The International Journal of Advanced Manufacturing Technology (2023) 126:2097–2116
https://doi.org/10.1007/s00170-023-11246-y

ORIGINAL ARTICLE

Achieving batch-size-of-one production model in robot flexible assembly cells

Ziyue Jin[1] · **Romeo M. Marian**[1] · **Javaan S. Chahl**[1,2]

Received: 7 December 2022 / Accepted: 8 March 2023 / Published online: 18 March 2023

Abstract
Manufacturing industry is facing new challenges in that fast-changing demands for products and services from customers push manufacturers to be more flexible and adaptive. The concept of batch-size-of-one production is presented in this paper, which defines a fully automated, highly customised, and short lead time production model. The desired batch-size-of-one production model is a promising solution for the above challenges in manufacturing industry, especially for highly customised or families of similar products like in the mobile phone industry. Along with the concept, we introduce a novel control method that enables the desired batch-size-of-one production model in operation of robots in manufacturing and assembly systems. The strategy was developed for robot control based on a distributed system to enable industrial robots to receive job commands on the fly and to conduct different jobs without the need for reconfiguration and reprogramming and without overheads. The aim of the research is to create the basis for a fully automated robot flexible assembly cell to perform batch-size-of-one assembly tasks with minimal human involvement by eliminating interruptions from the reconfiguration and reprogramming processes. The proposed strategy has been validated in practice in a multi-robot, multi-product flexible assembly cell.

Keywords Industrial robot control · Flexible robotic assembly · Batch size of one · OPC UA

1 Introduction

1.1 Challenges in manufacturing industry

After decades of enjoying the efficiency of mass manufacturing, the concept of flexible manufacturing system (FMS) was introduced and designed to allow larger variety through lower batch volumes and to shorten the lead time while keeping the company profitable as customers expect more personalised products within very tight time frames [1]. Even so, today, the industry is facing new and increasing challenges that fast-changing requirements from customers for products and services have further forced manufacturers to become more adaptable and to follow the pace of the market dynamically and promptly [2–4]. Manufacturers try to satisfy such increasing and diversified demands by increasing their system flexibility to higher levels [4–7]. These challenges can evolve exponentially, and they require a paradigm shift in the search and implementation of new approaches and solutions [8–10]. The transformation of manufacturing based on Industry 4 philosophy (blurring the lines between the physical, digital, and biological worlds) has the potential to lead to such solutions [11].

The industrial robots used in manufacturing have demonstrated their performance and were critical enablers of efficient mass production of complex products made at large scale, like cars. Despite their advantages, well-behaved robots and manipulators that follow exactly the prescribed programmes implemented to increase efficiency have reached a hard limit and cannot meet today's flexible production requirements [12]. Even if the current robot control systems are designed for quick and easy configuration, stoppages and production disruptions are still unavoidable while new tasks are assigned to the robots [13]. Batch volume is a constraint that limits the production efficiency of robotic

✉ Ziyue Jin
Ziyue.Jin@mymail.unisa.edu.au

[1] UNISA STEM, Australian Research Centre for Interactive and Virtual Environments, University of South Australia, Mawson Lakes, Australia

[2] Joint and Operations Analysis Division, Defence Science and Technology Organisation, Melbourne, Australia

systems, which in turn reduces the systems' flexibility and adaptability.

In order to increase the adaptability of the rigid robotic systems (rigidity in the sense that they strictly follow a set programme), human operators are therefore still a highly critical constituent required to achieve the desired performance in operational flexibility [14, 15]. However, lately, even extensive use of human operators in specific operations is a bottleneck when flexibility is pushed to its limits, when the output is a suite of small batches of products or even down to batch-size-of-one (BSO). The limitation of the human operators to efficient flexibility comes from limitations in fast-learning or potentially too many very similar sequences of operations for similar products, which leads to the danger of confusions and possible errors.

In particular, the assembly process is the most human labour-intensive function in the traditional manufacturing model. Due to specific constraints (mainly originating from diversity and complexity of tasks), assembly is the most difficult to automate and adapt for extensive use of robots. Thus, a paradigm shift is needed for robotic control methods to enhance robotic systems' adaptiveness, robustness, and resilience [16]. Higher flexibility and lower production volumes, ideally down to batch size of one (also known as market-of-one [17]), as well as shorter lead times, are still beyond the capability of industrial robotic systems in the state of the art.

1.2 Batch-size-of-one production model in Industry 4

The concept of BSO originally refers to a make-to-order production model, which defines a manufacturing strategy with high customisation but suffers from low efficiency. However, in the era of Industry 4, BSO production model means that highly customised products are supposed to be made with full automation and low lead time [9, 18]. This is referred to as BSO production model in this paper. Achieving the desired BSO in robotic assembly cells is a promising solution to the aforementioned challenges for manufacturing industry for highly customised or families of similar products.

In terms of the industrial application of the desired BSO, mobile phone industry is one of the typical examples where the proposed strategy can be deployed. Mobile phone assembly is a typical labour-intensive process as the traditional production lines can hardly offer the desired automation and flexibility [19, 20]. Mobile phone giants prefer to set up their factories in countries that offer cheap labours [21, 22], which, in turn, challenges the supply chain management.

Figure 1 shows the typical components of a mobile phone, including PCB, sensors, cameras, antenna, cables, etc. In the process of assembling a mobile phone, the components will be put in place in a particular sequence. However, depending on the specific models of the mobile phone, the size and number of the components and assembly sequence are specific to each model. Therefore, the proposed BSO could help the manufacturer to achieve highly flexible production regarding the batch volume to satisfy dynamic demands from the market, instead of employing significant human operators in production lines to obtain such flexibility.

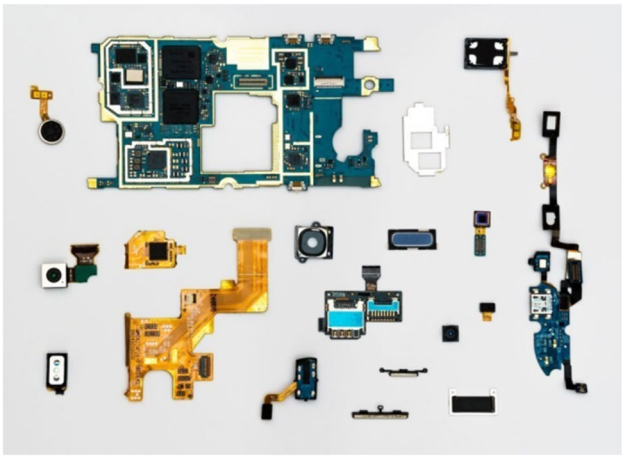

Fig. 1 Typical components for a mobile phone [23]

Currently, most studies are focusing on methods of optimising production schedules for enhancing the productiveness and adaptiveness in the mobile phone assembly process [24–26] However, less studies are dealing with building automated processes of mobile phone assembly. And fewer of them discuss achieving BSO production model in the process.

Industrial robots and machine vision technology are major catalysts for enabling the automation in mobile phone assembly systems. The versatile robotic arms act as the manipulator for moving parts in associate with the positioning function assisted by the machine vision system. Zhang, He, and Li [20] proposed a system for fulfilling automated camera lens insertion in mobile phone assembly process. However, the accuracy and precision of positioning parts via the machine vision system are a challenge. Another study by Yuan et al. [19] developed a machine vision system that employs a 9-point calibration method for improving object detection performance.

On the other hand, the enhancement of industrial robots can also contribute to achieve automated assembly operation for mobile phones. Force-guided robot, for instance, can mimic human's hands to perform notch-locked assembly [27] and solve misalignment problems in the mobile phone assembly [28].

However, despite extensive literature search, the authors could not find satisfactory solutions for achieving the desired BSO production model in either mobile phone assembly

systems or general robotic assembly applications, which is the initial motivation of this research.

1.3 Research problem

In this paper, we are proposing a strategy to solve the challenge of fulfilling the desired BSO production model in robotic assembly cells. This will be done by adding flexibility to a robot system, in particular the possibility to control the robots on the fly, without delays and interruptions, at the level of elementary task to be executed.

We are, then, testing and validating the proposed strategy on a very constrained problem in a robotic flexible assembly cell (RFAC) for the BSO production model. The RFAC is a multi-robot system, with several robots working in a serial/parallel arrangement to execute the required tasks.

The Research Problem, the focus of this paper, is how to develop a strategy to enable flexible control of a multi-robot RFAC to efficiently achieve automated BSO assembly.

An extended goal of this research, which is part of future work, is to accomplish a robust self-reconfigurable RFAC. In case a robot needs to be taken off-line (due to failure or maintenance requirements), tasks can be reallocated to the other functional robot(s) in the cell.

At the moment, as analysed in the next section that examines the state of the art, flexible control of robots is still an elusive goal that has not been attained.

The results of the research presented in this paper will contribute to enhance the flexibility and adaptiveness of robot systems. In particular, for the case of RFAC, the batch size will no longer be a constraint limiting the system's efficiency. Therefore, the assembly tasks can be further optimised in satisfying the dynamic demands of the market, which, in turn, builds strong flexibility, adaptiveness, and resilience of the RFAC.

The paper is organised as follows. Section 2 discusses the related works in enhancing the flexibility and adaptiveness of robotic assembly cells. Section 3 introduces the proposed robotic control and programming strategy. Section 4 presents the RFAC context based on the proposed method. Then, in Sect. 5, a case study is presented for the verification of the proposed method. Following that, the research results are discussed in Sect. 6. A conclusion section, 7, including recommendations for future work, closes the paper.

2 State of the art in flexible control of industrial robots

This literature review is focussed on the current robotic control strategies for building the flexibility of robotic assembly cells. There are limited references in the literature on applications of RFAC. Thus, in this section, the discussion is developed based on reviewing the related works that are or can be used in a robotic assembly context.

Reducing batch volume size is the core philosophy of FMS. In current FMS, the batch size has successfully reduced from mass production volume down to small volumes. However, it is difficult to further reduce the batch size down to one while maintaining the automation and efficiencies required for assembly systems.

2.1 Distributed robotic control systems

Employing a distributed system in robotic assembly cells is a common strategy used in the recent years to enhance the flexibility of the system [29–32]. Benefiting from the boom of information technology in manufacturing, the industrial robots are capable to communicate with other networked mechanisms via industrial communication protocols. Thus, a motion control mechanism can then be developed and placed within the local network or on a cloud server to control robots in the workcell remotely [15].

Distributed industrial robotic system, also known as service-oriented architecture (SOA) robotic system [33], includes three main sections: manipulators, control mechanism, and communication middleware.

In an early study by Blankenburg et al. [34], a distributed framework was proposed for flexibly controlling robots. Robot Operating System (ROS) was used to build the middle layer for the communication between the nodes in the network. The robotic motion commands were sent to the robots via the user interface—MoveIt, which is ROS-based robot motion control software. The feasibility was briefly verified by programming robots to conduct Pick-and-Place tasks. However, the proposed robot control architecture is unable to achieve automated new task allocation as the robotic control commands are sent manually via the control panel.

Vick et al. [33] introduced a service-oriented architecture (SOA) to achieve flexible robotic motion control strategy via a cloud-based control system. The intention was fulfilled by modularising the robot controller. Part of the function of the physical robot controller, such as trajectory planning, was transferred into a virtual controller, which was installed on a cloud-based computer. The industrial robot users could send robot motion commands to the planning service module via a human–machine interface (HMI) to control the robots. Although the study provided a feasible framework of such a system, the authors did not elaborate the detailed methods on how to establish an automated industrial robot system.

Another study by Tsarouchi et al. [35] developed a SOA structure to control robots, enabling a machine-to-cloud (M2C) communication mechanism. The method employed XML files to define the task sequence for the assembly tasks. Also, the ROS was used as the middleware to establish the communication channels between the devices. The proposed

solution established a distributed system in a robotic assembly cell and optimised the total flexibility of the system. However, the flexibility achieved in this case still relays on human operators' involvement.

Similarly, Vick and Kruger [15] proposed an architecture for establishing distributed industrial robot cell by using OPC Unified Architecture (OPC UA) [36] standard as the communication layer for the whole system. (Robotic services are stored in the cloud data centre.) Comparing with ROS, OPC UA brings higher flexibility and scalability for the system. However, the proposed strategy is still focussing on human–robot collaborative assembly processes.

2.2 Cyber-physical systems-based robot control systems

In the current Industry 4 paradigm shift, cyber-physical systems (CPS) are being developed and deployed to optimise traditional manufacturing system configurations. Comparing with traditional physical manufacturing systems, digitised systems offer much better accessibility against geographical constraints [33], which has a great potential for increasing the desired flexibility and adaptiveness of robot assembly cells.

In a recent study by Kaarlela et al. [37], the authors proposed a solution for remotely controlling the robots via a digital twin system. The digital twin consisted of a simulation environment for testing virtual replication and a robot programming software where the motion commands for the robots were coded up. Codes were then sent to the robots via the communication layer based on OPC UA protocol. The main contribution of this paper was that it demonstrates an application of using OPC UA as middleware to enable remote and flexible control of industrial robots.

Similarly, Muller, Deuerlein, and Koch [38] developed a CPS to make use of a mixed-reality (MR) environment for enhancing the teach-in method of industrial robot programming. The proposed approach could be run in an offline digital twin virtual context without the need of physical robots, which in turn shortens the robots' downtime for programming. However, human operators' involvement is still essential for transferring the pre-made programming for new task setup.

2.3 Synopsis of state-of-the-art solutions

Key characteristics of the reviewed cases are summarised into Table 1. It is clear that a distributed framework provides a promising architecture in building the desired flexible control for robotic workcells as the control mechanism can be geographically decoupled from the physical manipulators. CPS-based control system is a new concept in the era of Industry 4, which can enhance the systematic flexibility by building the digitised context.

There are a few solutions for the communication middleware, such as OPC UA, ROS, for robotic systems. Solutions can be chosen depending on the specific application, devices, and techniques. Comparing with others, it is believed that OPC UA is a much more promising solution in building communication middle layer in current manufacturing context as it offers high connectivity, interoperability, and scalability.

In terms of the control mechanism, using pre-scripted robotic motion commands is a major strategy for robots to conduct assembly tasks in the reviewed cases, which still limits the systematic automation and flexibility in the process. However, the encapsulated robotic motion commands that were developed by Tsarouchi et al. [35] have drawn the authors' attention, as the encapsulated task execution file can be developed, prepared, and sent apart from the running robots. With a well-developed control algorithm, it could achieve the on-the-fly robotic control.

Despite of the enhanced efficiency in those robot control processes, none of them has achieved the expected automation in new task assignment process. Human operators are still necessary for changing tasks for the systems. As defined in the BSO production model, new task assignment or any task change shall be conducted automatically for maintaining the desired automation level, which in turn achieves the expected high efficiency and short lead time.

3 Proposed robot control strategy

As analysed in the state of the art, the proposed strategy is also developed based on a distributed robotic system. A control algorithm is developed for sending encapsulated motion commands to the robots in a real-time and on-the-fly manner. The algorithm stays in a loop of receiving task execution file, sending commands to the robots, and waiting for the response of the robots. The task execution file is prepared for each individual assembly task. OPC UA standard is employed to build the communication middle layer as it is usable for industrial machinery and devices in the manufacturing context. Figure 2 illustrates a high level of data process flowchart for the strategy.

The flexibility in manufacturing systems is twofold and is defined as the capacity of the system to not only conduct a variety of similar tasks but also be adaptive to the unpredicted disruptions and dynamic production demands [39]. Two major strategies are employed to obtain the desired flexibility in robotic assembly cells. One strategy is to enhance the reconfigurability of the system, including system layout reconfiguration [40] and rapid robotic reprogramming

Table 1 Key characteristics of reviewed flexible robotic control systems vs. proposed strategy

Reference	System framework	Communication middle layer	Robotic control mechanism	Automated task change	Human operator involvement
Blankenburg et al. [34]	Distributed robotic system	ROS/ZeroMQ	Pre-scripted motion commands sent via MoviIt	No	Yes
Vick et al. [33]		CAN-protocol	Robotic motion commands sent via HMI by human operator	No	Yes
Tsarouchi et al. [35]		ROS	Robotic motion commands are modelled in XML file and sent via HMI	No	Yes
Vick and Kruger [15]		OPC UA/field-bus	Pre-scripted motion commands in Workerbot3 control system	No	Yes
Kaarlela et al. [37]	CPS-based robotic system	OPC UA	Pre-scripted motion commands sent via Digital Twin	No	Yes
Muller, Deuerlein, and Koch [38]		TCP/IP	Mixed reality interface	No	Yes
Proposed strategy	Distributed robotic system	OPC UA	Motion script developed for the robots to listen up on-coming motion commands from the developed control unit	Yes	No human operator needed for task change

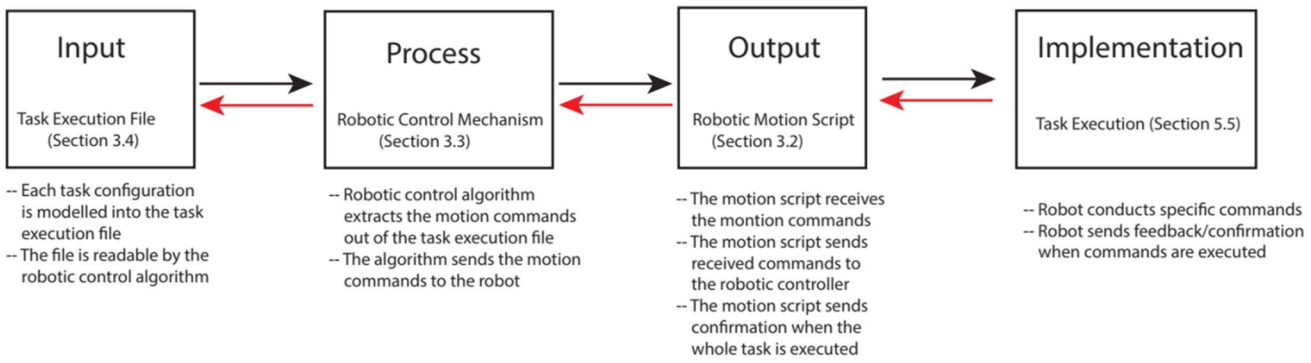

Fig. 2 Conceptual data process flowchart for the proposed robotic control strategy

capacity [41]. The second strategy is to deploy distributed robotic assembly systems, enabling robots to receiving motion commands on the fly. The proposed method in this paper falls into the second of the two categories.

3.1 System architecture

The proposed robot control strategy is developed based on a distributed architecture, which gives higher flexibility and scalability in comparison with traditional monolithic robot control architecture. The key elements and data flow diagrams are shown in Fig. 3. The robots in the workcell are networked and connected with a private cloud server. A robot motion control unit is integrated within the network as well, which enables on-the-fly robotic control strategy. The motion control unit is configured to communicate with robots in the workcell via industrial communication protocol, which in turn enables the real-time data exchange between the control unit and the robots. OPC UA is used for building the communication layer in the proposed strategy.

3.2 Robot configuration

In the traditional industrial applications, robots need flow chart before changing to a different job, and this results in interruptions. In the proposed strategy, there is no need for reprogramming while a new job is assigned to the robot/s.

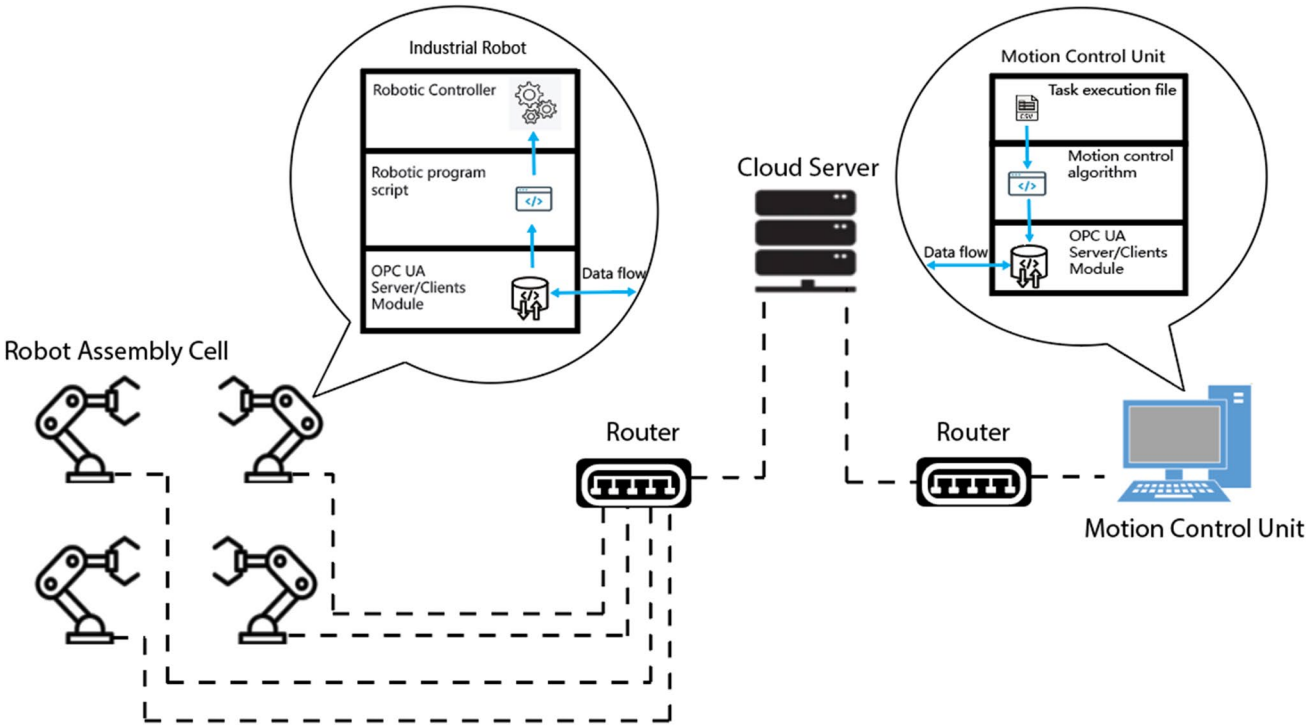

Fig. 3 Proposed framework and key elements for RFAC

This is because the unit of programme sent to be executed by the robot is not a programme consisting of a succession of tasks but just one task. The robots are set and configured to receive motion commands from the motion control software in real time. Data that is received by the robot consists of the details of the following task to be executed: desired type of motion (how does the robot need to move), and tool centre point (TCP) coordinates and orientation for the target points (where does the robot need to move).

A script is created to define the series of tasks to be executed by the robot. Then, the robot is programmed to run the script in a loop until termination signal is received. In the script, the robot will be initialised at the beginning, which is followed by defining variables for OPC UA data exchange. Then, a loop is defined for running a *check-execution-feedback* cycle. In the first place, the script will check if the variables have been updated with new motion command via OPC UA channel. Once new commands have been received, the robot will act to fulfil the desired motion. Then, a feedback variable will be updated to indicate the received command has been executed. Figure 4 shows the logic of the task execution script.

3.3 Motion control algorithm

Robot motion control software is developed and placed in the motion control unit, sending the tasks to be executed to the robots and receiving the execution feedback information from the robots. For each assembly task, the assembly plan and motion control commands are defined in a task execution file. The data in the task execution file can be extracted by the motion control algorithm and subsequently sent to the robot/s.

Figure 5 shows a detailed logical flowchart of the robotic motion control algorithm. Specifically, the OPC UA modules need to be initialised to create the desired communication channel at the beginning. Before entering the main loop, a series of variables will be defined. Then, the algorithm will check whether a new task execution file is received. Once the file is received, the algorithm will extract the data and pick up the first line of commands.

After checking the availability of the target robot, the data of the motion type code, target point's coordinates and approaching orientation will be sent to the robot. In the case of the target robot being unavailable (breakdown or planned maintenance), the algorithm will seek a backup robot within the RFAC and send the command to it. If there is no backup robot available, the algorithm will send a warning message, pause the programme, and wait for further advice. Once the robot finishes the required action, a confirmation of execution will be sent back to the motion control algorithm. The algorithm will then jump back to select the next line of the motion command from the extracted data. After all commands in the task execution

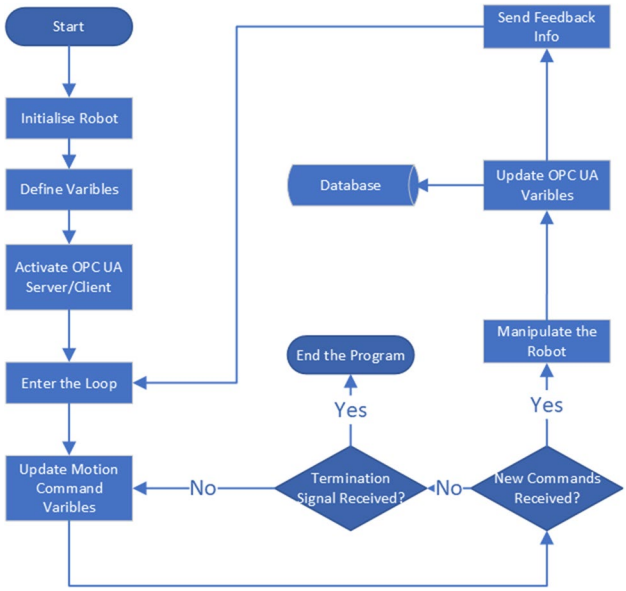

Fig. 4 Logical flowchart of robotic task execution script

file are conducted, the algorithm will jump further back to the beginning of the main loop to check if another new task execution file is received.

3.4 Task execution file

The assembly task plan is defined in a task execution file, including task sequence, assembly constraints, robot identifier, robot motion type, coordinates and orientation for the tool centre point (TCP), and so on. Specifically, the robot identifier encodes the robot to which the command is being sent. The robot motion type is a code that links to a unique robot functional movement, such as moveL, moveJ, open/close griper, etc., which has been well defined in the robot controller. When the robot receives the data, the requested robot movement can be easily selected. Moreover, if the command is for manipulating the robot to a certain position, the coordinates and approach orientation of the target point should be provided. This also permits avoidance of collision for multi-robot cells.

The task execution file can be created by various methods. For instance, it can be prepared in a virtual robot context, which is separated from the physical workcell. Alternatively, it can be developed by using the method of programming by demonstration. The developed file can then be stored in the local hard drive of the workstation, or in the cloud drive as long as it is accessible for the motion control module.

The information of the file name can be encoded into a QR code or RFID. When using a QR code scanner or RFID

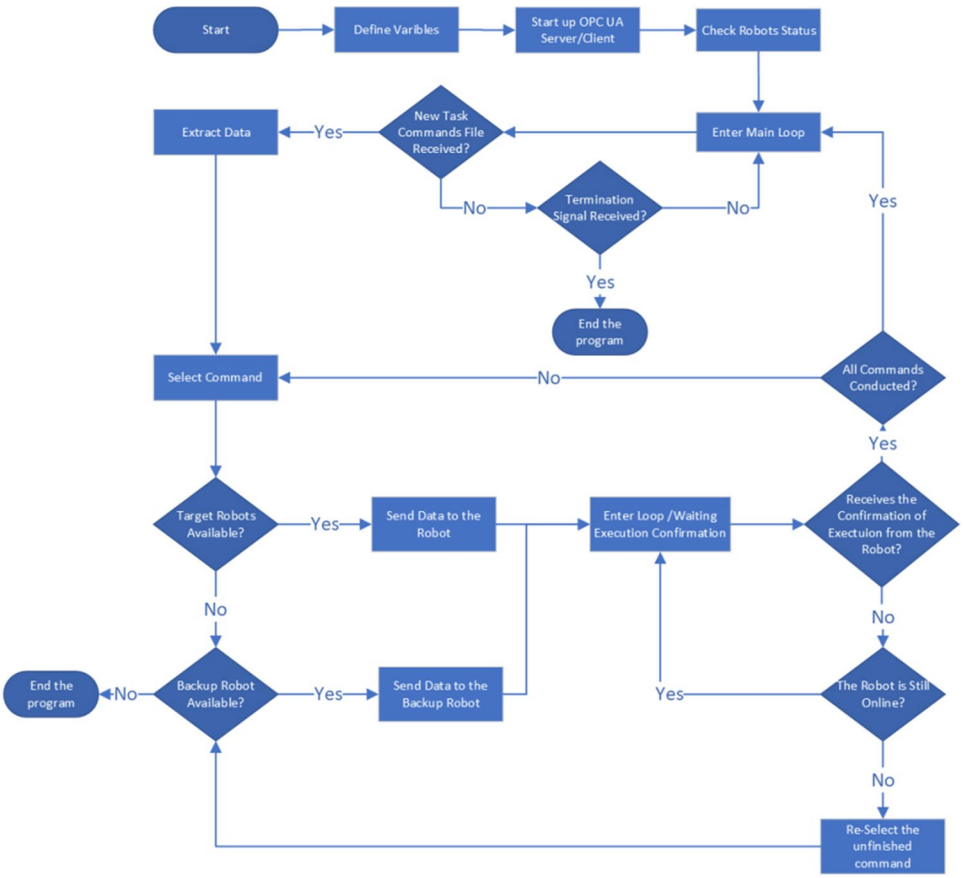

Fig. 5 Process flow of the motion control algorithm

reader, the process of loading new task execution file can be conducted automatically. As long as robots are available and functional in the RFAC to execute tasks, the algorithms will run in the loop to repeat data extraction and data transmission until an implicit or explicit requirement will stop the programme.

Therefore, a new task can be set without direct human operator involvement, and the desired flexibility, automation, and collaborative operation can be achieved in the robot cell with maximum efficiency.

The concepts of flexible robot automation and programming were implemented and tested on a RFAC testbed in University of South Australia (UniSA), being able to run batch-size-of-one assembly tasks in a fully automated manner, without or with minimal need of human involvement.

4 The robot flexible assembly cell

In this section, we present the RFAC and its modelling with the proposed framework, which is used as a test bed for validating the proposed strategy.

As a major subset of FAS, robot assembly cells (RAC) have attracted great attention from both academia and industries as they have significant potential in achieving a much higher level of flexibility [42]. In most cases, in the state of the art, the robots are pre-programmed to carry out specific task(s) in various manufacturing contexts [43], which, in turn, results in the unavoidable stoppage for scheduling and loading the new tasks. Therefore, the flexibility and adaptiveness of the RAC have reached a limitation as high frequency in changing assembly tasks impacts efficiency.

In order to further enhance the assembly system's flexibility, the concept of robot flexible assembly cells (RFAC) has been proposed and discussed [44]. The RFAC pushes the limits even further, being responsive to changes in product mix, set maintenance of parts of the cell or breakdowns.

The RFAC programming and scheduling is, likely, one of the most constrained robot problems. It involves multiple robots sharing a common space, the assembled products are continuously changing, making the task flexible and requiring continuous reconfiguration and adaptation [45].

The key difference between traditional RAC and RFAC is that the latter aims at eliminating human operators' involvement in the assembly processes where possible, while maintaining high level of flexibility, adaptiveness, and automation. Thus, RFAC has great potential in manufacturing and assembly systems to increase reliability, resilience, and robustness to reduce costs and avoid accidents, while being capable of responding to the fast-changing requirements and challenges from the market. Figure 6 illustrates a typical 2 robots RFAC. In the assembly process, the set of component parts will be transferred into the workcell by conveyor or AGV. Depending on the assembly plan, robots can work separately or collaboratively on the assembling task. The finished assembly can then be transferred away via the conveyor or AGV.

In the current Industry 4 era, a number of cutting-edge technologies have been successfully deployed in the manufacturing industry in the last decade, including industrial collaborative robots, cyber-physical system (CPS), internet-of-things (IoT), digital twin (DT), artificial intelligence (AI), cloud computing, etc. This unprecedented paradigm shift brings significant potential for the RFAC to achieve its full potential.

4.1 System architecture

The overall system architecture for implementing the proposed robot control strategy has been shown in Fig. 3. A robot motion control unit was developed and linked with all robots via an ethernet cable in the RFAC. An OPC UA protocol is utilized as the middleware to establish the communication layer between devices.

4.2 Motion control unit

In this functional module, a workstation is used as a control terminal where robot motion control software is developed and deployed. OPC UA server/client modules are built within the software, which allows the control terminal to establish the real-time data exchange with the industrial robots for sending/receiving data. The data sent

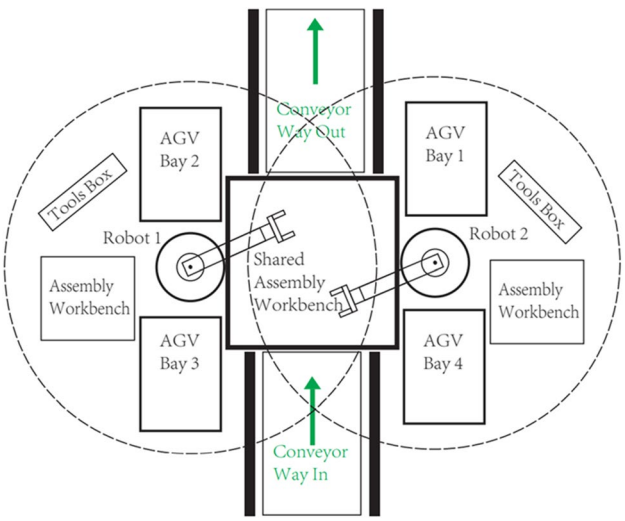

Fig. 6 Typical 2-robot RFAC configuration

from the terminal to the robots consists of the motion commands that the industrial robots need for executing the assembly tasks. The data sent from the robots back to the terminal involve execution confirmation for the expected movements.

4.3 Physical manipulators

Depending on the nature of the RFAC's configuration, there can be a single or multiple industrial robots. The proposed control strategy is suitable for both cases, although the definition of the RFAC requires at least the capability to control multiple robots. In this study, two industrial collaborative robots (cobot) are employed as the physical manipulators to conduct assembly tasks in the proposed RFAC (Fig. 7). One is URe5 [46] from Universal Robots; another one is AUBO-i5 [47] from AUBO Robotics Technology. Each robot is equipped with a Robotiq 2F85 gripper [48] for conducting the pick-and-place tasks. A portable work bench is placed between two robots, acting as the assembly platform.

Currently, OPC UA is widely supported by major industrial robot vendors supplying the global market. Some of those industrial robots have factory-installed OPC UA server/client modules in the system, and some others accept a Plug-and-Play OPC UA software module that is created by third-party CAE companies, such as Universal Robotics. Besides these two cases, some other industrial robots are developed based on open-source systems (such as Ubuntu), which allows the end-users to develop their own OPC UA server/client modules, such as for the AUBO-i5 robot. The industrial robots, in either of the cases presented here, are supported by this functional module.

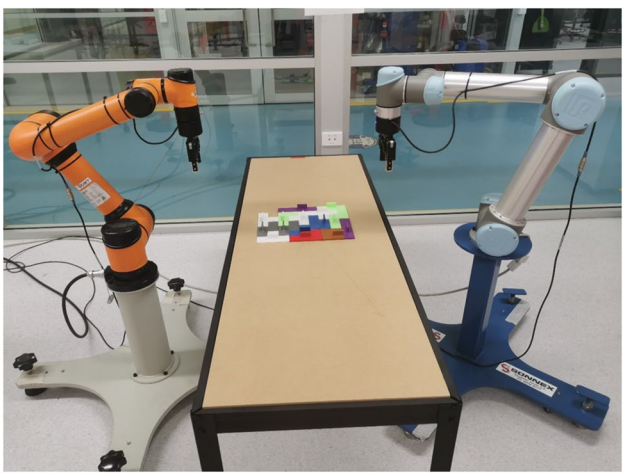

Fig. 7 Industrial cobots for the proposed RFAC

5 Implementation of the proposed robot control strategy–The case study of a real RFAC

The proposed robot motion control approach and the developed framework for RFAC were implemented, integrated, and experimentally verified in a real case study, which has been conducted in the Smart Cobots Assembly Cell (SCAC) at the University of South Australia (UniSA). The SCAC is an Industry 4-oriented RFAC. Its development has been introduced in our previous publications.

5.1 Objectives of the experiment

The objective of the experiment is to validate, in practice, the proposed robot control strategy for industrial robots developed and presented in this paper. Also, it is designed to unravel all complexities and detect all potential integration issues. These issues are documented here, along with the approach to solve them.

It is expected that the proposed approach is capable of executing assembly tasks in the BSO production model. In other words, assembly tasks for different products can be continuously conducted without interruption/stoppage for reprogramming.

5.2 Extent of the experiment

The extent of the experiment is set so that, on top of the programming methodology and core algorithms, only the essential supporting elements will be developed and tested in the experiment presented in this paper. This is because the intention of the case study is to verify the feasibility of the proposed robot control strategy and the algorithms, which are installed in the motion control unit.

On top of that, no other parts or subsystems will be considered in this experiment. However, alternative solutions were applied to ensure the desired processes and experiments could be conducted. For example, material handling systems, such as feed-in devices, conveyors, sensors, PLC, etc., were not developed/integrated for the test at this stage. Instead, the test product is inserted and removed by hand. The signals that are supposed to be sent by the sensors or PLC of material handling system were manually entered into the testing system. The fixtures and jigs were not considered nor implemented in the experiment at this stage.

The robots are able to assemble a test product in this case study. This test product is versatile enough to demonstrate feasibility of the concept and the programming and control strategy. The test product is a Tetris-like puzzle developed to test the smart assembly algorithms, which only require the Pick-and-Place capability of the industrial robots. Figure 8 shows a few different configurations of the developed test product.

Although the test product and assembly tasks are a simplified version compared to those of a real product, they pose the same level of challenges to test the RFAC's capacity to achieve the seamless programme change on the fly and I4's requirement of BSO.

5.3 Robot programming and configuration

As discussed in Sect. 3.2, the programme script that runs in the robots is not designed for executing any specific task. Instead, the programme will enable robots to stay online and wait for the motion commands from the motion control module. The general process flow is the same for both robots. However, the codes of the programme for each robot are slightly different as the two robots use different scripting languages with different syntax. URe5 uses URscript [49], which is quite similar to Python, and the Aubo-i5 uses Lua language.

Another key work for configuring the robots is to set up the OPC UA server/client modules. For URe5, the work is relatively easy as there is a programme called URCap available from Rocketfarm. After the URCap is installed, the variables can be defined within the functional module. The robot's programme can receive and send the data via the established OPC communication layer.

On the other hand, the desired OPC UA modules for AUBO-i5 must be developed for the experiment as no such application is available in the market. The challenge here is that the control system of AUBO-i5 does not directly support OPC UA protocol. The workaround was an SQLite database was deployed between the developed OPC UA server and the robot's programme so that the data can be transferred via the database.

The programme of OPC UA server for AUBO-i5 is written in Python, which is placed into the local hard drive of the AUBO-i5's controller (an Ubuntu system). It needs to be executed in the terminal before running the scripting interface programme via the teach pendant. The logic for the OPC UA module is quite straightforward. It starts with defining local variables and activating OPC UA server by using a free OPC UA library in Python. The function of the main body of the programme is to fetch data from the motion control software and then update the relevant variables in the database. The robot's programme can then obtain the data for the expected robot movement.

5.4 Robot motion control software

In the motion control unit, the motion control software has been developed in Python. A standalone workstation is employed for deploying the software, because a unique IP address is required for the software to build the communication layer over the OPC UA protocol.

At the beginning of the Python script, the algorithm of the robot motion control software will initialise the programme by defining the local variables. Also, OPC UA server/client modules are started up and the variables/tags are created to establish the communication connection with the robots.

As discussed in Sect. 3.2, the assembly plan for each task will be packed into a task execution file. In the case study, the file is saved in.csv format, which is easy to read and edit.

It is worthwhile to mention that, in the task execution file, the robot's TCP's orientation values (Rx, Ry, Rz) will be defined based on roll-pitch-yaw (RPY) model. However, the TCP orientation system of UR5e is developed based on rotation vector (RV) model, instead of using the RPY model that AUBO-i5 employs. Thus, a function is defined within the Python script for converting the TCP's orientation from RPY model to RV model, and as a result, the angle values can be converted into RV model before being sent to the UR5e (Fig. 9).

5.5 Execution of the experiment

As described above, a dual-robot RFAC has been established and configured. Before running the test, both robots need to be calibrated based on a common coordinate system, where

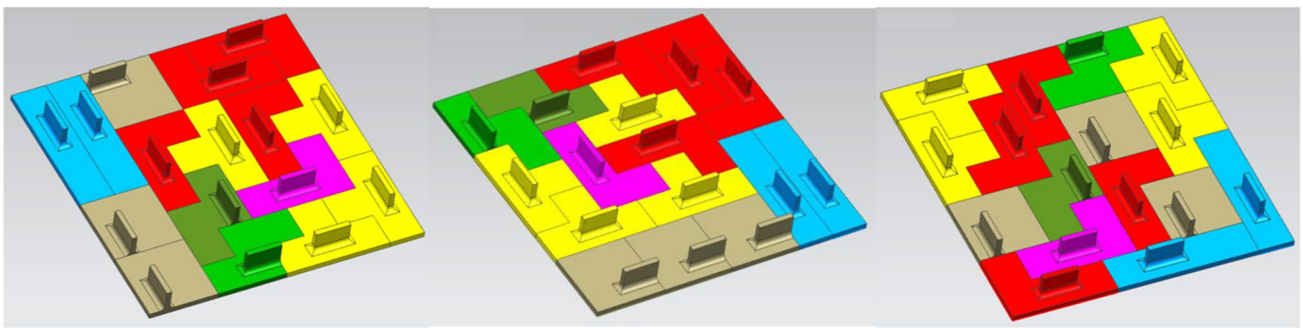

Fig. 8 Three different configurations of the Tetris-like product

the origin for the robots is selected at the central point of the work bench. This needs to be done for the experimental setting so that the coordinates of any target point work for both robots are established relative to a set invariant reference. In an industrial setting, the calibration needs are less onerous once the robots are bolted in their set place.

In the experiment, three task execution files were developed for the Tetris-like test products that have different configurations. Figure 10 shows the developed feed-in tray and the configuration of the test product A that is expected to be assembled. The parts in the raw material feed-in tray have been numbered from 1 to 16.

A task execution file (Fig. 11) is developed for conducting the assembly task of the test product A, which can be fed into the developed robotic control algorithm. In the algorithm, a function is developed to read the task execution file line by line to extract the data. Then, the data will be stored into a pre-defined array before being sent to the robot. Table 2 shows a pseudocode for the task execution file extraction function in the robotic control algorithm.

To be specific, in the column A in the file, the number indicates what robot the current command is sending to. In this case, there are 2 robots in the RFAC. Thus, number 1 indicates UR5e robot, and number 2 indicates Aubo-i5 robot. For example, the number is 1 in the cell 3A in Fig. 11, meaning the row 3 is the command for robot 1—UR5e robot.

The column B is a description that shows the information of the motion command. In column C of the file, Motion Type, the number indicates the desired robotic motion. Each robot will know what motion it is about to conduct once the number of the motion type is received. For instance, motion type number is 1 in the cell 3C. It means that robot

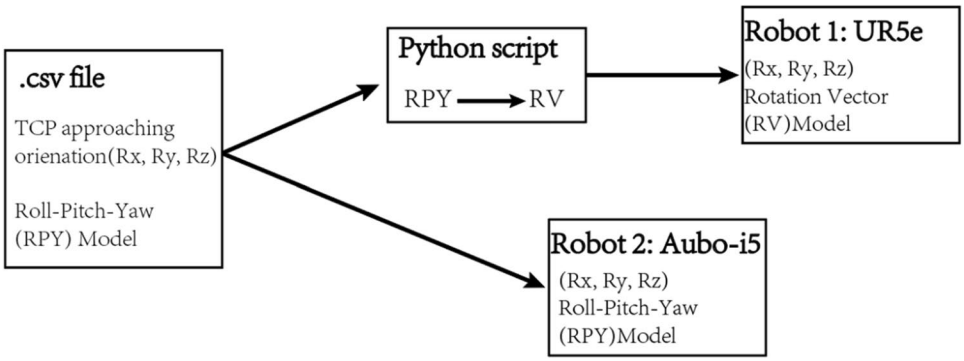

Fig. 9 Robot orientation conversion

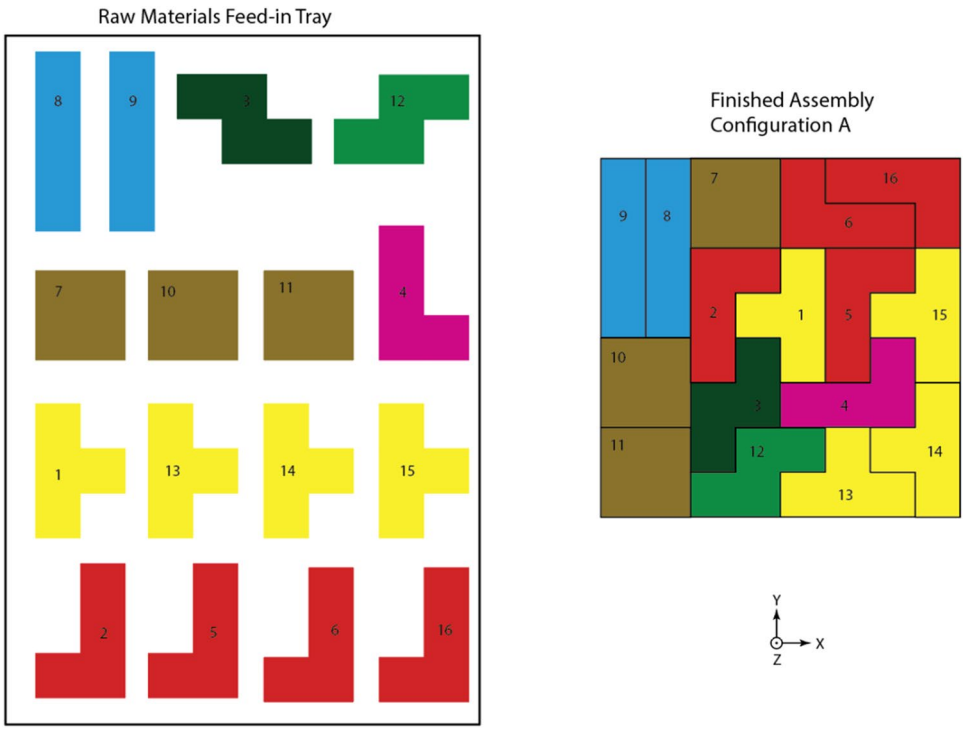

Fig. 10 Feed-in tray and the configuration-A test product

Fig. 11 Task execution file for configuration-A test product

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Robot	Motion	Motion	Pick-up Points						Drop-off Points					
2	Indicator	Description	Type	Pos X	Pos Y	Pos Z	Ori RX	Ori RY	Ori RZ	Pos X	Pos Y	Pos Z	Ori RX	Ori RY	Ori RZ
3	1	Go to standby	1	0.000	0.000	0.000	180	0	-180	0.000	0.000	0.000	180.0	0.0	-180.0
4	2	Go to standby	1	0.000	0.000	0.000	180	0	-180	0.000	0.000	0.000	180.0	0.0	-180.0
5	1	Pick n' Place Part 1	40	-0.680	-0.125	0.150	180	0	-180	0.000	0.000	0.150	180.0	0.0	0.0
6	2	Pick n' Place Part 2	40	-0.645	-0.250	0.150	180	0	-180	-0.070	0.000	0.150	180.0	0.0	0.0
7	1	Pick n' Place Part 3	40	-0.555	0.160	0.150	180	0	-180	-0.035	-0.070	0.150	180.0	0.0	90.0
8	2	Pick n' Place Part 4	40	-0.440	0.000	0.150	180	0	-180	0.035	-0.070	0.150	180.0	0.0	-90.0
9	1	Pick n' Place Part 5	40	-0.565	-0.250	0.150	180	0	-180	0.035	0.000	0.150	180.0	0.0	0.0
10	2	Pick n' Place Part 6	40	-0.485	-0.250	0.150	180	0	-180	0.035	0.070	0.150	180.0	0.0	90.0
11	1	Pick n' Place Part 7	40	-0.680	0.000	0.150	180	0	-180	-0.070	0.105	0.150	180.0	0.0	-180.0
12	2	Pick n' Place Part 8	40	-0.680	0.160	0.150	180	0	-180	-0.105	0.070	0.150	180.0	0.0	-180.0
13	1	Pick n' Place Part 9	40	-0.635	0.160	0.150	180	0	-180	-0.140	0.070	0.150	180.0	0.0	-180.0
14	2	Pick n' Place Part 10	40	-0.600	0.000	0.150	180	0	-180	-0.140	-0.035	0.150	180.0	0.0	-180.0
15	1	Pick n' Place Part 11	40	-0.520	0.000	0.150	180	0	-180	-0.140	-0.105	0.150	180.0	0.0	-180.0
16	2	Pick n' Place Part 12	40	-0.440	0.160	0.150	180	0	-180	-0.035	-0.105	0.150	180.0	0.0	-180.0
17	1	Pick n' Place Part 13	40	-0.600	-0.125	0.150	180	0	-180	0.035	-0.140	0.150	180.0	0.0	-90.0
18	2	Pick n' Place Part 14	40	-0.520	-0.125	0.150	180	0	-180	0.105	-0.105	0.150	180.0	0.0	0.0
19	1	Pick n' Place Part 15	40	-0.440	-0.125	0.150	180	0	-180	0.105	0.000	0.150	180.0	0.0	0.0
20	2	Pick n' Place Part 16	40	-0.405	-0.250	0.150	180	0	-180	0.070	0.105	0.150	180.0	0.0	-90.0

Table 2 Pseudocode for the task execution file extraction function

```
array1 # Defining one array for storing data
array2 # Defining another array for storing data
open task execution file:
data = task execution file # Open and read the task execution file
for a row in data: # iterating all rows in the spreadsheet
for i in range (0, 16): # iterating all columns in each row
array.append(row[i]) # store the data into the array for each row
array2.append(array1) # store all rows into another array
return(array2)
```

needs to move from the original position to the stand-by point; and the number is 40 in the cell 5C, meaning the robot moves from the stand-by point to the desired pick-up position to grip a part, followed by dropping it off to the target point. The numbers in the column C are just identifiers for indicating the specific motion definitions, which have been well defined in the script of the robotic programme (refers to Sect. 3.2). That is why the robot will know the exactly desired motion while the number is received. Despite many of the motion codes that have been defined and are available, only the code numbers 1 and 40 are used for the validation tests.

Subsequently, in the columns D to O, target point coordinate and approaching orientation are provided for supporting the robot motion. Specifically, the values in Pick-up point columns (columns D to I) are the coordinates and end-effector's approaching orientation of those 16 parts in the feed-in tray. Similarly, the values in the columns of Drop-off points (columns J to O) are the coordinates and end-effector's approaching orientation for the parts in the finished assembly.

In the row 5, for instance, the numbers in cells of 5D to 5F define the coordinate $(-0.680, -0.125, 0.150)$ of the point where the end-effector of UR5e needs to move to grip the part 1, with an approaching orientation of $(180, 0, -180)$ that is indicated by the numbers in the cells of 5G to 5E. Following that, the robot needs to drop off the part 1 to a desired location that the coordinate and approaching orientation are defined by the numbers in the cells of 5 J to 5O.

It is worth to mention that the configuration of both robots at stand-by point has been set in the robot motion script. Thus, coordinates and orientation are not required for the "go to stand-by" in the task execution file.

Figure 12 shows stand-by points selected for both robots, which are an operational strategy for collision avoidance between robots. The point S1 is for the URe5, and S2 is for the AUBO-i5. At the beginning of the test, the robots move to their stand-by position separately by executing the first two lines of the commands in the task execution file.

After that, the following 16 lines of the commands are sent to the target robot respectively, line by line, to complete the assembly of the Tetris-like product that consists of 16 parts (refer to Fig. 11).

After the total of 16 parts is moved to the expected positions, the set assembly task is finished. As indicated in the flowchart (Fig. 5), the algorithm will jump back to the beginning of the loop to wait for a new task execution file. In the meantime, the finished product will be moved away from the work bench by the material handling system. A new feed-in tray with parts will be set in position while the task execution file is received by the motion control software.

In the test, the other two task execution files are continuously run for assembling another two configurations of the product, which simulates the processes of the automated batch-size-of-one production mode. Figures 13, 14, 15, and 16 detail the task execution files and the configuration for the products B and C.

A RFID or a QR code can be attached to the feed-in tray, which contains the name of expected task execution file. By scanning the RFID tag or the QR code, the motion control algorithm is triggered to access and activate the desired task execution file and conduct the new assembly task. However, in the test, the name of the new task execution file will be manually entered into the system, at this stage of the experiment, after the new feed-in tray is in position. Therefore,

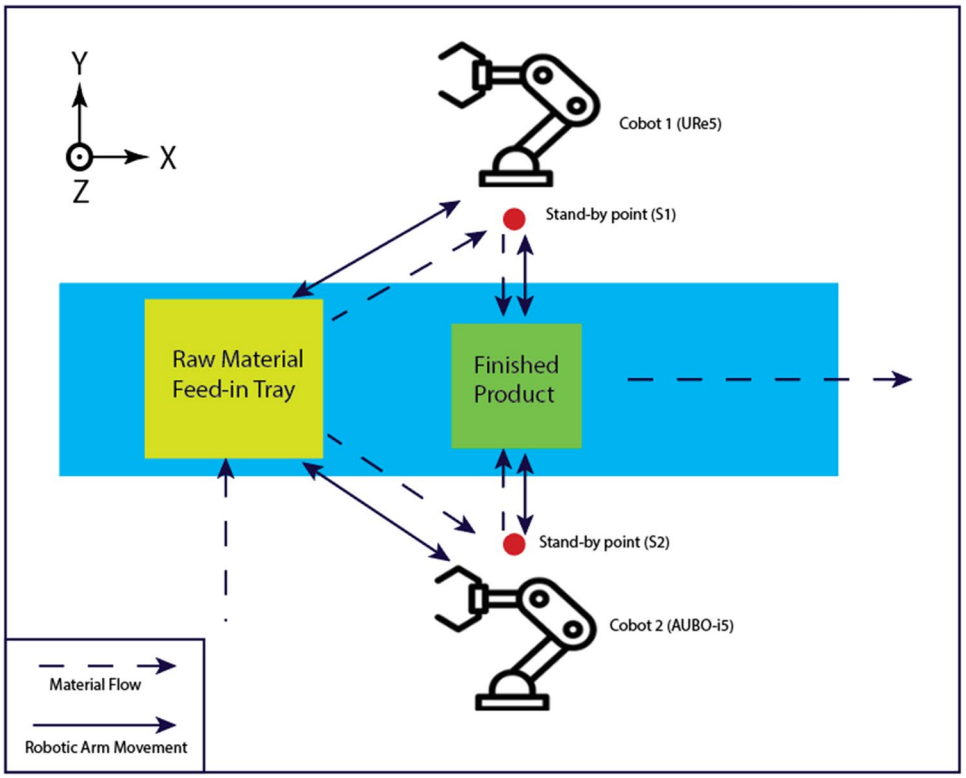

Fig. 12 The demonstration of the processes of the physical execution module

Fig. 13 Task execution file for configuration-B test product

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Robot	Motion	Motion	Pick-up Points						Drop-off Points					
2	Indicator	Description	Type	Pos X	Pos Y	Pos Z	Ori RX	Ori RY	Ori RZ	Pos X	Pos Y	Pos Z	Ori RX	Ori RY	Ori RZ
3	1	Go to standby	1	0.000	0.000	0.000	180	0	-180	0.000	0.000	0.000	180	0	-180
4	2	Go to standby	1	0.000	0.000	0.000	180	0	-180	0.000	0.000	0.000	180	0	-180
5	1	Pick n' Place Part 2	40	-0.645	-0.250	0.150	180	0	-180	0.000	0.000	0.150	180	0	-90
6	2	Pick n' Place Part 1	40	-0.680	-0.125	0.150	180	0	-180	0.000	0.035	0.150	180	0	-90
7	1	Pick n' Place Part 4	40	-0.440	0.000	0.150	180	0	-180	-0.070	0.000	0.150	180	0	-180
8	2	Pick n' Place Part 16	40	-0.405	-0.250	0.150	180	0	-180	0.000	0.105	0.150	180	0	-90
9	1	Pick n' Place Part 3	40	-0.555	0.160	0.150	180	0	-180	-0.070	0.105	0.150	180	0	-180
10	2	Pick n' Place Part 12	40	-0.440	0.160	0.150	180	0	-180	-0.140	0.070	0.150	180	0	-90
11	1	Pick n' Place Part 15	40	-0.440	-0.125	0.150	180	0	-180	-0.140	0.000	0.150	180	0	-180
12	2	Pick n' Place Part 13	40	-0.600	-0.125	0.150	180	0	-180	-0.105	-0.070	0.150	180	0	-90
13	1	Pick n' Place Part 14	40	-0.520	-0.125	0.150	180	0	-180	0.000	-0.070	0.150	180	0	-90
14	2	Pick n' Place Part 11	40	-0.520	0.000	0.150	180	0	-180	-0.140	-0.105	0.150	180	0	-180
15	1	Pick n' Place Part 10	40	-0.600	0.000	0.150	180	0	-180	-0.070	-0.105	0.150	180	0	-180
16	2	Pick n' Place Part 7	40	-0.680	0.000	0.150	180	0	-180	0.000	-0.105	0.150	180	0	-180
17	1	Pick n' Place Part 8	40	-0.680	0.160	0.150	180	0	-180	0.070	-0.070	0.150	180	0	-180
18	2	Pick n' Place Part 9	40	-0.635	0.160	0.150	180	0	-180	0.105	-0.070	0.150	180	0	-180
19	1	Pick n' Place Part 6	40	-0.485	-0.250	0.150	180	0	-180	0.105	0.035	0.150	180	0	-180
20	2	Pick n' Place Part 5	40	-0.565	-0.250	0.150	180	0	-180	0.070	0.070	0.150	180	0	0

automation can be achieved in the process of job change. As long as the new task execution file is developed and accessible for the robots, there will be no interruption occurring between the assembly of different products.

5.6 System repeatability and accuracy

Repeatability and accuracy are key measurable characteristics for industrial robots, which significantly impact the robot system on the effectiveness of task execution [50]. The performance of repeatability and accuracy of an industrial robot commonly depends on its design, construction, operation, and maintenance history, and they can be different for each robot brand and for each particular robot in a series.

In terms of the proposed robot control strategy, the repeatability and accuracy of the system will be exactly the same as the performance capability for each of the robots. In other words, the proposed method does not affect the performance of the robots' repeatability and accuracy. This is because the original robot controller, the control, and motion algorithm of both robots, as well as the structure and factory settings, are not changed nor affected at all. The role of the developed flexible robot control strategy is strictly interesting the capacity and facility to sending the task execution file to the robot controllers on the fly, which in turn achieves the desired BSO production model. Thus, there was no requirement nor justification to design and conduct any extra test for validating the repeatability and accuracy of the system.

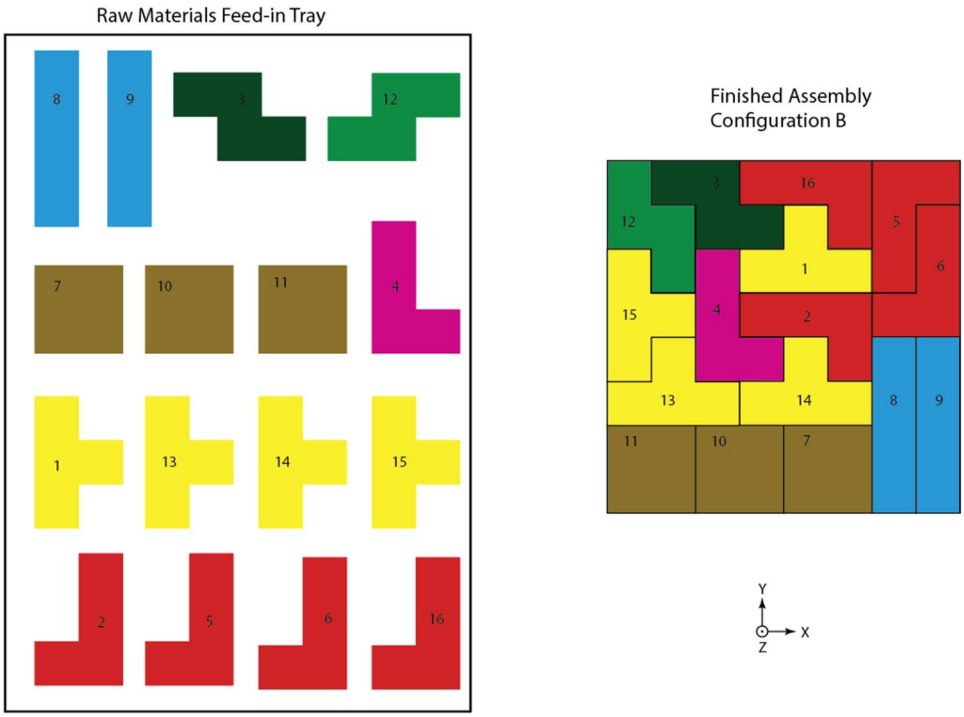

Fig. 14 Feed-in tray and the configuration-B test product

Fig. 15 Task execution file for configuration-C test product

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Robot	Motion	Motion	Pick-up Points						Drop-off Points					
2	Indicator	Description	Type	Pos X	Pos Y	Pos Z	Ori RX	Ori RY	Ori RZ	Pos X	Pos Y	Pos Z	Ori RX	Ori RY	Ori RZ
3	1	Go to standby	1	0.000	0.000	0.000	180	0	-180	0.000	0.000	0.000	180	0	-180
4	2	Go to standby	1	0.000	0.000	0.000	180	0	-180	0.000	0.000	0.000	180	0	-180
5	1	Pick n' Place Part 10	40	-0.600	0.000	0.150	180	0	-180	0.000	0.000	0.150	180	0	-180
6	2	Pick n' Place Part 12	40	-0.440	0.160	0.150	180	0	-180	0.070	0.070	0.150	180	0	-180
7	1	Pick n' Place Part 6	40	-0.485	-0.250	0.150	180	0	-180	-0.035	0.035	0.150	180	0	0
8	2	Pick n' Place Part 3	40	-0.555	0.160	0.150	180	0	-180	-0.070	-0.070	0.150	180	0	-90
9	1	Pick n' Place Part 5	40	-0.565	-0.250	0.150	180	0	-180	0.000	-0.105	0.150	180	0	0
10	2	Pick n' Place Part 14	40	-0.520	-0.125	0.150	180	0	-180	-0.105	0.070	0.150	180	0	90
11	1	Pick n' Place Part 2	40	-0.645	-0.250	0.150	180	0	-180	-0.070	0.000	0.150	180	0	-180
12	2	Pick n' Place Part 1	40	-0.680	-0.125	0.150	180	0	-180	-0.140	0.000	0.150	180	0	-180
13	1	Pick n' Place Part 7	40	-0.680	0.000	0.150	180	0	-180	-0.140	-0.070	0.150	180	0	-180
14	2	Pick n' Place Part 4	40	-0.440	0.000	0.150	180	0	-180	-0.070	-0.140	0.150	180	0	-90
15	1	Pick n' Place Part 16	40	-0.405	-0.250	0.150	180	0	-180	-0.105	-0.175	0.150	180	0	90
16	2	Pick n' Place Part 11	40	-0.520	0.000	0.150	180	0	-180	0.035	-0.105	0.150	180	0	-180
17	1	Pick n' Place Part 8	40	-0.680	0.160	0.150	180	0	-180	0.035	-0.175	0.150	180	0	90
18	2	Pick n' Place Part 9	40	-0.635	0.160	0.150	180	0	-180	0.105	-0.105	0.150	180	0	-180
19	1	Pick n' Place Part 15	40	-0.440	-0.125	0.150	180	0	-180	0.070	-0.035	0.150	180	0	-180
20	2	Pick n' Place Part 13	40	-0.600	-0.125	0.150	180	0	-180	0.105	0.035	0.150	180	0	0

5.7 System error rate

Tests are conducted to validate the reliability and error rate of the proposed robotic control method in three scenarios: (i) run UR5e robot only; (ii) run Aubo-i5 robot only; (iii) run both robots collaboratively. Each test was repeated 25 times. In the test (i), the error rate is 0. All of 25 repeated runs met the expectation. However, in the test (ii), 3 of 25 runs failed. One of the 3 failure runs was due to communication failures between OPC UA servers and clients. The other 2 failed runs were due to mechanical issues at a joint of the Aubo-i5 robot. In test (iii), 1 of 25 runs failed. The issue was also from the Aubo-i5 robot. The results are summarised in Fig. 17.

It is believed that the developed flexible robotic control strategy has demonstrated its desired reliability as no error occurred in the test (i). The failures happened in tests (ii) and (iii) are believed coming from the system errors of the developed OPC UA server and the control system for AUBO-i5 robot. This is being investigated and debugged.

5.8 System efficiency

The test for product A was conducted again to validate the efficiency of the developed flexible robotic control strategy. In comparison with the original robotic control method for both robots, latency time was identified in the developed control method. Although the robot moving speed in each joint is not degraded, latency time occurred between motion commands. Before sending the next motion command, the OPC UA server in the developed control mechanism takes

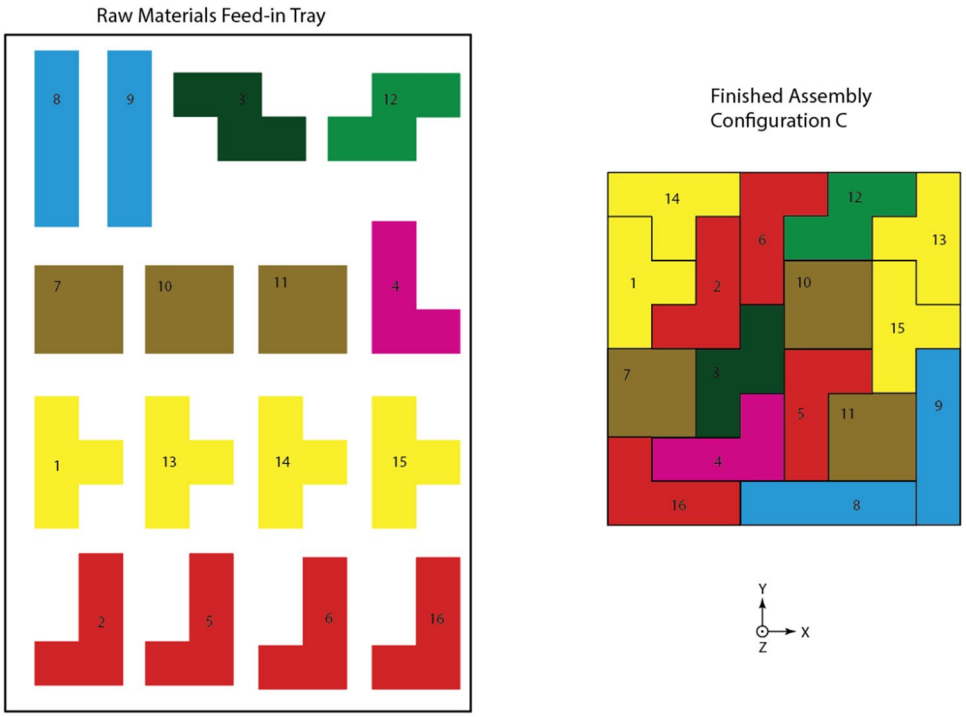

Fig. 16 Feed-in tray and the configuration-C test product

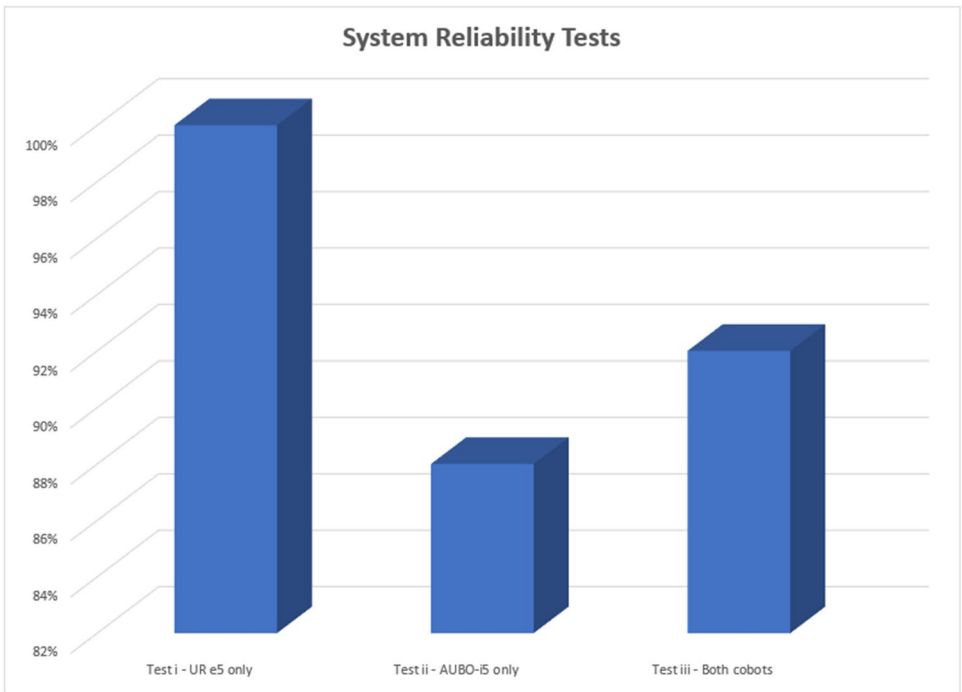

Fig. 17 System reliability test

time to fetch data from the OPC UA server in the robots for confirming the current motion command has been successfully conducted. Tests were conducted for both robots individually, repeated 5 times. Referring to Fig. 11, there are 16 parts in the task execution file for testing product A. Thus, 16 groups of averaged latency time are measured and presented in Fig. 18.

5.9 Software debugging

Software debugging can effectively reduce failure rate of the software components in manufacturing systems at the early stage of the development [51]. Given the testing results in Sects. 5.7 and 5.8, it is believed that software debugging and optimisation can improve the system reliability and reduce the

latency time. Three debugging and optimisation exercises have been done after the tests in subsections 5.7 and 5.8; (i) debug and optimisation for the motion control algorithm; (ii) debug and optimisation for the OPC UA server in Aubo-i5 robot; (iii) debug and optimisation for the motion script for both robots.

Tests are conducted again after the implementation of the debugging actions. It is clear that the system reliability has been improved from the perspective of the performance of Aubot-i5 robot (Fig. 19), and the system latency time has been reduced by 15% on average (Fig. 20). The works (i) and (ii) optimised the connectivity between OPC UA server of Aubo-i5 robot and the motion control mechanism. The works (i) and (iii) contributed the improvements in reducing the latency time.

To sum up, the developed RFAC includes two physical robots and the robotic motion control system. The robotic motion control algorithm was verified by continuously conducting the assembly tasks in various configurations of the Tetris-like product three times. High automation and flexibility are achieved throughout the processes. No interruptions occurred due to changing tasks. Therefore, we can confidently say that the proposed robotic motion control strategy is able to achieve the desired BSO production mode in our implementation of the RFAC.

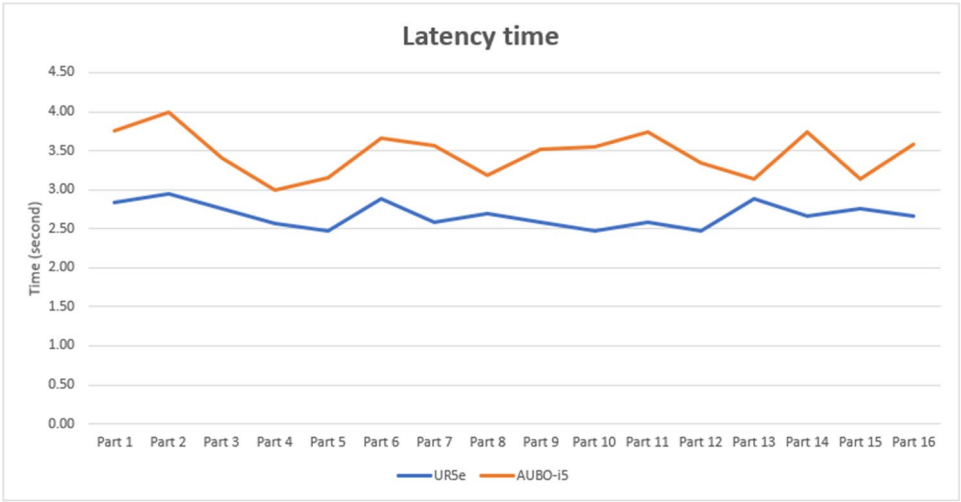

Fig. 18 Latency time between sending each motion command in proposed control method

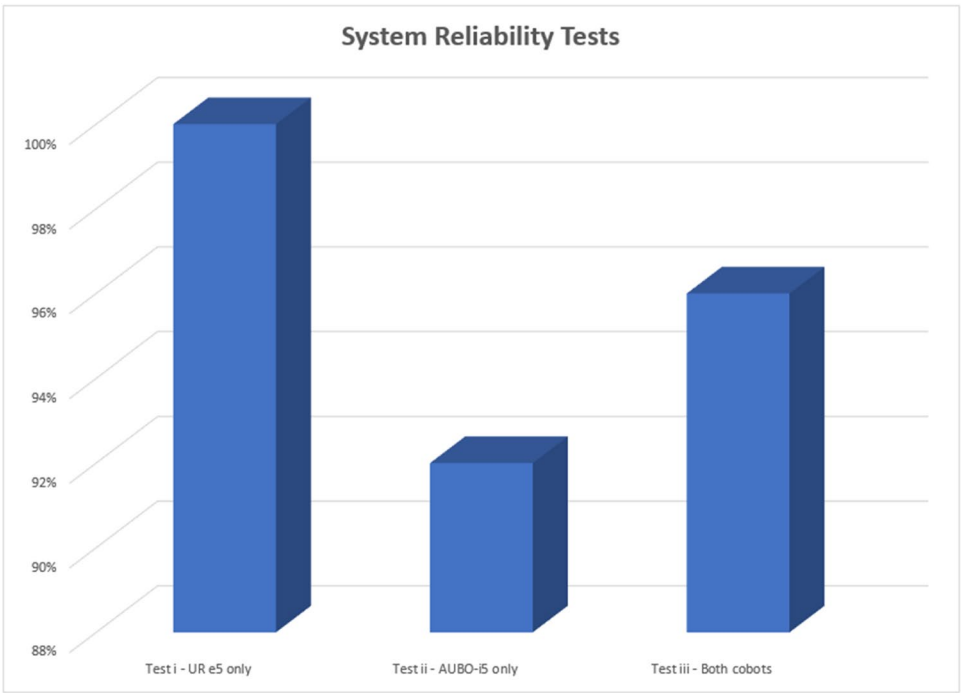

Fig. 19 System reliability test after software debug and optimisation

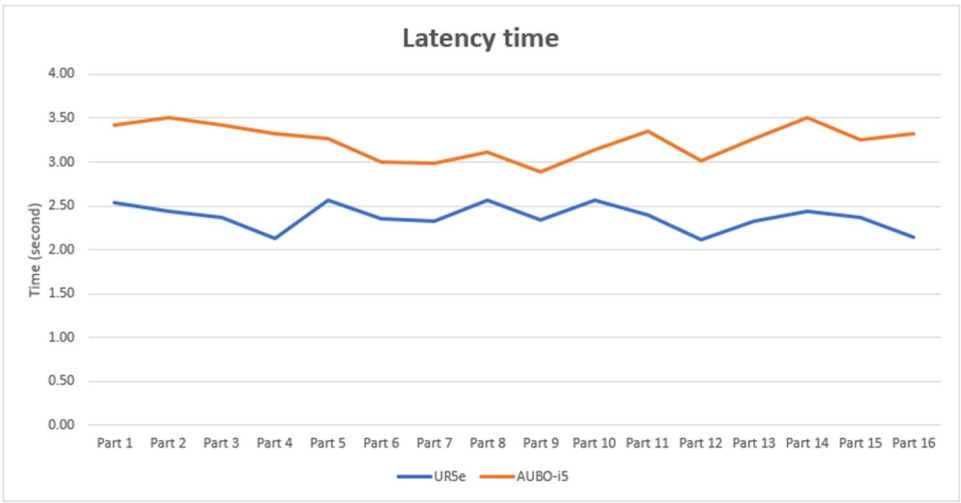

Fig. 20 Latency time between sending each motion Command in proposed control method after software debug and optimisation

6 Discussion

The robotic flexible assembly cell developed in this study has successfully demonstrated, in practice, that the proposed strategy for task scheduling, programming, and control of industrial robots is viable and works for one of the most constrained problems in robotics.

6.1 Advantages of the proposed method

The main advantage and uniqueness in this research are that the proposed method enables automated task change process in RFAC. This permits the RFAC to achieve a higher level of flexibility and adaptiveness. Also, the proposed method is suitable for controlling both a single robot and multiple robots to carry out assembly tasks. Critically, it does not need any work on the development or modification of robots' existing control cabinet, which makes the strategy more generic and much easier to be deployed for a broad range of robots and robot tasks.

6.2 Limitations of the proposed method

The main limitation of the proposed method is that, at this stage and without extra systems (like vision or sophisticated sensors), it requires explicit position of the parts for both components and finished product. As shown in the task execution file, coordinates are required for both pick-up and drop-off positions for the parts, which makes the method limited to structured robot applications, such as PCB or mobile phone assembly. It is possible to solve this issue, in the future, with a vision system, that can be integrated into the RFAC. This is not part of the current implementation and can be considered as part of future work.

Another limitation is the latency that occurred between each motion control command that the control mechanism sent to the robots. This is reasonable at this stage of the research, considering that the main intention of this paper is to validate the proof of concept for the proposed robot control strategy. Thus, the latency time is acceptable at the moment. However, it is clear that the latency time needs to and can be reduced by optimising the proposed control algorithm and OPC UA server integration for both robots, which is a proposed work in the future, before the strategy can be implemented in an industrial setting.

6.3 Challenges in the research

Significant challenges were identified, and solutions were developed, as follows:

- Two robots from different vendors were used. These required specific development steps for a unified communication interface. Despite OPC UA being chosen to build the communication layer for the developed RFAC, neither of these two robots have OPC UA server/client function by default. Thus, the works for developing OPC UA server/client module for both robots were conducted (refer to Section 5.3). It is worthwhile to note that, comparing with UR5e, the work for Aubo-i5 was much more challenging. Due to its relatively low market share, there is little technical information available for integration and translation. This required extensive consulting with colleagues from industry and the original manufacturer. A software development kit (SDK) was finally developed for enabling OPC UA communication with an Aubo-i5 robot.
- The product of the assembly process needed to be representative for a family of products, as well as being simple

to procure. A family of mobile phones is representative—similar footprint, similar parts, but the result is a large combination of capabilities, depending on the actual model. The Tetris-like puzzle introduced in this research emulates the combinatorial complexity of such products, enabling a huge number of combinations of end models, while being easy and fast to make and manipulate.
- The control of the robots needed to be central so that the granularity of the programming will not be limited to address the assembly of a product but can evolve down to the level of a task (e.g., pick-and-place).
- The RFAC is resilient. It is possible to take one robot offline (due to malfunction or maintenance) and continue the assembly with just one robot. This was briefly checked and confirmed as a major capability. This will be described in detail in a future article.

The non-technical challenges in implementing this robot control strategy and the system that demonstrates it in practice were numerous. Funding is always a limitation, which led to only implementing the critical elements that make the difference between classical (albeit fancy and flashy solutions and demonstrations) and a paradigm shifting strategy. The experiments, besides the critical components, were minimalistic (e.g., material and product handling to and from the RFAC). A pandemic, with laboratories locked down and access cancelled just as the critical mass of robots, servers, protocols, and settings were established and connected, was, in retrospect, a blessing in disguise. The remote access protocols and infrastructure, programming, debugging, and checking capabilities, which needed to be developed to permit work to continue during lockdowns, became critical and are expected in the I4 context.

7 Conclusions and recommendations for future work

In this paper, a novel strategy for controlling industrial robots was proposed and the details of the implementation discussed. The strategy was validated in a robot flexible assembly cell, a multi-robot cell, conceived to assemble products from a family of similar products. The system is designed to enable seamless and uninterrupted introduction of new and removal of old products from the mix, as the market evolution dictates. When a new product is being added to the family of items to be assembled in the workcell, the respective code is loaded into the central control unit and made available to be used when required.

This robot control strategy enables the desired BSO production mode, i.e., possibility to switch from one assembled item to another, continuously and without interruption, which is one of the core requirements of the Industry 4 philosophy. The proposed approach is designed to be used for workcells with multiple robots but works perfectly with only one robot.

Regarding future developments of the concept, the following aspects are being investigated:

- A bi-directional digital twin (BDT) is being developed for the RFAC, which will be integrated with the robotic control mechanism. Real and extensive data from the BDT from the operation of the cell will permit the development of novel maintenance strategies.
- Proactive maintenance strategy is being developed and will be integrated with the current system as well. The developed strategy will employ machine learning method, which forms the biological component for the RFAC with respect to building the SCAC as a genuine Industry 4 facility.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s00170-023-11246-y.

Acknowledgements This research work was accomplished with the support of the Australian Government Research Training Program (RTP).

Author contribution All the authors contributed to the study conception and design. Case study implementation, data collection and analysis were performed by Ziyue Jin. The first draft of the manuscript was written by Ziyue Jin, and all the authors reviewed, commented, and edited the manuscript. All the authors read and approved the final manuscript.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions.

Data availability Data sharing is not applicable to this article as no datasets were generated or analysed during the current study.

Declarations

Ethics approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

Competing interests The authors declare no competing interests.

References

1. ElMaraghy HA (2005) Flexible and reconfigurable manufacturing systems paradigms. Int J Flex Manuf Syst. https://doi.org/10.1007/s10696-006-9028-7
2. Cohen Y, Faccio M, Galizia FG, Mora C, Pilati F (2017) Assembly system configuration through Industry 4.0 principles: the expected change in the actual paradigms. IFAC-Papers OnLine. https://doi.org/10.1016/j.ifacol.2017.08.2550
3. Bortolini M, Galizia FG, Mora C (2018) Reconfigurable manufacturing systems: literature review and research trend. J Manuf Syst. https://doi.org/10.1016/j.jmsy.2018.09.005
4. Koren Y, Shpitalni M (2010) Design of reconfigurable manufacturing systems. J Manuf Syst. https://doi.org/10.1016/j.jmsy.2011.01.001
5. Rosati G, Faccio M, Carli A, Rossi A (2013) Fully flexible assembly systems (F-FAS): a new concept in flexible automation. Assem Autom. https://doi.org/10.1108/01445151311294603
6. Zhang S, Li S, Wang H, Li X (2022) An intelligent manufacturing cell based on human-robot collaboration of frequent task learning for flexible manufacturing. Int J Adv Manuf Technol. https://doi.org/10.1007/s00170-022-09005-6
7. Huang Z, Jowers C, Kent D, Dehghan-Manshadi A, Dargusch MS (2022) The implementation of Industry 4.0 in manufacturing: from lean manufacturing to product design. Int J Adv Manuf Technol. https://doi.org/10.1007/s00170-022-09511-7
8. Abdulhameed O, Al-Ahmari A, Ameen W, Mian SH (2019) Additive manufacturing: challenges, trends, and applications. Adv Mech Eng. https://doi.org/10.1177/1687814018822880
9. Lasi H, Fettke P, Kemper HG, Feld T, Hoffmann M (2014) Industry 4.0. Bus Inf Syst Eng. https://doi.org/10.1007/s12599-014-0334-4
10. Xu LD, Xu EL, Li L (2018) Industry 4.0: state of the art and future trends. Int J Prod Res. https://doi.org/10.1080/00207543.2018.1444806
11. Schwab K (2016) The fourth industrial revolution. World Economic Forum, Geneva
12. Faccio M, Bottin M, Rosati G (2019) Collaborative and traditional robotic assembly: a comparison model. Int J Adv Manuf Technol. https://doi.org/10.1007/s00170-018-03247-z
13. Shi J, Menassa R (2010) Flexible robotic assembly in dynamic environments. In: Proceedings of the 10th Performance Metrics for Intelligent Systems Workshop. Association for Computing Machinery, New York, NY. https://doi.org/10.1145/2377576.2377626
14. Caldeira R, Honnungar S (2018) Feasibility study for converting traditional line assembly into work cells for termination of fiber optics cable. In AIP Conf Proc. https://doi.org/10.1063/1.5029623
15. Vick A, Krueger J (2018) Using OPC UA for distributed industrial robot control. In: ISR 2018 50th International Symposium on Robotics. VDE, Munich, Germany, pp 1–6
16. Cohen Y, Faccio M, Pilati F, Yao X (2019) Design and management of digital manufacturing and assembly systems in the Industry 40 era. Int J Adv Manuf Technol. https://doi.org/10.1007/s00170-019-04595-0
17. Gong X, Jiao R, Jariwala A, Morkos B (2021) Crowdsourced manufacturing cyber platform and intelligent cognitive assistants for delivery of manufacturing as a service: fundamental issues and outlook. Int J Adv Manuf Technol. https://doi.org/10.1007/s00170-021-07789-7
18. Wilkesmann M, Wilkesmann U (2018) Industry 4.0–organizing routines or innovations? VINE J Inf Knowl Manag Syst. https://doi.org/10.1108/VJIKMS-04-2017-0019
19. Yuan F, Shen X, Wu J, Wang L (2022) Design of mobile phone automatic assembly system based on machine vision. J Physics 2284(1):012012. https://doi.org/10.1088/1742-6596/2284/1/012012. (**IOP Publishing**)
20. Zhang JZ, He YY, Li J (2012) An application of machine vision in automatic mobile-phone lens assembly equipment. Appl Mech Mater 130:3543–3547. https://doi.org/10.4028/www.scientific.net/AMM.130-134.3543. (**Trans Tech Publications Ltd**)
21. Lee K, Jung M (2015) Overseas factories, domestic employment, and technological hollowing out: a case study of Samsung's mobile phone business. Rev World Econ. https://doi.org/10.1007/s10290-015-0219-8
22. Wilhelm M, Hutchins M, Mars C, Benoit-Norris C (2015) An overview of social impacts and their corresponding improvement implications: a mobile phone case study. J Clean Prod. https://doi.org/10.1016/j.jclepro.2015.04.025
23. Pathak T (2018) India Imported $13 Billion Worth of Mobile Phone Components in 2018. CounterPoint. https://www.counterpointresearch.com/india-imported-13-billion-worth-mobile-phone-components-2018/. Accessed 17 Feb 2023
24. Abd K, Abhary K, Marian R (2014) Simulation modelling and analysis of scheduling in robotic flexible assembly cells using Taguchi method. Int J Prod Res. https://doi.org/10.1080/00207543.2013.867082
25. Ruan LZ, Tao WD, Peng JL, Chen Y (2011) Assembly line balancing and simulation optimization of the TM921C mobile phone assembly line of DFTX company. In: 2011 IEEE 18th International Conference on Industrial Engineering and Engineering Management. IEEE, Changchun, China. https://doi.org/10.1109/ICIEEM.2011.6035249
26. Zheng J, Zhang X, Li H, Huang Y (2022) High-efficiency transmission of industrial heterogeneous data in a typical mobile phone assembly production line. In 2022 4th Asia Pacific Information Technol Conf. https://doi.org/10.1145/3512353.3512372
27. Chin KS, Ratnam MM, Mandava R (2003) Force-guided robot in automated assembly of mobile phone. Assem Autom. https://doi.org/10.1108/01445150310460123
28. Chin KS, Ratnam MM, Mandava R (2004) Force-guided robot in automated assembly of mobile phone: overcoming component misalignment. Assem Autom. https://doi.org/10.1108/01445150410517192
29. De Almeida AT, Nunes UC, Dias JM, Araujo HJ, Batista J (1990) A distributed system for robotic multi-sensor integration. Ind Metrol. https://doi.org/10.1016/S0921-5956(90)80009-K
30. Maoudj A, Bouzouia B, Hentout A, Kouider A, Toumi R (2019) Distributed multi-agent scheduling and control system for robotic flexible assembly cells. J Intell Manuf 30(4):1629–1644. https://doi.org/10.1007/s10845-017-1345-z
31. Edan Y, Berman S, Boteach E, Mendelson (2013) Distributed multi-robot assembly/packaging algorithms. Intell Autom Soft Comput. https://doi.org/10.1080/10798587.2004.10642888
32. Rizzi AA, Gowdy J, Hollis RL (2001) Distributed coordination in modular precision assembly system. Int J Robotics Res. https://doi.org/10.1177/02783640122068128
33. Vick A, Vonásek V, Pěnička R, Krüger J (2015) Robot control as a service—towards cloud-based motion planning and control for industrial robots. In 2015 10th Int Workshop Robot Motion Control (RoMoCo). https://doi.org/10.1109/RoMoCo.2015.7219710
34. Blankenburg J, Banisetty SB, Alinodehi SPH, Fraser L, Feil-Seifer D, Nicolescu M, Nicolescu M (2017) A distributed control architecture for collaborative multi-robot task allocation. In 2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids). https://doi.org/10.1109/HUMANOIDS.2017.8246931
35. Tsarouchi P, Makris S, Michalos G, Matthaiakis AS, Chatzigeorgiou X, Athanasatos A, Stefos M, Aivaliotis P, Chryssolouris G (2015) ROS based coordination of human robot cooperative assembly tasks-an industrial case study. Procedia CIrP. https://doi.org/10.1016/j.procir.2015.08.045
36. Hannelius T, Salmenpera M, Kuikka S (2008) Roadmap to adopting OPC UA. In: 2008 6th IEEE International Conference on Industrial Informatics. IEEE, Daejeon, South Korea. https://doi.org/10.1109/INDIN.2008.4618203

37. Kaarlela T, Pieskä S, Pitkäaho T, Solvang WD, Shu B, Arnarson H, Solvang B (2022) Robot cell digital twins as a tool for remote collaboration between organizations. In 2022 IEEE/SICE Int Symp Syst Integr (SII). https://doi.org/10.1109/SII52469.2022.9708902
38. Müller F, Deuerlein C, Koch M (2021) Cyber-physical-system for representing a robot end effector. Procedia CIRP. https://doi.org/10.1016/j.procir.2021.05.071
39. Florescu A, Barabas SA (2020) Modeling and simulation of a flexible manufacturing system—a basic component of industry 4.0. Applied sciences. https://doi.org/10.3390/app10228300
40. Chen IM (2001) Rapid response manufacturing through a rapidly reconfigurable robotic workcell. Robotics Comput-Integr Manuf. https://doi.org/10.1016/S0736-5845(00)00028-4
41. Araiza-Illan D, De San BA, Hongchao F, Shin LY (2019) Augmented reality for quick and intuitive robotic packing re-programming. In 2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI). https://doi.org/10.1109/HRI.2019.8673327
42. Marvel J, Bostelman R, Falco J (2018) Multi-robot assembly strategies and metrics. ACM Comput Surv. https://doi.org/10.1145/3150225
43. Michniewicz J, Reinhart G (2014) Cyber-physical robotics–automated analysis, programming and configuration of robot cells based on cyber-physical-systems. Procedia Technol. https://doi.org/10.1016/j.protcy.2014.09.017
44. Marian RM, Kargas A, Luong LHS, Abhary K (2003) A framework to planning robotic flexible assembly cells. In: 32nd International Conference on Computers and Industrial Engineering. CSIRO, Limerick, Ireland, pp 607–615
45. Abd K, Abhary K, Marian R (2012) Efficient scheduling rule for robotic flexible assembly cells based on fuzzy approach. Procedia CIRP. https://doi.org/10.1016/j.procir.2012.07.083
46. Blankemeyer S, Wiemann R, Posniak L, Pregizer C, Raatz A (2018) Intuitive robot programming using augmented reality. Procedia CIRP. https://doi.org/10.1016/j.procir.2018.02.028
47. Yuan C, Liu G, Zhang W, Pan X (2020) An efficient RRT cache method in dynamic environments for path planning. Robot Auton Syst. https://doi.org/10.1016/j.robot.2020.103595
48. Falco J, Hemphill D, Kimble K, Messina E, Norton A, Ropelato R, Yanco H (2020) Benchmarking protocols for evaluating grasp strength, grasp cycle time, finger strength, and finger repeatability of robot end-effectors. IEEE Robotics Autom Lett 5(2):644–651
49. Margaria T, Schieweck A (2019) The digital thread in industry 4.0. Int Conf Integr Formal Methods. https://doi.org/10.1007/978-3-030-34968-4_1
50. Conrad KL, Shiakolas PS, Yih TC (2000) Robotic calibration issues: accuracy, repeatability and calibration. In: Proceedings of the 8th Mediterranean Conference on Control and Automation, vol 1719. MED2000, Rio, Patras, Greece, pp 1–6
51. Lazarova-Molnar S, Mohamed N (2019) Reliability assessment in the context of industry 4.0: data as a game changer. Procedia Comp Sci. https://doi.org/10.1016/j.procs.2019.04.092