elements, as defined in Tables 8 and 9, respectively, are used to describe the user skill levels outlined in Table 10. The results of the recorded program tasks are shown in Table 11 and represented graphically in Figure 7. The programmed tasks listed in Table 11 were selected to represent each of the task complexity levels in Table 4. The aggregate skill levels in the table are the product of the corresponding robotic and programming skill levels defined in Table 10.

Table 11. Task complexity versus skill level.

Task Complexity Level	Programmed Task	Programming Method	Robotic Skill Level	Programming Skill Level	Aggregate Skill Level
1–2 (Low)	Simple Polishing	Teach Pendant	2	1	2
2–4 (Low–medium)	Simple Pick and Place	Teach Pendant	4	2	8
5–6 (Medium)	Simple Assembly	Teach Pendant	6	4	24
7–8 (Medium–high)	Parts Insertion	Teach Pendant	6	4	24
7–8 (Medium–high)	Parts Insertion	Script-based	9	9	81
9–10 (High)	Complex Assembly	Script-based	10	10	100

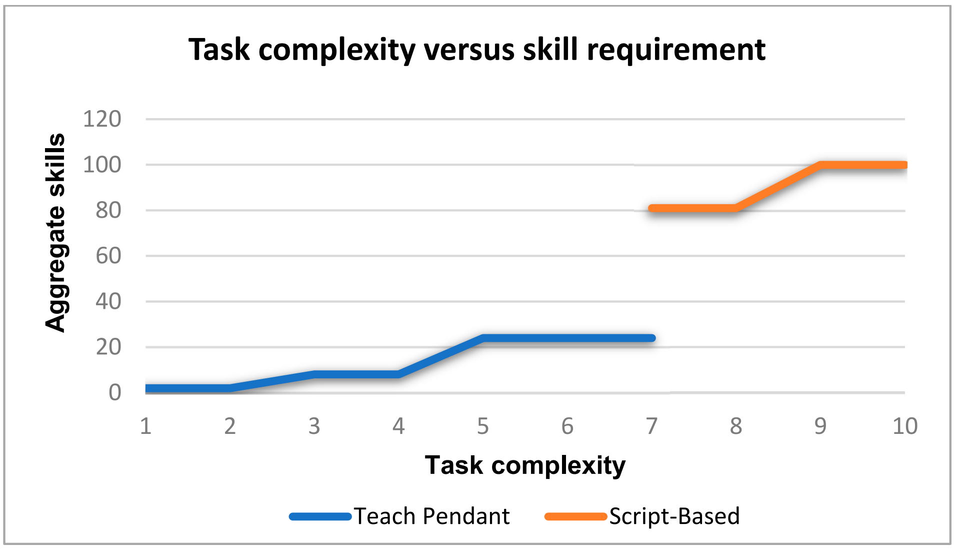


Figure 7. Robotics and programming skills required versus task complexity.

The effectiveness of the teach pendant as a programming tool is demonstrated in Figure 7, with a low level of skills required to program cobot tasks in the lower complexity range. However, from the medium task complexity level, the required skill level is three times higher than the previous one. When using the teach pendant, the required robotics skills are 1.5 times higher than programming skills for all complexity levels. This is because of the greater precision and finer tool positioning typically associated with more complex tasks. At the same time, there is a more gradual increase in skills required with the simplified graphical programming interface of the teach pendant. As task complexity enters the medium to high band, for tasks beyond the capability of the teach pendant, the required skills are 3.3 times higher for script-based programming. Because of the need to precisely define joint coordinates and other cobot parameters within the syntax of a textual programming language, the required robotic and programming skills increase at the same rate. The lack of skills issue has imposed constraints on the programming of complex tasks.

6. Analysis of the Findings of This Review

An analysis was undertaken into the relationship between the complexity of the task to be executed by the cobot and the relative skill levels required of the programmer. Moreover, a mapping of this relationship was established. In the quest to find the most suitable candidate for the cobot programming role, consideration was given to the frequency of cobot reprogramming, knowledge of the task to be performed by the cobot, economic efficiency and organisational logistics. Finally, the existing cobot programming and control methods and those proposed in the literature were evaluated to determine if they were appropriate for the person with the programming role or whether another solution was needed. Section 6.1 evaluates the effectiveness of existing cobot programming and control methodologies, while Section 6.2 presents the practicalities of the main cobot programming and control proposals in the literature. A summary of the findings, including responses to the research questions, is provided in Section 6.3.

6.1. Effectiveness of Existing Cobot Program and Control Methodologies

Cobot tasks programmed with script-based methods require some expertise in formal programming, an understanding of the specific cobot functionality and a sound knowledge of robotics in general. Proprietary Teach Pendants, however, are typically marketed as user-friendly tools designed for programming a cobot without the need for robotic or programming skills. While the graphical interface of the teach pendant is easier to use than a script-based alternative, this study has found that some foundation skills in robotics and programming are still required. Basic programming and robotics knowledge will contribute to a more accurate and safer outcome, even when programming a relatively simple cobot task. Knowledge of coordinate frames, for example, can allow more positional precision of joints and an understanding of efficient joint and linear movements, along with motor speed and acceleration settings, could help with collision avoidance. Furthermore, programming fundamentals such as sequences and loops can add vital insights into program structure and process iteration. When marketing their teach pendants, cobot vendors tend to highlight the ease of use for non-experts without emphasising the extent to which task complexity may complicate the process. The teach pendant, therefore, does require both robotic and programming skills, which increase as the task becomes more complex. It is, however, a significantly more intuitive cobot programming tool than a script-based language, which requires a high level of competency in robotics and formal programming. Some computer networking knowledge may also be required, depending on the method of connection between a terminal and its host.

6.2. The Practicality of the Primary Cobot Program and Control Methodologies Proposed in the Literature

The main cobot programming and control methods proposed in recent literature and reviewed in Section 3.5.2 are critiqued in this section.

6.2.1. Block-Based Interface

While alleviating syntax errors, programming with a block-based graphical interface [83] requires some knowledge of program flow control and an understanding of robotic movement when using it to control a cobot. There is also the potential for runtime errors if, for example, unattainable values have been entered as joint position, speed, or acceleration parameters.

6.2.2. Voice and Gesture Control

Although they present an intuitive approach to controlling cobots, gesture and voice control have their limitations. Gestures can be misinterpreted by the image processor or incorrectly posed by the human operator. An example of where such errors could occur is in the subtle pose differences between *'Axis Y move forward'* and *'Axis Z move forward', where only a slight difference in left arm position separates the two commands*. Voice control is affected by noise variation and also human error in the case of an incorrect command or poor diction [91], which could be challenging in a production environment. Collectively, this method may be difficult for a human to orchestrate, having to coordinate the correct gestures and voice commands, especially when sequencing the cobot through a complex task. In addition, task parameterising is time-consuming, some of the more difficult gestures to pose cause fatigue and while the gesture recognition rate is generally high, instances of false positives and false negatives occur, of which the user is notified, following a validation process [92].

6.2.3. Virtual and Augmented Reality Systems

Work in virtual and augmented environments is progressing; however, VR [93] and AR [95,96] systems for cobot operations do not currently appear to be developed and tested to the point of practical deployment for the industry. What has been proposed in the literature are VR systems that allow users to simulate robotic movement in an immersive environment. However, each use case often requires creating a new virtual environment and current capability is limited to simulation rather than functioning as a practical programming method [93]. AR systems produce tracking inaccuracies, which can misdirect the cobot, and there are also visual constraints. A reduced field of view from the AR wearable devices and occlusion problems restrict the user's perspective of the computer-generated content and could impose risks when deployed in workplaces. In addition, users can be distracted as they continuously swap between augmented and real environments, disrupting the AR system's fluidity [95].

6.3. Summary of Findings and Responses to Research Questions

Section 3 of this paper describes the common programming and control methods for cobots, existing and proposed in the literature. Solutions from seven leading cobot vendors were evaluated, and systems proposed in the literature were critiqued. The impact of a cobot's task complexity on the skills required to program it was addressed in Section 5, with a graphical representation and summary of the analysis provided at the end of that section. The effectiveness of existing cobot programming and control methods and the practicality of methods proposed in the literature were assessed in Sections 6.1 and 6.2, which focused on existing and proposed programming and control methods, respectively. The final considerations were to determine the most appropriate candidate for the role of cobot programmer and assess whether the current or proposed solutions can be matched to that person's skillset. In Section 2, it was argued that rather than focusing on the skills required to *program* a cobot, the knowledge of the *task* should be the focal point. The view was that it is often the case that the person who knows the task well, is not competent with programming and the person who is competent with programming does not know the task well. Operators with knowledge of and experience with a task that a cobot is intended to assist with possess valuable insights into precision and efficiency, details of which may be difficult to relay to a contracted programmer. Moreover, adaptive task-sharing principles aim to embed cobot programming into workers' duties. Based on these findings, a task-experienced worker would be the most appropriate programmer, but the programming methods analysed have not been specifically developed for a novice, as discussed in Section 5. Compounding this problem, the scale in Figure 7 indicates that existing cobot programming and control methods require more skill as cobot task complexity increases. In addition, the investigation in Section 6.2 revealed that the methods proposed in the literature were impractical or not sufficiently developed for reliable deployment. In general, these current cobot programming and control methods and those proposed in the literature require more advanced skills or are underdeveloped to be used by a worker with no robotic or programming experience. In the absence of a designated cobot programmer, therefore, either the upskilling of cobot novices, who are task-savvy operators or the development of a new generation of simplified cobot programming and control methods are the only viable options.

7. Discussion and Future Work

Several elements restrict the establishment of a complete, accurate, and clearly defined cobot task complexity to skill requirement matrix. Cobot task complexity is difficult to define due to variations in the definers' perceptions of *complexity*, which in turn, is conceivably shaped by their levels of skill and approaches to the programming of the task. For example, someone with more programming expertise may see a task as less complex, which could seem daunting to a less skilled programmer. Another key issue is that of the programming method used, especially concerning the tentative selection area between programming methods of different complexity, as discussed in Section 5.4 and summarised in Figure 6. Within that discretionary zone, a task may require more skills if programmed with a script-based method and less with a Teach Pendant.

The ramifications of an incorrectly programmed cobot can be severe, regardless of the programmer's skill level, although programming errors might be more likely with a less skilled programmer. Consequences of a poorly programmed cobot could range from a delay in a manufacturing process, damage to or destruction of the cobot, equipment or products, to human injury or even fatality, particularly if manipulator speed or force settings are not constrained. In addition, companies expose themselves to possible legal action due to injuries caused by their robots, with significant financial claims filed, especially if negligence is a factor [105].

The purpose of this paper was to review cobot programming and control methods and present a broad view of the complexity of the methods and cobot tasks from the perspective of a user skill level. Considerably more research and analysis should be conducted in cobot programming and control methods, focusing on flexibility and ease of use. Humans are the principal collaborators with cobots, so there should be a close connection to the human skills required to interact with a cobot partner. With further development, existing, proposed or hybrid systems could lead to a new generation of human-centric cobot smart control.

8. Conclusions

In a collaborative environment, there is an interaction between humans, who know the task, and cobots, who have been programmed to perform their part. Such an environment consists of many variables, from the type of cobot and programming method used and the programmer's skillset to the complexity of the task. Task and program complexity, along with programming skill levels, have been considered in broad terms and compared with existing and proposed programming and control methods in the literature to evaluate the relationship between task complexity and the skills required to program cobots. This analysis was then used to assess the claims made by the leading cobot vendors about the skills required to program their cobots. Cobot vendors typically emphasise the ease of programming their cobots without reference to task complexity. The findings of this study have revealed that even for tasks of relatively low complexity, some level of robotic and programming skill is required to ensure a safe and effective outcome. As task complexity increases, so do the required robotics and programming skills, contributing to a skills dilemma. Furthermore, to complete highly complex tasks, the programming skills required often exceeded those of the robotics skills. A task-focused approach highlights the benefit of the cobot programming role being performed by the worker with the best knowledge of the task rather than an expert with the best programming knowledge. The worker's lack of programming experience adds to the skills dilemma. Existing and proposed solutions are not currently suitable for that type of deployment. The solution is a programming system requiring no technical expertise, regardless of the task complexity level.

Author Contributions: Conceptualization, P.G., C.-T.C. and T.Y.P.; methodology, P.G.; validation, P.G.; formal analysis, P.G.; investigation, P.G.; resources, C.-T.C. and T.Y.P.; writing—original draft preparation, P.G., C.-T.C. and T.Y.P.; writing—review and editing, P.G., C.-T.C., T.Y.P. and K.N.; visualisation, P.G.; supervision, C.-T.C., T.Y.P. and K.N.; project administration, C.-T.C. T.Y.P. and K.N.; funding acquisition, C.-T.C. and T.Y.P. 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.

Acknowledgments: We would like to acknowledge the support of RMIT University for providing resources and facilities to conduct this research.

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

References

1. Bloss, R. Collaborative robots are rapidly providing major improvements in productivity, safety, programming ease, portability and cost while addressing many new applications. *Ind. Robot. Int. J.* **2016**, *43*, 463–468. [CrossRef]
2. De Backer, K.; DeStefano, T.; Menon, C.; Suh, J.R. Industrial robotics and the global organisation of production. *OECD Sci. Technol. Ind. Work. Pap.* **2018**, *3*, 1–44.
3. Benotsmane, R.; Kovács, G.; Dudás, L. Economic, social impacts and operation of smart factories in Industry 4.0 focusing on simulation and artificial intelligence of collaborating robots. *Soc. Sci.* **2019**, *8*, 143. [CrossRef]
4. Bryndin, E. Collaboration Robots with Artificial Intelligence (AI) as digital doubles of person for communication in public life and space. *Bp. Int. Res. Exact Sci.* **2019**, *1*, 1–11.
5. Gasparetto, A.; Scalera, L. A brief history of industrial robotics in the 20th century. *Adv. Hist. Stud.* **2019**, *8*, 24–35. [CrossRef]
6. Collaborative Robot Applications. Available online: https://www.universal-robots.com/applications (accessed on 10 December 2022).
7. Knudsen, M.; Kaivo-Oja, J. Collaborative robots: Frontiers of current literature. *J. Intell. Syst. Theory Appl.* **2020**, *3*, 13–20. [CrossRef]
8. Bi, Z.M.; Luo, C.; Miao, Z.; Zhang, B.; Zhang, W.J.; Wang, L. Safety assurance mechanisms of collaborative robotic systems in manufacturing. *Robot. Comput.-Integr. Manuf.* **2021**, *67*, 102022. [CrossRef]
9. Fischer, S. Robotics—Market Data Analysis & Forecasts. Statista Technology Market Outlook. 2022. Available online: https://www.statista.com/study/116601/robotics-market-data-analysis-and-forecasts (accessed on 2 October 2022).
10. Roehl, C. Know Your Machine: Traditional Industrial Robots vs. Cobots. 2022. Available online: https://www.universal-robots.com/blog/know-your-machine-traditional-industrial-robots-vs-cobots (accessed on 12 January 2023).
11. Vojić, S. Applications of collaborative industrial robots. *Mach. Technol. Mater.* **2020**, *14*, 96–99.
12. Dmytriyev, Y.; Carnevale, M.; Giberti, H.; Todeschini, G. On cobot programming in industrial tasks: A test case. In Proceedings of the 2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications, Ankara, Turkey, 9–11 June 2022; IEEE: New York, NY, USA, 2022; pp. 1–9.
13. El Zaatari, S.; Marei, M.; Li, W.; Usman, Z. Cobot programming for collaborative industrial tasks: An overview. *Robot. Auton. Syst.* **2019**, *116*, 162–180. [CrossRef]
14. Schou, C.; Andersen, R.S.; Chrysostomou, D.; Bøgh, S.; Madsen, O. Skill-based instruction of collaborative robots in industrial settings. *Robot. Comput.-Integr. Manuf.* **2018**, *53*, 72–80. [CrossRef]
15. Schoen, A.; White, N.; Henrichs, C.; Siebert-Evenstone, A.; Shaffer, D.; Mutlu, B. CoFrame: A System for Training Novice Cobot Programmers. In Proceedings of the 2022 17th ACM/IEEE International Conference on Human-Robot Interaction, Sapporo, Japan, 7–10 March 2022; IEEE: New York, NY, USA, 2022; pp. 185–194.
16. Kildal, J.; Tellaeche, A.; Fernández, I.; Maurtua, I. Potential users' key concerns and expectations for the adoption of cobots. *Procedia CIRP* **2018**, *72*, 21–26. [CrossRef]
17. Aaltonen, I.; Salmi, T. Experiences and expectations of collaborative robots in industry and academia: Barriers and development needs. *Procedia Manuf.* **2019**, *38*, 1151–1158. [CrossRef]
18. Kopp, T.; Baumgartner, M.; Kinkel, S. Success factors for introducing industrial human-robot interaction in practice: An empirically driven framework. *Int. J. Adv. Manuf. Technol.* **2021**, *112*, 685–704. [CrossRef]

19. Giannopoulou, G.; Borrelli, E.M.; McMaster, F. "Programming-It's not for Normal People": A Qualitative Study on User-Empowering Interfaces for Programming Collaborative Robots. In Proceedings of the 2021 30th IEEE International Conference on Robot & Human Interactive Communication, Vancouver, BC, Canada, 8–12 August 2021; IEEE: New York, NY, USA, 2021; pp. 37–44.
20. Bogue, R. Europe continues to lead the way in the collaborative robot business. *Ind. Robot. Int. J.* **2016**, *43*, 6–11. [CrossRef]
21. Schumacher, S.; Hall, R.; Waldman-Brown, A.; Sanneman, L. Technology Adoption of Collaborative Robots for Welding in Small and Medium-sized Enterprises: A Case Study Analysis. In Proceedings of the Conference on Production Systems and Logistics, Vancouver, BC, Canada, 17–20 May 2022; Publish-Ing: Hannover, Germany, 2022; pp. 462–471.
22. Fantini, P.; Pinzone, M.; Sella, F.; Taisch, M. Collaborative robots and new product introduction: Capturing and transferring human expert knowledge to the operators. In *Advances in Ergonomics of Manufacturing: Managing the Enterprise of the Future, Proceedings of the AHFE 2017 International Conference on Human Aspects of Advanced Manufacturing, Los Angeles, CA, USA, 17–21 July 2017*; Springer International Publishing: Berlin/Heidelberg, Germany, 2018; Volume 8, pp. 259–268.
23. Pieskä, S.; Kaarela, J.; Mäkelä, J. Simulation and programming experiences of collaborative robots for small-scale manufacturing. In Proceedings of the 2018 2nd International Symposium on Small-scale Intelligent Manufacturing Systems, Cavan, Ireland, 16–18 April 2018; IEEE: New York, NY, USA, 2018; pp. 1–4.
24. Ionescu, T.B.; Schlund, S. A participatory programming model for democratizing cobot technology in public and industrial Fablabs. *Procedia CIRP* **2019**, *81*, 93–98. [CrossRef]
25. Schmidbauer, C.; Schlund, S.; Ionescu, T.B.; Hader, B. Adaptive task sharing in human-robot interaction in assembly. In Proceedings of the 2020 IEEE International Conference on Industrial Engineering and Engineering Management, Singapore, 14–17 December 2020; IEEE: New York, USA, 2020; pp. 546–550.
26. Owen-Hill, A. What Are the Different Programming Methods for Robots? Available online: https://blog.robotiq.com/what-are-the-different-programming-methods-for-robots (accessed on 19 October 2022).
27. ABB Robotics Operating Manual Introduction to RAPID. Available online: http://rovart.cimr.pub.ro/docs/OpIntroRAPID.pdf (accessed on 4 October 2022).
28. Universal Robots. The URScript Programming Language. Available online: https://s3-eu-west-1.amazonaws.com/ur-support-site/18383/scriptmanual_en_1.3.pdf (accessed on 4 October 2022).
29. Gusan, V.; Țîțu, M.A.; Oprean, C. Industrial robots versus collaborative robots—The place and role in non-conventional technologies. *Acta Tech. Napoc.-Ser. Appl. Math. Mech. Eng.* **2022**, *65*, 101–110.
30. Michaelis, J.E.; Siebert-Evenstone, A.; Shaffer, D.W.; Mutlu, B. Collaborative or simply uncaged? understanding human-cobot interactions in automation. In Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, Honolulu, HI, USA, 25–30 April 2020; pp. 1–12.
31. Tilley, J. *Automation, Robotics, and the Factory of the Future*; McKinsey & Company: Los Angeles, CA, USA, 2017.
32. Simoes, A.C.; Lucas Soares, A.; Barros, A.C. Drivers impacting cobots adoption in manufacturing context: A qualitative study. In *Advances in Manufacturing II: Volume 1— Solutions for Industry 4.0*; Springer: Cham, Switzerland, 2019; pp. 203–212.
33. Trattner, A.; Hvam, L.; Forza, C.; Herbert-Hansen, Z.N.L. Product complexity and operational performance: A systematic literature review. *CIRP J. Manuf. Sci. Technol.* **2019**, *25*, 69–83. [CrossRef]
34. Transeth, A.A.; Stepanov, A.; Linnerud, Å.S.; Ening, K.; Gjerstad, T. Competitive high variance, low volume manufacturing with robot manipulators. In Proceedings of the 2020 3rd International Symposium on Small-Scale Intelligent Manufacturing Systems, Gjovik, Norway, 10–12 June 2020; IEEE: New York, USA, 2020; pp. 1–7.
35. Kootbally, Z. Industrial robot capability models for agile manufacturing. *Ind. Robot. Int. J.* **2016**, *43*, 481–494. [CrossRef]
36. Harbers, M.; Peeters, M.M.; Neerincx, M.A. Perceived autonomy of robots: Effects of appearance and context. In Proceedings of the A World with Robots: International Conference on Robot Ethics, Lisbon, Portugal, 23–24 October 2017; Springer International Publishing: Berlin/Heidelberg, Germany, 2017; pp. 19–33.
37. Khamis, A.; Hussein, A.; Elmogy, A. Multi-robot task allocation: A review of the state-of-the-art. *Coop. Robot. Sens. Netw.* **2015**, *2015*, 31–51.
38. Ho, M.; Farid, A.; Majumdar, A. Towards a Framework for Comparing the Complexity of Robotic Tasks. In Proceedings of the Algorithmic Foundations of Robotics XV: Proceedings of the Fifteenth Workshop on the Algorithmic Foundations of Robotics. University of Maryland, College Park, MD, USA, 22–24 June 2022; Springer International Publishing: Cham, Switzerland, 2022; pp. 273–293.
39. Bengoa, P.; González-Ojeda, I.D.; Ibarguren, A.; Goenaga, B.; Martínez-De-Lahidalga, S.; Gkournelos, C.; Lotsaris, K.; Angelakis, P.; Makris, S.; Antolín-Urbaneja, J.C. Coordination of Two Robots for Manipulating Heavy and Large Payloads Collaboratively: SOFOCLES Project Case Use. In *Advances and Applications in Computer Science, Electronics, and Industrial Engineering, Proceedings of the Conference on Computer Science, Electronics and Industrial Engineering (CSEI 2021), Ambato, Ecuador, 26 May 2022*; Springer International Publishing: Cham, Switzerland, 2022; pp. 255–271.
40. Chromjakova, F.; Trentesaux, D.; Kwarteng, M.A. Human and cobot cooperation ethics: The process management concept of the production workplace. *J. Compet.* **2021**, *13*, 21–38. [CrossRef]
41. Alitappeh, R.J.; Jeddisaravi, K. Multi-robot exploration in task allocation problem. *Appl. Intell.* **2022**, *52*, 2189–2211. [CrossRef]
42. Seenu, N.; Kuppan Chetty, R.M.; Ramya, M.M.; Janardhanan, M.N. Review on state-of-the-art dynamic task allocation strategies for multiple-robot systems. *Ind. Robot. Int. J. Robot. Res. Appl.* **2020**, *47*, 929–942.

43. Schneider, E.; Sklar, E.I.; Parsons, S. Mechanism selection for multi-robot task allocation. In Proceedings of the Towards Autonomous Robotic Systems: 18th Annual Conference, TAROS 2017, Guildford, UK, 19–21 July 2017; Springer International Publishing: Berlin/Heidelberg, Germany, 2017; pp. 421–435.
44. Murugappan, E.; Subramanian, N.; Rahman, S.; Goh, M.; Chan, H.K. Performance analysis of clustering methods for balanced multi-robot task allocations. *Int. J. Prod. Res.* **2022**, *60*, 4576–4591. [CrossRef]
45. Semwal, T.; Jha, S.S.; Nair, S.B. On ordering multi-robot task executions within a cyber physical system. *ACM Trans. Auton. Adapt. Syst. (TAAS)* **2017**, *12*, 1–27. [CrossRef]
46. Sathyan, A.; Ma, O. Collaborative control of multiple robots using genetic fuzzy systems. *Robotica* **2019**, *37*, 1922–1936. [CrossRef]
47. Johnson, L.B.; Choi, H.L.; How, J.P. The role of information assumptions in decentralized task allocation: A tutorial. *IEEE Control. Syst. Mag.* **2016**, *36*, 45–58.
48. Cutting Edge Industrial Collaborative Robots, Built to Do More. Available online: https://www.universal-robots.com/e-series/ (accessed on 15 December 2022).
49. Karami, H.; Darvish, K.; Mastrogiovanni, F. A task allocation approach for human-robot collaboration in product defects inspection scenarios. In Proceedings of the 2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), Naples, Italy, 31 August–4 September 2020; IEEE: New York, NY, USA, 2020; pp. 1127–1134.
50. Lee, M.L.; Behdad, S.; Liang, X.; Zheng, M. Task allocation and planning for product disassembly with human–robot collaboration. *Robot. Comput.-Integr. Manuf.* **2022**, *76*, 102306. [CrossRef]
51. Roncone, A.; Mangin, O.; Scassellati, B. Transparent role assignment and task allocation in human robot collaboration. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; IEEE: New York, NY, USA, 2017; pp. 1014–1021.
52. Müller, R.; Vette, M.; Geenen, A. Skill-based dynamic task allocation in human-robot-cooperation with the example of welding application. *Procedia Manuf.* **2017**, *11*, 13–21. [CrossRef]
53. Liau, Y.Y.; Ryu, K. Task allocation in human-robot collaboration (HRC) based on task characteristics and agent capability for mold assembly. *Procedia Manuf.* **2020**, *51*, 179–186. [CrossRef]
54. Liau, Y.Y.; Ryu, K. Genetic algorithm-based task allocation in multiple modes of human–robot collaboration systems with two cobots. *Int. J. Adv. Manuf. Technol.* **2022**, *119*, 7291–7309. [CrossRef]
55. Ranz, F.; Hummel, V.; Sihn, W. Capability-based task allocation in human-robot collaboration. *Procedia Manuf.* **2017**, *9*, 182–189. [CrossRef]
56. Malik, A.A.; Bilberg, A. Complexity-based task allocation in human-robot collaborative assembly. *Ind. Robot. Int. J. Robot. Res. Appl.* **2019**, *9*, 182–189. [CrossRef]
57. Johannsmeier, L.; Haddadin, S. A hierarchical human-robot interaction-planning framework for task allocation in collaborative industrial assembly processes. *IEEE Robot. Autom. Lett.* **2016**, *2*, 41–48. [CrossRef]
58. Evangelou, G.; Dimitropoulos, N.; Michalos, G.; Makris, S. An approach for task and action planning in human–robot collaborative cells using AI. *Procedia Cirp* **2021**, *97*, 476–481. [CrossRef]
59. Yu, T.; Huang, J.; Chang, Q. Optimizing task scheduling in human-robot collaboration with deep multi-agent reinforcement learning. *J. Manuf. Syst.* **2021**, *1*, 487–499. [CrossRef]
60. Make the Simple Tasks Easy and the Complex Tasks Possible. Available online: https://www.universal-robots.com/products/polyscope (accessed on 15 December 2022).
61. Offline Simulator—E-Series—URSim for non Linux 5.12.5. Available online: https://www.universal-robots.com/download/software-e-series/simulator-non-linux/offline-simulator-e-series-ur-sim-for-non-linux-5125/ (accessed on 30 November 2022).
62. Wizard Easy Programming. Available online: https://new.abb.com/products/robotics/application-software/wizard (accessed on 15 December 2022).
63. Technical Reference Manual. RAPID Instructions, Functions and Data Types. Available online: https://library.e.abb.com/public/688894b98123f87bc1257cc50044e809/Technical%20reference%20manual_RAPID_3HAC16581-1_revJ_en.pdf (accessed on 15 December 2022).
64. RobotStudio. Available online: https://new.abb.com/products/robotics/robotstudio/robotstudio-desktop (accessed on 15 December 2022).
65. 5 Key Reasons Why the Fanuc CRX Collaborative Robot Is Right for Everyone. Available online: https://crx.fanucamerica.com/why-cobots-collaborative-robots/ (accessed on 15 December 2022).
66. Programming a Robot Is Child's Play. Available online: https://www.festo.com/us/en/e/journal/programming-a-robot-as-child-s-play-id_28594/ (accessed on 16 December 2022).
67. iiQKA: Robots for the People. Available online: https://www.kuka.com/en-au/future-production/iiqka-robots-for-the-people (accessed on 16 December 2022).
68. User-Friendly Touchscreen Teach Pendant for Novice Robot Users. Available online: https://www.motoman.com/en-us/products/programming/smart-pendant (accessed on 16 December 2022).
69. HC Series Cobots—Your New Work Assistants. Available online: https://www.yaskawa.eu.com/products/robots/cobot (accessed on 16 December 2022).
70. Motoman NX100 Controller. Inform II User's Manual. Available online: http://www.wtech.com.tw/public/download/manual/yaskawa/NX100/YASKAWA%20NX100%20INFORM%20II%20USER%20Manual.pdf (accessed on 3 January 2023).

71. Automation Control Environment (ACE) Version 4 User Manual. Available online: https://www.edata.omron.com.au/eData/Robotics/I633-E-04.pdf (accessed on 10 December 2022).
72. RoboDK Basic Guide. Available online: https://robodk.com/doc/en/Basic-Guide.html (accessed on 4 January 2023).
73. Why Is It Worth Using . . . ArtiMinds Robot Programming Software Solutions? Available online: https://www.artiminds.com/robot-programming-software/ (accessed on 18 December 2022).
74. Wandelbots. Everyone Can Work with Robots. Available online: https://wandelbots.com/en/ (accessed on 5 January 2023).
75. Pickit. Guide your Robot. Available online: https://www.pickit3d.com/en (accessed on 5 January 2023).
76. Robotmaster Offline Programming Software for Robots. Available online: https://www.robotmaster.com/en (accessed on 6 January 2023).
77. Universal Robots. Simplify Robot Programming with G-Code. Available online: https://www.universal-robots.com/blog/simplify-robot-programming-with-g-code/ (accessed on 5 January 2023).
78. ROS—Robot Operating System. Available online: https://www.ros.org/ (accessed on 7 January 2023).
79. Plant Automation Technology. Different Types of Robot Programming Languages. Available online: https://www.plantautomation-technology.com/articles/different-types-of-robot-programming-languages (accessed on 12 December 2022).
80. U.S. Social Security Administration. Code Of Federal Regulations. 404.1568. Skills Requirements. Available online: https://www.ssa.gov/OP_Home/cfr20/404/404-1568.htm (accessed on 2 February 2023).
81. Schou, C.; Madsen, O. A plug and produce framework for industrial collaborative robots. *Int. J. Adv. Robot. Syst.* **2017**, *14*, 1729881417717472. [CrossRef]
82. Heimann, O.; Guhl, J. Industrial robot programming methods: A scoping review. In Proceedings of the 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation, Vienna, Austria, 8–11 September 2020; IEEE: New York, NY, USA, 2020; Volume 1, pp. 696–703.
83. Weintrop, D.; Afzal, A.; Salac, J.; Francis, P.; Li, B.; Shepherd, D.C.; Franklin, D. Evaluating CoBlox: A comparative study of robotics programming environments for adult novices. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, Montreal, QC, Canada, 21–26 April 2018; pp. 1–12.
84. Codejig. Block Coding. Available online: https://www.codejig.com/en/block-based-coding (accessed on 10 January 2023).
85. Fogli, D.; Gargioni, L.; Guida, G.; Tampalini, F. A Hybrid Approach to User-Oriented Programming of Collaborative Robots. *Robot. Comput.-Integr. Manuf.* **2022**, *73*, 102234. [CrossRef]
86. GitHub. Your AI Pair Programmer GitHub Copilot. Available online: https://github.com/features/copilot/ (accessed on 17 February 2023).
87. Borboni, A.; Reddy, K.V.; Elamvazuthi, I.; AL-Quraishi, M.S.; Natarajan, E.; Azhar Ali, S.S. The Expanding Role of Artificial Intelligence in Collaborative Robots for Industrial Applications: A Systematic Review of Recent Works. *Machines* **2023**, *11*, 111. [CrossRef]
88. Hiltner, J. Self-learning Intelligence for Object Recognition. *Quality*, 1 July 2019.
89. De Marchi, M.; Gualtieri, L.; Rojas, R.A.; Rauch, E.; Cividini, F. Integration of an Artificial Intelligence Based 3D Perception Device into a Human-Robot Collaborative Workstation for Learning Factories. In Proceedings of the 11th Conference on Learning Factories (CLF), Graz, Austria, 31 June–2 July 2021.
90. Bai, Y.; Lindqvist, B.; Karlsson, S.; Kanellakis, C.; Nikolakopoulos, G. Multi-Robot Task Allocation Framework with Integrated Risk-Aware 3D Path Planning. In Proceedings of the 2022 30th Mediterranean Conference on Control and Automation (MED), Athens, Greece, 28 June–1 July 2022; IEEE: New York, NY, USA, 2022; pp. 481–486.
91. Kaczmarek, W.; Panasiuk, J.; Borys, S.; Banach, P. Industrial robot control by means of gestures and voice commands in off-line and on-line mode. *Sensors* **2020**, *20*, 6358. [CrossRef]
92. Mendes, N.; Safeea, M.; Neto, P. Flexible programming and orchestration of collaborative robotic manufacturing systems. In Proceedings of the 2018 IEEE 16th International Conference on Industrial Informatics, Porto, Portugal, 18–20 July 2018; IEEE: New York, NY, USA, 2018; pp. 913–918.
93. Pérez, L.; Diez, E.; Usamentiaga, R.; García, D.F. Industrial robot control and operator training using virtual reality interfaces. *Comput. Ind.* **2019**, *109*, 114–120. [CrossRef]
94. Psarakis, L.; Nathanael, D.; Marmaras, N. Fostering short-term human anticipatory behavior in human-robot collaboration. *Int. J. Ind. Ergon.* **2022**, *87*, 103241. [CrossRef]
95. De Pace, F.; Manuri, F.; Sanna, A.; Fornaro, C. A systematic review of Augmented Reality interfaces for collaborative industrial robots. *Comput. Ind. Eng.* **2020**, *149*, 106806. [CrossRef]
96. Ong, S.K.; Yew, A.W.W.; Thanigaivel, N.K.; Nee, A.Y. Augmented reality-assisted robot programming system for industrial applications. *Robot. Comput.-Integr. Manuf.* **2020**, *61*, 101820. [CrossRef]
97. Introducing Wizard Easy Programming. Available online: https://youtu.be/2l-IKmdcJsM (accessed on 18 December 2022).
98. Omron Adept Robot Programming: It's That Easy! Available online: https://www.youtube.com/watch?v=9Y-3i0I07Bc (accessed on 18 December 2022).
99. Robotiq. Skills to Integrate a Robot. Available online: https://blog.robotiq.com/skills-to-integrate-a-robot?_ga=2.195912812.1784018681.1676172069-1475687650.1676171346 (accessed on 30 January 2023).
100. Doyle-Kent, M.; Kopacek, P. Adoption of collaborative robotics in industry 5.0. An Irish industry case study. *IFAC-Pap.* **2021**, *54*, 413–418. [CrossRef]

101. Gjeldum, N.; Aljinovic, A.; Crnjac Zizic, M.; Mladineo, M. Collaborative robot task allocation on an assembly line using the decision support system. *Int. J. Comput. Integr. Manuf.* **2022**, *35*, 510–526. [CrossRef]
102. Universal Robots. The UR5e. Available online: https://www.universal-robots.com/products/ur5-robot/ (accessed on 10 October 2022).
103. Park, J.; Jung, W. A study on the development of a task complexity measure for emergency operating procedures of nuclear power plants. *Reliab. Eng. Syst. Saf.* **2007**, *92*, 1102–1116. [CrossRef]
104. Park, J.; Jang, I. Can we determine the complexity level of a proceduralized task? In Proceedings of the Transactions of the Korean Nuclear Society Virtual Spring Meeting, Jeju, Korea, 13–14 May 2021.
105. Kirschgens, L.A.; Ugarte, I.Z.; Uriarte, E.G.; Rosas, A.M.; Vilches, V.M. Robot hazards: From safety to security. *arXiv* **2018**, arXiv:1806.06681.