POLITECNICO
MILANO 1863

POLITECNICO DI MILANO
DIPARTIMENTO DI ELETTRONICA, INFORMAZIONE E BIOINGEGNERIA
DOCTORAL PROGRAMME IN INFORMATION TECHNOLOGY

ENHANCING HUMAN-ROBOT COLLABORATION FOR FLEXIBLE MANUFACTURING IN INDUSTRY 4.0

Doctoral Dissertation of:
Riccardo Maderna

Supervisor:
Prof. Paolo Rocco

Tutor:
Prof. Luca Bascetta

The Chair of the Doctoral Program:
Prof. Barbara Pernici

Year 2020 – Cycle XXXIII

"Uno no es por lo que escribe, sino por lo que ha leído"
Jorge Luis Borges

Abstract

THE current global economy is increasingly characterised by quickly changing markets, small batch production, and mass customization. For this reason, companies are required to meet a high level of flexibility to stay competitive. Flexible manufacturing is an approach that allows satisfying the needs of customers while maintaining high quality standards. In this context, advances in industrial robotics and human-robot collaboration are playing a key role to provide companies with an adaptable and powerful tool capable of enhancing efficiency and flexibility of manufacturing processes. Humans leverage their superior cognitive and manipulative skills to perform operations that are difficult to automate or that require high-level decision-making. On the other hand, robots can relieve operators from repetitive and laborious tasks and assist humans in many ways.

This thesis aims to propose a solution to the problem of controlling the production of a flexible manufacturing cell. In particular, the focus has been placed on multi-product assembly lines, consisting of several humans and robots working together in a shared workspace. Versatile scheduling algorithms are needed to organise the complex work-flow and fully exploit the available flexibility, ensuring the optimal use of resources and the smart management of unexpected events. Also, good productivity must be kept regardless of changes in production plans and the intrinsic variability of human behaviour. Thus, the work develops around two main objectives:

1. Monitoring and predicting the evolution of the ongoing human activity in real-time. This is beneficial for task scheduling, as knowing the

progress of the current task allows for better coordination among the agents. Two different strategies have been developed, which account for the fact that humans can perform the same operation in many ways and with different speeds, occasional errors, and short pauses.

2. Dynamically scheduling the tasks of the collaborating agents so as to optimise productivity. In the proposed approach, a Digital Twin based on Timed Petri Nets tracks the state of the collaborative workspace in real-time based on data coming from the robot controllers and the human monitoring unit. The Digital Twin is then used to simulate the future evolution of the system and determine the optimal instructions for humans and robots with a receding horizon approach. This allows dynamically adapting the schedule to the system variability and the occurrence of robot faults.

Additionally, the convenience of introducing collaborative robots in the kitting process has been investigated. Kitting is a key logistic task in flexible manufacturing, which consists in grouping separate items together to be supplied as one unit to the assembly line. At present, it is usually performed manually by human operators, but robots may help to reduce the operator's effort and increase productivity. Thus, an online scheduling algorithm to guide the picking operations of the human and the robot is also presented in this thesis.

A consideration that underlies the entire work is that flexible manufacturing is characterised by frequent production changes, which reflect in changes in the workspace layout and in the operations to be performed. Therefore, requirements on the framework developed in this thesis were ease of set up and update, and strong learning capabilities. This is especially true when human behaviour is considered, which is difficult to fully define a priori and also changes over time, e.g. due to training and fatigue.

Sommario

L'ATTUALE economia globale è sempre più caratterizzata da mercati in rapida evoluzione, produzione a piccoli lotti e personalizzazione di massa. Per questo motivo, le aziende devono raggiungere un elevato livello di flessibilità per rimanere competitive. La produzione flessibile è un approccio che consente di soddisfare le esigenze dei clienti mantenendo elevati standard di qualità. In questo contesto, i progressi nella robotica industriale e la collaborazione uomo-robot stanno giocando un ruolo chiave per fornire alle aziende uno strumento versatile e potente in grado di migliorare l'efficienza e la flessibilità dei processi produttivi. Le persone sfruttano le loro migliori capacità cognitive e manipolative per eseguire operazioni difficili da automatizzare o che richiedono processi decisionali di alto livello. D'altra parte, i robot possono alleviare gli operatori da compiti ripetitivi e faticosi e aiutare le persone in molti modi.

Questa tesi ha l'obiettivo di proporre una soluzione al problema del controllo della produzione di una cella di produzione flessibile. In particolare, l'attenzione è stata posta su linee di assemblaggio multi-prodotto, costituite da diversi umani e robot che lavorano insieme in uno spazio di lavoro condiviso. Sono necessari algoritmi di pianificazione versatili per organizzare il complesso flusso di lavoro e sfruttare appieno la flessibilità disponibile, garantendo l'uso ottimale delle risorse e la gestione intelligente di eventi imprevisti. Inoltre, deve essere mantenuta una buona produttività indipendentemente dai cambiamenti nei piani di produzione e dall'intrinseca variabilità che caratterizza il comportamento umano. Pertanto, il lavoro si sviluppa attorno a due obiettivi principali:

1. Monitorare e prevedere in tempo reale l'evoluzione dell'attività umana in corso. Ciò è utile per la pianificazione delle attività, poiché conoscere lo stato di avanzamento dell'attività corrente consente un migliore coordinamento tra gli agenti. Sono state sviluppate due strategie distinte, che considerano il fatto che le persone possono eseguire la stessa operazione in molti modi e a velocità diverse, facendo errori occasionali e brevi pause.

2. Pianificare dinamicamente le operazioni degli agenti collaborativi per ottimizzare la produttività. Nell'approccio proposto, un Digital Twin basato su Reti di Petri Temporizzate tiene traccia in tempo reale dello stato dell'area di lavoro condivisa sulla base dei dati provenienti dai controllori dei robot e dall'unità di monitoraggio degli operatori. Il Digital Twin viene quindi utilizzato per simulare l'evoluzione futura del sistema e determinare le istruzioni ottimali per uomini e robot seguendo un approccio *receding horizon*. Questo consente di adattare dinamicamente il piano alla variabilità del sistema e al verificarsi di guasti del robot.

In aggiunta, è stata studiata la convenienza di introdurre robot collaborativi nel processo di kitting. Il kitting è un'operazione chiave nella logistica per la produzione flessibile e consiste nel raggruppare insieme articoli separati da fornire come un'unica unità alla catena di montaggio. Oggigiorno, viene solitamente eseguita manualmente da operatori umani, ma i robot possono aiutare a ridurre lo sforzo sostenuto dall'operatore e aumentare la produttività. Perciò, in questa tesi viene presentato anche un algoritmo di pianificazione online per guidare le operazioni di picking dell'uomo e del robot.

Una considerazione che sta alla base dell'intero lavoro è che la produzione flessibile è caratterizzata da frequenti cambiamenti di produzione, che si riflettono in cambiamenti nel layout dell'area di lavoro e nelle operazioni da eseguire. Quindi, requisiti per il framework sviluppato in questa tesi sono stati la facilità di allestimento e modifica, e le elevate capacità di apprendimento. Questo è particolarmente vero quando si considera il comportamento umano, che è difficile da definire completamente a priori e cambia anche nel tempo, per esempio a causa di una maggiore esperienza o affaticamento.

Contents

CHAPTER 1

Introduction

SINCE decades, industrial robotics has been one of the key technologies that fostered productivity and cost reduction in manufacturing, largely contributing to the rise of mass production. Starting from the automotive industry, robots have been progressively adopted for various applications, such as welding, painting, packaging, palletizing, assembly, and machine tending. In general, robots have been designed to replace human operators in performing repetitive, dangerous, or laborious activities. However, due to safety concerns, their adoption has been subjected to a strict separation between humans and robots, with the latter placed in isolated cells protected by fences [42, 65].

The present-day global economy is rapidly transitioning from mass production to mass customization, which is characterised by high-mix low-volume production and frequent changes of products and processes. For this reason, companies are required to meet a high level of flexibility to stay competitive. In recent years, the novel paradigm of Industry 4.0 is gaining momentum to respond to the current economic situation [46]. Industry 4.0 fosters the transformation toward the so-called *smart factory*, where interconnected cyber-physical systems cooperate to optimise performance in

real-time [47]. The rigidity of the long-established paradigm for industrial automation and robotics does not fit this new scenario [11]. Instead, advances in human-robot collaboration are playing a key role in providing companies with an adaptable and powerful tool capable of enhancing efficiency and flexibility of manufacturing processes [24, 53, 76, 159].

Collaborative robotics removes the physical separation and allows robots and humans to share the same environment and work closely with each other [5]. On the one hand, the reduction in cost and space consumption favours the spread of robots within Small and Medium-sized Enterprises (SMEs). In fact, despite the constant growth in the number of robot installations [66], most SMEs do not have the budget and the expertise to buy, install, and program fully robotised manufacturing lines. On the other hand, collaborative robotics is particularly interesting for those companies and manufacturing processes where manual work is still prevalent. Whereas robots can relieve workers from fatiguing and alienating tasks, humans can leverage their superior cognitive and manipulative skills to perform operations that are difficult to automate or that require high-level decision-making. Moreover, the presence of the human reduces the need for a structured environment and allows for greater flexibility in the face of changes in the production. For this reason, several works in the literature concentrate on defining systematic and quantitative methods to identify the operations of a manufacturing process that could benefit the most from the introduction of human-robot collaboration [50, 156, 171]. Moreover, thanks to the reduction in cost and the greater adaptability and ease of use that human-robot collaboration entails, non-industrial applications are also developing, such as medical and household robotics.

1.1 Background works

Several challenges arise when dealing with human-robot interaction, that must be solved to achieve effective collaboration. Primarily, the safe coexistence of humans and robots in the shared workspace must be ensured. As such, safety issues have been the main research focus in the past years and have been addressed from different perspectives [135]. Standard industrial robots are designed to work in protected environments and cannot work alongside human workers without the risk of causing severe injuries due to high speed and forces [52]. Therefore, as already mentioned, a new generation of robots called *collaborative robots* (or *cobots*) has been designed to be inherently safe. From the point of view of the appearance, this means lighter structures, the elimination of dangerous edges, and the adoption of

soft covers to limit damages in case of collisions. For the same reason, the maximum speed and force that the robot can exhibit are limited by design [43]. Novel engineering solutions have been specifically developed for the design of cobots. For instance, a new actuation concept that reduces the impact loads is developed in [172], while [101] proposes a framework for safer robot design based on injury analysis by merging injury biomechanics data and robot collision behaviour.

In addition to hardware solutions, safety-oriented functionalities are embedded in the form of advanced control algorithms that comprise collision avoidance, collision detection, and human motion tracking. Collision avoidance strategies try to prevent impacts between humans and robots altogether. Current safety standards [42,43] define the so-called *speed and separation monitoring* principle, i.e. safety is ensured by keeping a minimum distance between the manipulator and the worker's body, which depends on the robot velocity and payload [168]. If the operator moves too close to the cobot, the latter slows down until a complete stop. Safety-oriented control architectures are presented in [72], which introduces the artificial potential field method, in [12, 82, 126], based on the definition of a *Danger Field*, and in [52], which leverages injury knowledge based on medical observation. As for real-time collision detection, examples can be found in [51], which compares different collision detection and reaction strategies to minimise the negative effects following an impact with the human, [53], which exploits a vision sensor, and [45, 141], which instead address the problem without requiring dedicated sensors.

More advanced strategies can be conceived using a sensing system that can detect the presence of human beings and track their movements in the space near the robot. [33, 125, 136] propose approaches based on vision sensors, while [108] detects the worker with the help of high-visibility clothing. Depth data are also exploited in [41, 109, 127] to better quantify distances and occupancy volumes. Instead, [110] proposes the adoption of pressure-sensitive sensors embedded in the floor to track human motion. With this information, the robot can operate in the nominal condition when there is no danger for the workers and take appropriate actions otherwise, such as reducing velocity [80, 168] or modifying the trajectory online with a reactive [41,107,127] or proactive [17,83] approach. One can notice that the application of the aforementioned strategies results in non-deterministic durations of robot trajectories. In [118], the authors provide an estimate considering a probabilistic description of the space occupied by the humans.

At present, the maturity of the state of the art on safe human-robot inter-

action makes it possible for the agents to coexist in a shared workspace without the risk of serious danger. Also, collaborative robots have become faster, more accurate, and reliable. As a matter of fact, many robot manufacturers have designed cobots that are being sold to companies all around the world. As such, attention has turned from allowing mere coexistence to enhancing real collaboration among the agents working in the shared workspace. The strategic importance of this field of research is confirmed by the funding of numerous European projects on the topic, most of which are still in progress. For instance, the *CoLLaboratE* project [1] aims to develop a comprehensive framework for human-robot collaboration for assembly operations. Robots are equipped with adaptable skills and can learn collaborative tasks from human demonstrations. In the *CoLRobot* project [2], the robot acts as a *third hand* by delivering parts and tools to the human worker. The operator, for his/her part, communicates with the cobots using gestures and tactile commands. Instead, the *SHAREWORK* project [3] intends to provide companies with a modular system capable of perceiving the environment and understanding human actions through smart sensors, augmented reality, and gesture and speech recognition in order to ensure more effective cooperation. Finally, the *THOMAS* project [4] wants to develop a mobile dual-arm cobot endowed with high cognitive capabilities for reconfigurable manufacturing systems.

A prerequisite that enables fluent and effective human-robot collaboration is the understanding of human behaviour. To do so, exteroceptive sensors, artificial intelligence, and machine learning are becoming pervasive in robotics. Several probabilistic models have been exploited to predict human intentions: Gaussian processes [91, 164], conditional random fields [74], hidden Markov models [87], and others [6, 95, 132]. Given knowledge on the current human intention, the robot can better assist the human in accomplishing the collaborative task. For instance, [79, 143] deal with the comfortable and reliable handover of parts, while [67, 119, 139] address human-robot co-manipulation of bulky objects. Specifically, [139] focuses on learning collaborative strategies from demonstrations whereas [67, 119] concentrate more on ergonomics aspects. The pros and cons of anticipatory robot response, which can benefit performance but also be perceived negatively by the operator, have been investigated in [7, 64].

Nowadays, assembly operations are one of the most typical applications for human-robot collaboration. Monitoring the human allows for better coordination of the agents, which in turn fosters correct task allocation and scheduling. Usually, a one-to-one scenario where a single human cooperates with a single robot is considered. In [58, 81] the robot predicts when

and which specific assistive actions, e.g. providing parts and tools, are required based on the sequence of operations performed by the human. To infer the worker's behaviour, the authors exploit hidden Markov models and dynamic Bayesian networks, respectively. Several planning strategies to determine the best sequence of actions for humans and robots can be found in the literature. For instance, centralised methods are presented in [21], based on genetic algorithms, [68], which leverages A* search on AND/OR graphs, and [49], which formalises the problem as a Mixed Integer Linear Programming optimization. Instead, [29, 93] propose decentralised control schemes.

Although most works aim to maximise productivity, some of them address the problem from perspectives that are peculiar to human-robot interaction. In [55, 128], the authors investigate the effects of the human's trust in the robot both in terms of efficiency and the worker's well-being. Instead, [106] studied how the operator's role during collaboration (that can be either leader or follower) influences his/her psychophysiological response and production rate. In [165], the authors propose to switch among predetermined assembly sequences based on human preferences. [20] exploits multimodal robot communication through speech, gaze, gestures, and manipulation to increase the fluency of collaboration. Also, [13] compares the effects of different multimodal feedback on user performance. As a matter of fact, providing the human with information to better understand the robot actions can be as important as the robot's understanding of human behaviour [32, 120]. Methods to improve human situation awareness in HRC through visual, auditory, and tactile feedback have been proposed in the literature [18,159]. Moreover, recent technologies such as augmented reality have been exploited for this purpose [97].

A more in-depth discussion of the state of the art for relevant topics is presented in the introductions of the chapters of the thesis, whereas the main contributions and the outline of the thesis is reported in the following Section.

1.2 Thesis contributions

The present thesis provides methods and tools to control the production of a flexible manufacturing cell, where multiple humans and robots cooperate to assemble different products. Flexible manufacturing is characterised by time-varying mix and frequent production changes, which reflect on modifications in the workspace layout and the operations to be performed. In this scenario, it is crucial to reduce set-up times as much as possible, favouring

plug-and-play and easily reconfigurable approaches. Also, learning capabilities are key assets to adapt to the variability of the process. This is especially true when human behaviour is considered, which is difficult to fully define a priori and also changes over time, e.g. due to training and fatigue. On the other hand, an estimate of the expected duration of the ongoing activities is beneficial for task scheduling, as it allows for better coordination among the agents. Versatile scheduling algorithms are needed to organise the complex work-flow and fully exploit the available flexibility, ensuring the optimal use of resources and the smart management of unexpected events. Also, good productivity must be maintained regardless of changes in production plans and the intrinsic variability of human behaviour.

In this view, this thesis provides the following contributions:

1. It presents two distinct strategies to model, monitor, and predict the advancement of the ongoing human activity in real time. Both methods consider the fact that humans can perform the same operation in many ways and with different speeds, occasional errors, and short pauses.

2. It introduces a formalism to automatically define and update a Digital Twin of a multi-product collaborative assembly process, which also models robot faults and human errors. The Digital Twin is the basis for the complete control architecture that is responsible for task allocation and scheduling.

3. It proposes a dynamic scheduling algorithm to plan the operations of the collaborating agents so to optimise productivity while adapting the schedule to the human variability and the occurrence of robot faults.

4. It describes a novel visuo-haptic user interface to give instructions to human operators.

5. It investigates the convenience of introducing human-robot collaboration to the kitting process to improve ergonomics and productivity. In particular, a way to associate an ergonomic score to each picking action and an online scheduling algorithm to guide the operations of the human and the robot are provided.

All the developed algorithms have been tested on realistic industrial scenarios to assess their performance and limitations.

The dissertation is organised as follows:

Part I discusses the problem of modelling and monitoring the current human activity. Attention focuses on assembly operations, although the proposed frameworks can generalise to a wider scope of application. Human tasks are seen as complex activities that can be accomplished following several sequences of low-level actions, which compose different variants of the same task. Also, the possibility of execution errors and small pauses is considered.

Chapter 2 describes a strategy to model and monitor the human activity learning from demonstrations. The structure of the task, with all its variants, is discovered in the training phase relying on the automatic segmentation of human motion trajectories. Then, motion segments are used to train a classifier to identify and parse the variant being performed by the operator at run-time. This information has been used to predict the most likely future evolution of the human activity in order to better plan the assistive operations of a robot during collaborative assembly.

Chapter 3 focuses on the real-time monitoring of human activity with the primary objective of estimating its expected duration. The proposed method is based on a modified version of the Dynamic Time Warping algorithm and does not require any training phase. Instead, it learns online from previous repetitions of the same activity and automatically recognises previously unseen variants, which are added to the activity model. The prediction performance of the algorithm is discussed with reference to an industrial assembly case and considers also the presence of peculiar variants of the task, such as those associated with errors.

Part II starts by defining the scheduling problem for flexible collaborative cells and presenting the overall control architecture. Then, each of the main system components is detailed. The proposed framework considers multiple humans and robots working together for the assembly of several products according to a time-varying mix. Particular attention is given to provide the system with strong adaptation and learning capabilities, as required by the frequent changes that characterise flexible manufacturing.

Chapter 4 describes the proposed control architecture. A Digital Twin of the collaborative cell tracks the state of the assembly process in real-time based on data coming from the robot controllers and the human monitoring unit. Then, the current state of the Digital Twin is taken

as the initial condition by the scheduling algorithm, which exploits a receding horizon approach to adapt the schedule to the system variability and the occurrence of robot faults. The Chapter continues by detailing how to build the Digital Twin of the flexible manufacturing cell, which models both the physical structure of the workspace and the assembly tasks, including situations that originate from human and robot failures. The Digital Twin is based on Petri Nets and is automatically generated from the definition of the assembly products via Augmented AND/OR Graphs. A procedure to easily update the Digital Twin in the face of changes is also provided. Moreover, a way to integrate the proposed architecture inside a commercial Product Lifecycle Management software is presented in Appendix A, which provides a virtual simulation environment for automatic manufacturing process verification and commissioning.

Chapter 5 presents the scheduling algorithm used to determine the optimal instructions for the humans and the robots working in the cell. Feasible future evolutions of the production are simulated starting from the current state of the process, as given by the Digital Twin. To do so a temporal description is added to the Petri Net model and its reachability tree is explored. The best task plan is selected according to a cost function that favours productivity and tracking of the target mix. Furthermore, the chapter discusses a novel visuo-haptic interface to give instructions to human operators and proves that it is a viable and effective solution for complex human-robot collaboration scenarios.

Chapter 6 applies the human-robot collaboration concept to the kitting process, which is a key logistic task in flexible manufacturing. Pure manual kitting, i.e. the current industrial standard, may induce the development of work-related musculoskeletal disorders. Therefore, a receding horizon dynamic scheduler is proposed with the aim of enhancing both ergonomics and productivity. For this purpose, a method to associate an ergonomic measure to each picking action is also presented. The scheduler is formalised as a Mixed-Integer Linear Programming optimization and ensures the coordination of the agents to prevent collisions.

Chapter 7 briefly reviews the contributions and the limitations of the thesis, while also suggesting directions for further developments.

The findings and results contained in this thesis are based on the following publications:

- R. Maderna, P. Lanfredini, A. M. Zanchettin and P. Rocco, "Real-time monitoring of human task advancement," *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Macau, China, 2019, pp. 433-440.
- R. Maderna, M. Poggiali, A.M. Zanchettin and P. Rocco, "An online scheduling algorithm for human-robot collaborative kitting," *2020 IEEE International Conference on Robotics and Automation (ICRA)*, Paris, France, 2020.
- R. Maderna, M. Ciliberto, A.M. Zanchettin and P. Rocco, "Robust real-time monitoring of human task advancement for collaborative robotics applications," *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, NV, 2020.

and on the following submitted material:

- R. Maderna, M. Pozzi, A. M. Zanchettin, P. Rocco and D. Prattichizzo, "Flexible Scheduling and Tactile Communication for Human-Robot Collaboration," submitted to journal, July 2020.

Finally, the following publications contain relevant results in the field of industrial robotics, which are not covered in the present document:

- R. Maderna, A. Casalino, A. M. Zanchettin and P. Rocco, "Robotic Handling of Liquids with Spilling Avoidance: A Constraint-Based Control Approach," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, QLD, 2018, pp. 7414-7420.
- R. Maderna, A. Gatti, D. Nicolis, A. M. Zanchettin and P. Rocco, "Telerobotic handling of liquids in open containers: an optimization-based bilateral controller for spilling avoidance," submitted to journal, July 2020.

Part I

Real-time monitoring of human activity

CHAPTER 2

Modelling and parsing of human activity based on demonstrations

In collaborative robotics applications, human behaviour is a major source of uncertainty. Therefore, monitoring and predicting the evolution of the current human activity is beneficial to the effectiveness of task planning, as it enables a higher level of coordination between robots and humans. In particular, robots must adapt to the operator, who can perform each task in many ways and with different speeds, occasional errors, and short pauses. Moreover, training, distraction, and fatigue influence the workers' actions.

The problem can be addressed at different levels, such as the classification of the current human task, the prediction of the future sequence of operation performed by the operator, and the monitoring of the ongoing activity. In [57], the author proposes a complete framework for human activity recognition to provide a human-machine interface for the integration of human workers in intelligent manufacturing systems. Instead, Hidden Markov Models were trained in [28, 71] to infer the current human activity. In [123] and [170], task classification is achieved by early prediction of the target of reaching motions. [123] relies on an offline training phase to construct motion libraries that are then used for Bayesian classification in

real-time. Differently, Zanchettin and Rocco [170] exploited a model-based approach to generate trajectories associated with each available target location. Each new measurement is compared to the model-based prediction to update the probability associated with each goal. Notably, the method does not require a training data set and allows the human to change target during motion. Differently, Malaise et al. [98] exploit a full-body suit and gloves equipped with inertial and force sensors to recognise pick and place tasks in industrial settings. The use of wearable sensors allows continuous monitoring in complex environments where the human may not be visible to a camera at all times. Finally, [15] and [153] deal with the recognition of human body postures. The former detects the 2D poses of multiple people from video streams using Convolutional Neural Networks. The latter uses RGB-D data to identify body postures through joint angles. The problem of predicting the most likely future task of the human is addressed, for instance, in [167], where the human activity pattern is modelled with higher-order Markov Chains. Previously, Li et al. [89] proposed a framework for activity prediction based on causality relations and semantics. In [144], the authors used gaze information to monitor the worker's activity and interpret his/her future intention. This allows a robot to perform the best cooperative action to help the human.

Monitoring the evolution of the current human activity allows correctly allocating tasks and reacting to unexpected changes [19, 134, 165]. Most of the aforementioned research works deal with the recognition of whole-body postures or reaching motions. However, they are not sufficient to monitor the activity of human workers during manufacturing operations, and assembly in particular. This is a tough challenge, since the human is neither fully controllable nor repeatable: even when the operator is instructed on the task to perform, it is impossible to control its execution. At each repetition, he/she will complete the same activity with different speeds and movements. Moreover, the ideal strategy should be non-intrusive and low cost to favour the workers' acceptance and favour its use by companies. In this Chapter, we propose a framework for modelling and parsing human activity based on demonstrations. The structure of the task, with all its variants, is learnt in the training phase relying on the automatic segmentation of human motion trajectories, recorded by an RGB-D camera. In real-time, a classifier trained on task segments identifies and parses the variant performed by the operator. Early identification of low-level actions enhances human-robot collaboration in two ways. First, the worker is notified in case of errors, so that he/she can immediately implement corrective actions. Second, prompter prediction of the future evolution of the task leads to better

planning of the robotic actions.

In the following, Section 2.1 outlines the proposed approach. Section 2.2 describes how to learn the model of the human operation, while the proposed classifier of human actions is detailed in Section 2.3. Then, the classification outcome is exploited at run time as described in Section 2.4. Finally, Section 2.5 presents experimental results obtained with a realistic assembly task.

2.1 Solution concept

In general, assembly operations consist in joining parts to form a single final product, possibly with the help of tools. In that situation, reaching motions to pick parts from feeders alternate with the actual assembly actions performed in a convenient position in front of the operator. The former are usually characterised by broad motions of one arm, whereas the latter see the two hands strongly interacting with each other and with the product. In the perspective of monitoring human activity, two main factors must be considered. First, the operator can execute the same operation following a different sequence of low-level actions. Second, the same component can be used multiple times at different points of the assembly. Therefore, the simple knowledge of the operator reaching a buffer and picking a specific part does not provide sufficient information to discriminate among the different variants of the operation. In other words, describing and monitoring the human activity only in terms of reaching motions, as in [123, 170], is limiting. Instead, a richer model must be introduced that also considers the actions performed between two consecutive reaching motions. In this view, a variant of the task is defined as an ordered sequence of human actions, which comprise both reaching motions and assembly actions.

Figure 2.1 depicts the pipeline of the proposed approach, which develops in an initial training phase and an online phase for real-time monitoring. During the offline training, an expert user builds a database of motion trajectories by demonstrating several times all the feasible variants of the operation. Then, each trajectory is automatically segmented to identify reaching and assembly actions. This is done with a hybrid method that combines velocity and position data of the operator's hand movements. From the result of trajectory segmentation, one can build a complete model of human activity that comprises all known task variants, which constitutes the basis to parse and predict the evolution of operations at run-time. On the other hand, the segmented dataset is used to define a classifier able to recognise online the action performed by the operator. The proposed classifier is based on

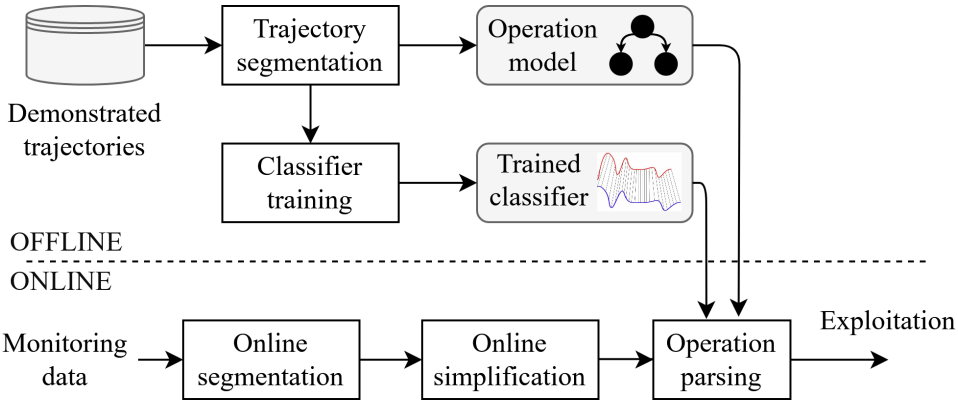


Figure 2.1: *Pipeline of the proposed monitoring algorithm.*

Dynamic Time Warping (DTW), which is an algorithm that measures the similarity between two time series and is better detailed in Section 2.3.

At run-time, an RGB-D camera provides real-time data on the operator's hands motion. The human trajectories are segmented online into the low-level actions identified during the training phase and undergo a preprocessing step to ease classification. The result of both the segmentation and classification stages provides information to parse the model of the operation in order to identify the variant being performed and predict the future evolution of the task. Finally, the output of the algorithm is exploited to notify errors to the worker and plan assistive operations from the robot.

2.1.1 Tracking of human motion

Real-time information from the workspace is required to monitor the evolution of the ongoing activity: this can be done by tracking the human operator's movements. The tracked features must be selected to obtain measurements that are informative and robust within the range of possible activities. Considering the case of industrial assembly tasks, it is meaningful to focus on the motion of the operator's hands, as the rest of the body remains largely stationary. Several methods for human hand tracking have been proposed in the literature. However, most of them underperform when the two hands are interacting [102, 124, 163], as it happens during assembly activities, or require an excessive amount of computing time [114], thus not being suitable for real-time tracking.

From preliminary experimental tests on diverse human actions, it has been decided to track the Cartesian positions of wrists and index fingers of each operator's arm. Fingers provide more informative movements but are

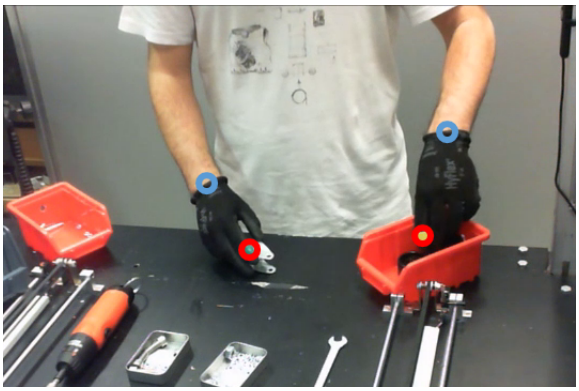


Figure 2.2: *Operator's tracked features: wrists (blue) and index fingers (red).*

prone to occlusions, whereas wrists give more robust data. The evolution of the human task is thus described by a 12-dimension time series built from the Cartesian coordinates of the four features. In the implementation, an RGB-D camera (Microsoft Kinect) was used. Wrist data are directly provided by the Kinect tracking software, while the position of the fingers is extracted from images with the help of coloured markers placed on the worker's gloves. Figure 2.2 depicts an example of the tracked features.

The algorithm used to extract finger data is outlined in Figure 2.3. Starting from the RGB image provided by the vision sensor, the same pipeline is followed for each one of the two hands in parallel. First, the known position of the wrist in the depth space is mapped on the colour image in order to identify a region of interest that contains the hand. In this way, we limit the size of the image to speed up further processing. Then, the cropped image is transformed from the RGB to the HSV colour space. A colour threshold filter is applied to the obtained image in order to identify the marker placed on the index finger. The HSV space description is preferred, as it is more robust against glares and changes in lighting conditions than the RGB model. Finally, the centre of the marker is taken as the finger position in the colour image, which is then mapped back to the depth space to obtain the 3D Cartesian coordinates.

2.2 Learn the model of human task

This Section details the first steps of the proposed algorithm that compose the first row of Figure 2.1. The starting point is the creation of the dataset of demonstrated trajectories. According to Section 2.1.1, each trajectory is a multivariate time series that describes the motion of the operator's hands

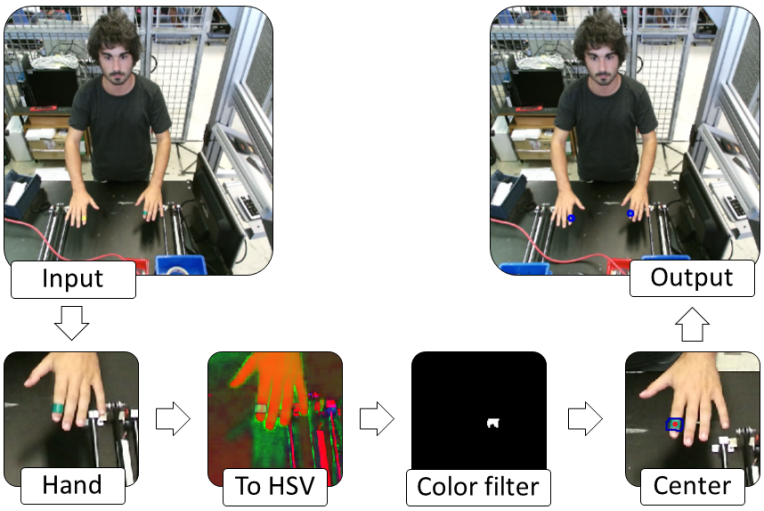


Figure 2.3: *Pipeline of the finger coordinate extraction algorithm.*

throughout the execution of the task. An expert user performs multiple times the operation to be monitored, showing all possible variants, i.e. all the suitable sequences of reaching motions and assembly actions that allow the worker to correctly complete the activity. Once that the dataset is available, it is possible to automatically segment the complete executions into reaching motions and assembly actions, and use the obtained information to build a model of the operation with all its variants.

2.2.1 Automatic segmentation of human activity

To identify the different variants of the task, one must be able to recognise the sequence of human actions that composes the complete executions of the operation contained in the dataset of demonstrated trajectories. This is possible with the help of an automatic segmentation algorithm that automatically detects the start and end point of each human action without manual intervention. The proposed approach relies only on the data acquired by the vision sensor, without any prior knowledge of the task or the workspace layout. Specifically, a hybrid velocity/position-based method is conceived, where velocity data are exploited to find segmenting points without requiring information on the location of parts in the environment.

While performing the task, the operator alternates reaching motions, i.e. he/she moves the hands towards part feeders to pick up items, and assembly actions, i.e. he/she uses the picked items to proceed in completing the product. The characteristics of the two classes of human actions are very

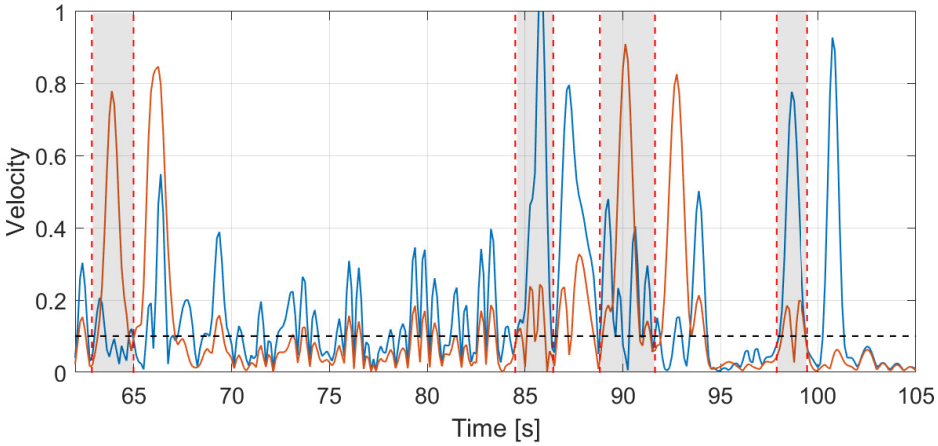


Figure 2.4: *Example of segmentation based on hybrid velocity/position data. The plot reports the velocity of the right (blue) and left (red) hand, and the velocity threshold (dashed black). Segments are identified as reaching motions (grey background) or assembly actions (white background).*

different and the automatic segmentation algorithm must be able to deal with both. First of all, skeletal points provided by the Kinect sensor locate the human in the environment and allow us to define the assembly station as a rectangular area on the worktable in front of the operator. When the operator performs a reaching motion, one hand starts moving toward the goal with increasing velocity, then it slows down and stops while grasping the object. As a consequence, one can identify the start of reaching motions as the point when one of the hands exits the assembly station and the end point as the one where the velocity goes below a predefined small threshold. An important note is that the operator can use either of the two hands to perform reaching motions. Therefore, the motion of both hands is considered to identify all segmenting points. The return motion from the part feeder back to the assembly area is considered as being part of the assembly action, which is defined as the sections of the operation between two subsequent reaching motions. Assembly actions can involve complex operations, such as insertion or screwing, so that the hands exhibit different motion patterns than those characterising reaching motions. In particular, velocities might exceed and returns below the threshold multiple times while positions remain contained in the assembly station. An example is shown in the second segment of Figure 2.4.

In order to build the operation model and the dataset to train the classifier, all segments must be labelled. As far as reaching motions are con-

cerned, labelling is automatic and follows directly from the segmentation algorithm. In fact, end points are associated with the goal positions of reaching motions, such as part feeders or tool storage. Therefore, reaching motions toward the same goal are grouped by means of simple K-means clustering with respect to the position of the segmenting points expressed in the Cartesian space. Furthermore, the centroid of the segmenting end-points associated with motions of the same class defines the goal location.

On the other hand, assembly actions are manually labelled by an expert user (possibly the one who demonstrated the operations). This step is needed to account for the fact that different assembly actions might follow a specific reaching motion. This happens, for instance, when the same component is used in several points of the assembly sequence, as it is considered in the experimental use-case presented in Section 2.5. Discriminating how the component is being assembled allows the monitoring algorithm to identify the correct variant of the task that the operator is following. Figure 2.4 reports an example of segmentation into actions for a section of an operation, where segments are labelled as being reaching motions or assembly actions.

2.2.2 Generation of the task model

After the segmentation step, each demonstrated trajectory is defined as an ordered sequence of labelled segments, i.e. of reaching motions and assembly actions. The set V of known task variants contains all the unique sequences of labels. Variants differ from each other after a possible common initial part, defined by the same sequence of labels. Consequently, a single model of the human activity that describes the possible ways to perform the operation can be represented as a tree $T = (N, A)$, where N is the set of nodes and A the set of arcs. Nodes are associated with labels, i.e. actions, and branches mark the presence of variants at that point of the execution. Algorithm 1 outlines the recursive procedure used to define the tree structure. In particular, the set of variants is partitioned to obtain subsets composed of variants having the first label in common. Then, a branch for each partition is created, whose structure depends on the further partition of the associated subsets, from which the initial common label has been removed. For instance, the set $V = \{\{l_1, l_2, l_3\}, \{l_1, l_3, l_2\}, \{l_2, l_1, l_3\}\}$ is partitioned at the first step as $V_1 = \{\{l_1, l_2, l_3\}, \{l_1, l_3, l_2\}\}$ and $V_2 = \{\{l_2, l_1, l_3\}\}$. Thus, two branches depart from the root node, one related to l_1 and the other to l_2. Then, children of the l_1 node stem from the partition of $\widetilde{V}_1 = \{\{l_2, l_3\}, \{l_3, l_2\}\}$, where label l_1 has been removed. Therefore, one branch

Algorithm 1 Recursive algorithm to generate the task model.

```
procedure GENERATEMODEL(V)
    if V ≠ ∅ then
        T ← ADDBRANCH(root, V)                    ▷ Start tree construction
    end if
end procedure

function ADDBRANCH(l, V)                          ▷ Add branch to the tree
    Tree.N ← l                                    ▷ Add root node
    Tree.A ← ∅
    P ← PARTITION(V)                              ▷ Partition the set of variants
    for all V_i ∈ P do
        l ← v_i[0], v_i ∈ V_i                     ▷ Get first common label
        for all v_i ∈ V_i do
            v_i = v_i[1 : end]                    ▷ Remove common label
        end for
        B ← ADDBRANCH(l, V_i)                     ▷ Recursive step
        Tree.N ← Tree.N ∪ B.N                     ▷ Add new nodes
        Tree.A ← Tree.A ∪ B.A ∪ (Tree.root, B.root)   ▷ Add new arcs
    end for
    return Tree
end function
```

refers to $V_{11} = \{l_2, l_3\}$, the second to $V_{12} = \{l_3, l_2\}$. The final model of the human operation is depicted in Figure 2.5.

2.3 Dynamic Time Warping-based classifier of human actions

Providing the robot with a tool to monitor and understand the operator's behaviour is beneficial for enhancing human-robot collaboration. To this purpose, the real-time recognition of the action being performed by the human is crucial, as early classification enables prompter robot assistance. The necessity of a fast response poses a constraint on the best classification strategy. The DTW algorithm provides a method to compare an incomplete input, associated with the ongoing human action, to a complete reference execution. In the following, Section 2.3.1 presents the basis of the DTW algorithm. Then, Section 2.3.2 explains the use of DTW as a classifier for the application under study.

2.3.1 Dynamic Time Warping algorithm

DTW is a widely used algorithm that measures the similarity between two temporal sequences. Examples of applications include gesture recogni-

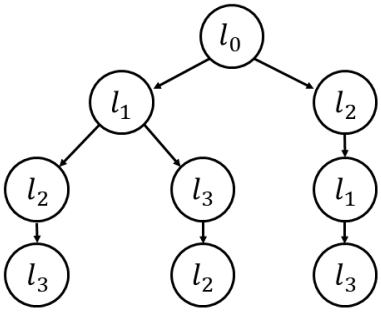


Figure 2.5: *Example of human task model with three variants.*

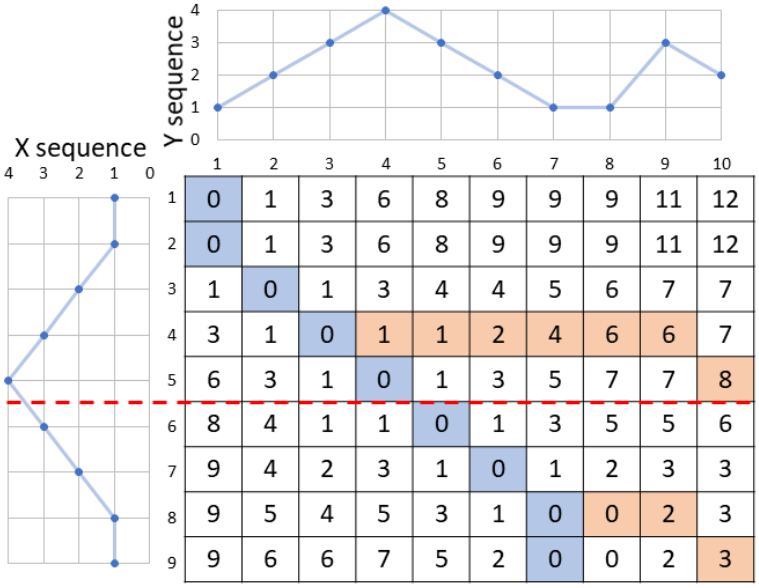
tion [57], data mining [129], image analysis [166], and fault diagnosis [75]. In its basic form, DTW calculates the optimal alignment between the points of two sequences and it is robust to nonlinear variations in the time dimension. This allows comparing two series regardless of the changes in their speeds. Modifications, called Open-ended DTW [157], already exist that allow for the comparison of incomplete input time series with complete references. The underlying idea of the DTW algorithm is to locally deform the time axis of the input sequence in order to associate each of its points with one point of the reference sequence in a way that minimises the cumulative distance between the aligned points. Let $X = (x_1, \dots x_{|X|})$ be the input sequence that describes the partial execution of the ongoing activity and $Y = (y_1, \dots y_{|Y|})$ the reference template sequence. The algorithm builds a $|X|$-by-$|Y|$ matrix where each element stores the cumulative distance $D(i, j)$ of the optimal warping for the subsequences $X^{(i)} = (x_1, \dots x_i)$ and $Y^{(j)} = (y_1, \dots y_j)$:

$$\begin{gathered} D(i,j) = \delta + \|x_i - y_j\| \\ \delta = \min\{D(i-1,j), D(i,j-1), D(i-1,j-1)\} \end{gathered} \tag{2.1}$$

where the following constraints are considered to handle limit cases:

$$\begin{cases} D(0,0) = 0 \\ D(i,0) = \infty \quad \forall i \\ D(0,j) = \infty \quad \forall j \end{cases}$$

The warping path $\phi(i)$ is a non-decreasing function that associates each point of X with the index of optimal truncation j_i^* for the reference sequence Y, so that $Y^{(j_i^*)}$ best matches the subsequence $X^{(i)}$. That is, for



	1	2	3	4	5	6	7	8	9	10
1	0	1	3	6	8	9	9	9	11	12
2	0	1	3	6	8	9	9	9	11	12
3	1	0	1	3	4	4	5	6	7	7
4	3	1	0	1	1	2	4	6	6	7
5	6	3	1	0	1	3	5	7	7	8
6	8	4	1	1	0	1	3	5	5	6
7	9	4	2	3	1	0	1	2	3	3
8	9	5	4	5	3	1	0	0	2	3
9	9	6	6	7	5	2	0	0	2	3

Figure 2.6: *DTW matrix for an incomplete input sequence X with the optimal open-ended warping path highlighted (blue). Warping paths obtained using global DTW when 5 and 9 input samples are available are indicated in red for comparison. Global DTW always considers the complete reference, leading to meaningless associations.*

$i = 1, \dots |X|$:

$$\phi(i) = \max\{j_i^*, \phi(i-1)\} \qquad j_i^* = \underset{j=1,\dots|Y|}{\arg\min}\, D(i,j)$$

Finally, the similarity measure between the input and reference sequences is given by $D(|X|, j^*_{|X|})$. Figure 2.6 shows an example of DTW matrix and compares the optimal warping paths obtained in case of incomplete input using global DTW, which align the partial input with the complete template, and the open-ended version exploited in this work. The latter is always able to find the best available partial match, while global DTW forces meaningless matches considering the complete reference.

Since the proposed algorithm aims to monitor the progress of the human activity, the inputs to the DTW are the time series describing the motion of the operator's tracked features, as described in Section 2.1.1. However, movements of the operator's hands are not bound to be synchronous. Thus, the relative temporal alignment among signals is not maintained for each repetition of the same activity. For instance, if the operator has to reach for two different objects, he/she can either grasp both objects simultaneously, in one sequence, or the reverse. This makes it difficult to describe the task

with a single time series, built with data coming from opposite sides of the body, and compare different executions with each other. The solution is to run two separate instances of the DTW algorithm, one that receives as input the motion of the right wrist and finger, the other that receives the trajectories associated with the features of the left hand. Then, the overall similarity value is the mean of the output of the two instances of DTW.

2.3.2 Classification of human actions

To classify human actions using DTW, the input to the algorithm is the time series composed of the samples belonging to the ongoing human action returned by the monitoring unit. Instead, the output of the automatic segmentation provides the training dataset for the classifier of human actions, which is defined as:

$$\mathcal{Y} = \{(Y_1, l_1), (Y_2, l_2), \ldots, (Y_N, l_N)\}$$

where Y_i are the time series that describes the trajectory segments and $l_i \in L$ the label associated with each human action. The simplest way to classify the current input is to compute its similarity with each and every available reference segment and compare the outputs returned by the DTWs. Then, the ongoing segment is classified as belonging to the same class of the reference with minimum DTW cost. However, this means running two instances per each trajectory in the dataset. Moreover, the DTW algorithm is computationally expensive and its complexity grows with the number of samples in the sequences as $O(|X||Y|)$. As a consequence, in the case of complex operations composed of a high number of segments, the computational time might increase up to a point that does not allow a real-time application. Also, the number of segments in the dataset depends on the number of demonstrated trajectories and contains several specimens belonging to the same class. A strategy to simplify the dataset of reference actions is introduced to lower the computational load by reducing:

1. The number of samples that compose the reference trajectories;
2. The number of reference sequences to a single trajectory per class.

In this way, the number of concurrent DTWs is limited to $2|L|$, i.e. twice the number of classes, and each DTW instance compares shorter sequences.

As for the first point, the goal is to represent each trajectory $Y_i = (y_{i,1}, y_{i,2}, \ldots, y_{i,|Y_i|})$ of the dataset as time series with a lower number of points $\widetilde{Y}_i = (\widetilde{y}_{i,1}, \widetilde{y}_{i,2}, \ldots, \widetilde{y}_{i,K})$, $K \ll |Y_i|$, which maintains the same shape of the original trajectory. Besides, simplification removes the high amount

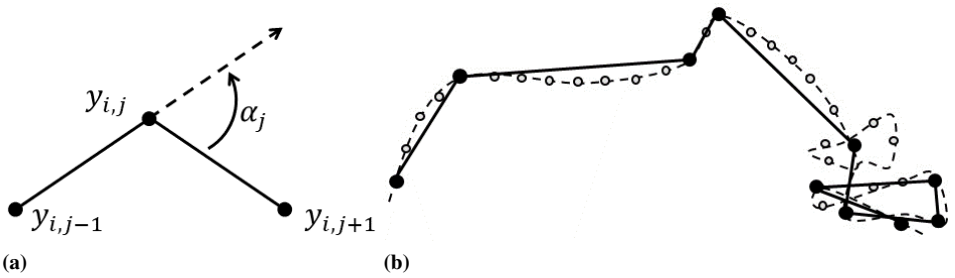


Figure 2.7: *(a) Graphical representation of the quantities used for the geometrical weight computation and (b) example of original trajectory (dashed) and simplified one (solid).*

of noise coming from the tracking system and the presence of spurious motion, especially at stop points, e.g. while grasping parts after a reaching motion. The proposed approach is based on the concept of the geometric weight presented in [88]. The weight w_j represents the influence of point $y_{i,j}$ on the overall shape of the trajectory and is defined as:

$$w_j = d(j) \cdot d(j-1) \cdot \alpha_j^3 \tag{2.2}$$

where $d(j) = \|y_{i,j+1} - y_{i,j}\|$ stands for the Euclidean distance between subsequent points and α_j is the local turning angle (see Figure 2.7a). That is, a point is given more importance if it is distant from other samples or describes a significant change in the local shape. Then, the simplified trajectory is obtained by iteratively removing the sample having the lowest weight until the K highest weighted samples are retained. Extreme points have infinite weight and cannot be removed. Algorithm 2 outlines the offline simplification routine, while Figure 2.7b shows an example of application. Note that the general shape of the original trajectory is preserved and the final noisy part is smoothed.

Algorithm 2 Offline simplification.

procedure SIMPLIFYOFFLINE(Y)
  for all $y_j \in Y$ **do**
    Compute w_j with (2.2)
  end for
  while $|Y| > K$ **do**
    $j^* = \arg\min_j w_j$
    $Y = Y \setminus \{y_{j^*}\}$
    Update w_{j^*-1}, w_{j^*} with (2.2)
  end while
end procedure

Algorithm 3 Online simplification.

procedure SIMPLIFYONLINE
  while true **do**
    $\widetilde{Y} \leftarrow$ GETNEWSAMPLE( )
    Compute $w_{|\widetilde{Y}|-1}$ with (2.2)
    if $w_{|\widetilde{Y}|-1} \geq$ threshold **then**
      $\widetilde{Y} \leftarrow$ REMOVE($y_{|\widetilde{Y}|-1}$)
    end if
  end while
end procedure

To address the second point, one can note that trajectories belonging to the same class have similar shapes. In order to obtain a single prototypical execution for each class of human action, it is sufficient to compute the average time series among all the simplified trajectories belonging to such class. The obtained trajectories are used as the reference sequences for the DTW algorithm. However, also the input sequence must undergo a simplification procedure to allow the comparison with the simplified reference. Since classification must be performed in real-time, an online version of Algorithm 2 is needed, which is reported in Algorithm 3. The main difference lies in the fact that it is not possible to know beforehand the length of the complete trajectory, nor the geometric weights associated with future samples. Therefore, every time a new point is available, the weight of the second to last sample can be computed. If the weight is below a given threshold, the second to last point is removed from the trajectory. The acceptance limit must be determined empirically so as to obtain similar compression as the offline simplification algorithm, which in the experiment was about 20%, meaning that only 20% of the original samples were retained. Although having input and reference sequences of different lengths is not a problem for DTW, input simplification help to reduce computational load and limit the influence of noise on the similarity measure.

2.4 Real-time parsing of human activity

This Section describes how the output of the offline training phase, that is the tree-like model of the human operation and the dataset of reference trajectories to be used by the DTW-based classifier, is used to monitor the activity of the human worker at run-time. The main steps of the online phase of the proposed algorithm are depicted in the third row of Figure 2.1.

The stream of monitoring data coming from the tracking algorithm of Section 2.1.1 is segmented online exploiting the positions of the assembly station and the goals of reaching motions identified during the offline segmentation phase (see Section 2.2.1). Specifically, a reaching motion starts when one of the hands of the operator exits the assembly station, as per the offline algorithm. Instead, the start of the subsequent assembly action is determined as the moment when the hand exits a region of space defined around the goal of the reaching motion. The position-based method is more robust for online use than the velocity-based one used in the training phase. Moreover, it allows checking the goal of the reaching motion, verifying the correct early classification obtained from the DTW, and rectifying possible errors made in the recognition of the completed human action.

The samples collected from the last segmentation point up to the current time instant compose the trajectory that describes the motion of the operator during the ongoing human action. This is taken as the input for the online simplification algorithm, whose output defines the input for the DTW. As already explained in Section 2.3.2, two instances of DTW per class are used to compare the ongoing trajectory with the reference segment execution. Then, early classification is attained searching for the class that returns the best similarity measure.

When the operator starts a new repetition of the operation, the monitoring of the ongoing activity starts from the root node of the task model. As soon as the first action is classified, the current node becomes the child of the root that is associated with the identified class. After that the segment has been completed, a new real-time classification is performed for the next one. This iterative procedure allows parsing the complete human activity from start to finish, identifying the correct variant being performed.

2.4.1 Exploitation

The state of the monitoring procedure can be exploited in several ways to improve human-robot collaboration and the individual performance of the worker. First, at each classification step, the current trajectory is always compared to all the available classes even if the current node of the task model has only one or few children. This allows the system to detect errors in the sequences of actions performed by the operator. In this case, we can notify the error to the human, who can immediately correct him or herself. Second, in the presence of a branch in the tree, i.e. of task variants, early recognition allows the robot to predict which will be the future actions performed by the human. Consequently, it can better decide which operation to execute and how. For instance, the robot can understand in which sequence it is better to provide different parts or equipment to the co-worker depending on the variant being performed. Or it can modify its motion to avoid and let free regions of the shared workspace that are going to be soon occupied by the operator. In other words, the more and prompter information the robot can extrapolate, the better the overall coordination between the agents becomes.

2.5 Experiments

The experimental campaign aimed to verify the performance of the proposed classification method and highlight the benefits that derive by employing the monitoring strategy to a human-robot collaborative task. The

Figure 2.8: *Final product.*

algorithm was implemented in C++ and run on an off-the-shelf laptop (Intel Core i7-8550U CPU 1.80GHz, 8 GB RAM). Human motion was recorder by a Kinect V2 RGB-D camera, whereas TCP/IP protocol was used to communicate with a UR5 robot. The test use-case consists in the partial assembly of the wheeled base shown in Figure 2.8, during which only two out of four caster wheels were mounted. The product is composed of several parts which are stored in appropriate feeders on the worktable. Namely, they are the swivel forks, the screws to fix the fork to the base, the screws and nuts to fix the wheel to the fork, and the wheels. Wheels for the left and right casters are supposed to be different and stored in separate buffers.

The considered task allows for several assembly sequences, however, we only focused on two of them that can highlight the added value of our approach over strategies based on reaching motions only, such as those presented in [123, 170]. In particular, the operator can first mount the left caster followed by the right one, or vice-versa. Instead, the sequence of actions to mount each wheel is the same: the operator picks up and mounts the fork on the base, takes the appropriate wheel, then fixes the wheel with screw and nut. During the execution of the operation, the boxes that contain the wheels are not directly accessible by the human. Instead, the robot must provide the right part depending on whether the operator mounted the right or left swivel fork. In this scenario, it is apparent that the monitoring of human activity and the early recognition of assembly actions are crucial to enable the correct and timely assistance from the robot.

In the following, Section 2.5.1 details the experimental training phase, while Section 2.5.2 discusses the obtained results.

Table 2.1: *Classes of human actions (RM = reaching motion, AA = assembly action).*

Label	Description	Type
l_1	Pick up big screw to fasten fork	RM
l_2	Pick up swivel fork	
l_3	Pick up left wheel	
l_4	Pick up right wheel	
l_5	Pick up small screw to fasten wheel	
l_6	Pick up small nut to fasten wheel	
l_7	Bring the big screw in the assembly station	AA
l_8	Mount the swivel fork onto the left hole using the big screw	
l_9	Mount the swivel fork onto the right hole using the big screw	
l_{10}	Position the left wheel between the mounted swivel fork	
l_{11}	Position the right wheel between the mounted swivel fork	
l_{12}	Insert the small screw into the wheel and the fork	
l_{13}	Use the small nut to fasten the wheel on the fork	

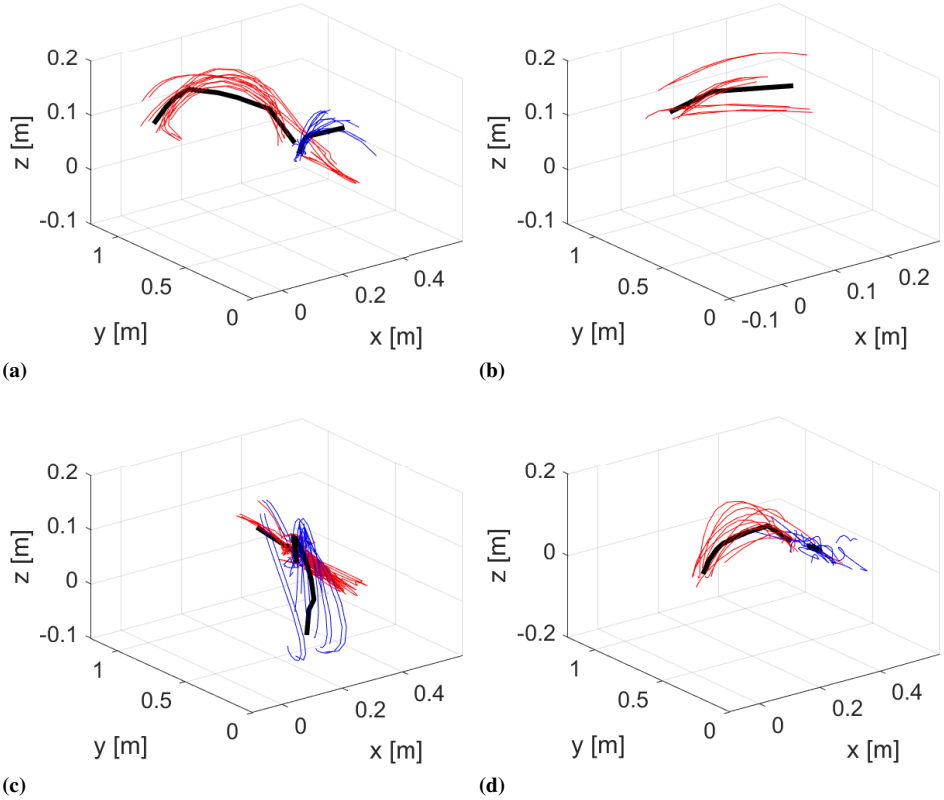


Figure 2.9: *Examples of classes of segment trajectories. Motion of the right hand (red), motion of the left hand (blue), and average class trajectory (black).*

2.5.1 Training phase

Initially, we demonstrated 10 complete executions of the assembly task, alternating between the two variants (right wheel first and left wheel first). The trajectories of the human hands have been recorded with the tracking algorithm presented in Section 2.1.1 and form the initial dataset. Following the proposed pipeline (Figure 2.1), the demonstrated trajectories are segmented to identify the human actions that compose the task and the goal locations of reaching motions. Overall, 6 classes of reaching motions have been identified to pick up as many parts, whereas 7 additional classes refer to assembly actions. Table 2.1 lists the human actions and associated labels. The obtained segments compose the training dataset and undergo the simplification process described in Section 2.3.2. Figure 2.9 reports some examples of raw segment trajectories together with the average class trajectory. Moreover, the model of the complete human task is built, which comprises the two demonstrated variants:

$$V_1 = (l_1, l_7, l_2, l_8, l_3, l_{10}, l_5, l_{12}, l_6, l_{13}, l_1, l_7, l_2, l_9, l_4, l_{11}, l_5, l_{12}, l_6, l_{13})$$
$$V_2 = (l_1, l_7, l_2, l_9, l_4, l_{11}, l_5, l_{12}, l_6, l_{13}, l_1, l_7, l_2, l_8, l_3, l_{10}, l_5, l_{12}, l_6, l_{13})$$

The first one corresponds to the case when the operator assembles first the left wheel and then the right one. Following V_2 he/she assembles the right caster, then the left one. Figure 2.10a illustrates the initial part of the obtained tree, which shows the presence of the branch and the alternation between reaching motions and assembly actions. For comparison, Figure 2.10b reports the equivalent model in case only reaching motions are monitored. On the one hand, considering fewer classes eases the classification task. However, it delays the recognition of the variant being performed, as demonstrated by the experimental results discussed in Section 2.5.2. In fact, the two variants start differentiating after that the operator picks up the first swivel fork, when he/she decides whether to mount it onto the left or the right hole (action l_8 or l_9). Instead, looking at reaching motions, one can tell the two variants apart only when the operator reaches either for the left or the right wheel (action l_3 or l_4).

2.5.2 Experimental results

To assess the performance of the algorithm, we monitored 6 repetitions of the assembly task in real-time, 3 per each variant of the operation, for a total of 120 actions. The trajectories recorded by the tracking system have been correctly segmented online by means of the position-based method outlined in Section 2.4. Performance of the classification stage have been analysed

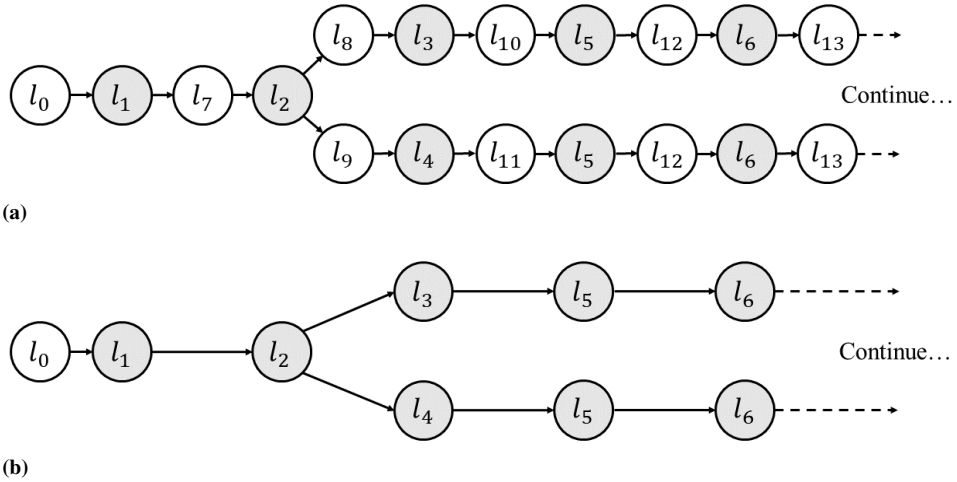


Figure 2.10: *Human task model for the proposed approach (a) and when only reaching motions are considered (b). Reaching motions are indicated with grey background, labels refer to Table 2.1. Including assembly actions anticipates the position of the branch and the recognition of the variant.*

in three different conditions, which differ in the amount of preprocessing undergone by the input and reference sequences that are given as input to the DTW-based classifier:

1. The first case compares the raw ongoing trajectory to the raw dataset of demonstrated segments. This means that no simplification step is performed to reduce the number of samples that compose the time series. Also, the number of concurrent DTW instances is equal to 400, i.e. twice the number of actions that form the training dataset.

2. The second case compares the simplified ongoing trajectory to the simplified dataset of demonstrated segments. The number of concurrent DTW instances is equal to 400, but each trajectory is composed of a fewer number of samples, with a compression ratio of 21.4%.

3. The third case compares the simplified ongoing trajectory to the average class trajectories. Therefore, only 26 instances of DTW run concurrently, two per each class.

The three conditions have been compared in terms of recognition rate, i.e. the percentage of human actions correctly classified, and in terms of computational time, which is crucial for online implementation. The latter cannot be considered as an absolute result since, in general, it depends on

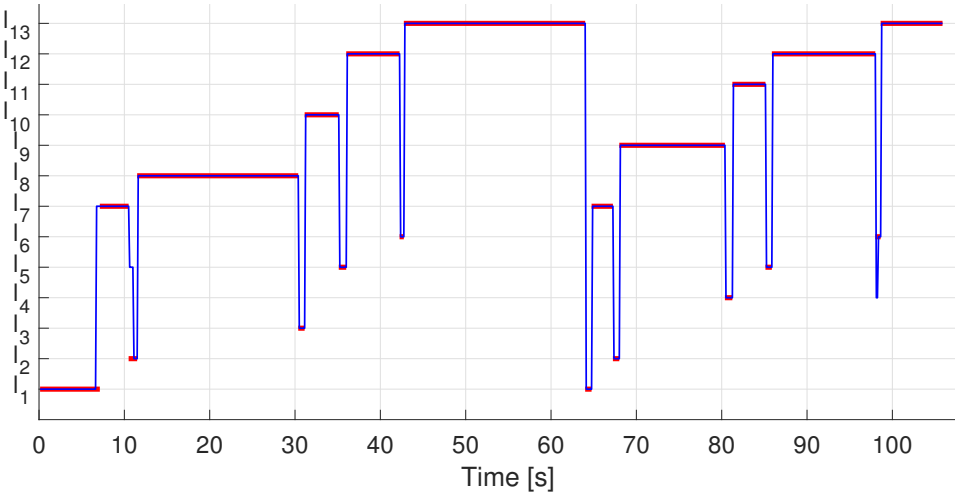
Table 2.2: *Performance indexes in the three conditions.*

Dataset	Recognition rate	Computational time/step	DTW instances	Compression ratio
Raw	98.75%	69 ms	400	100%
Simplified	98.75%	15 ms	400	21.4%
Mean	97.50%	1 ms	26	21.4%

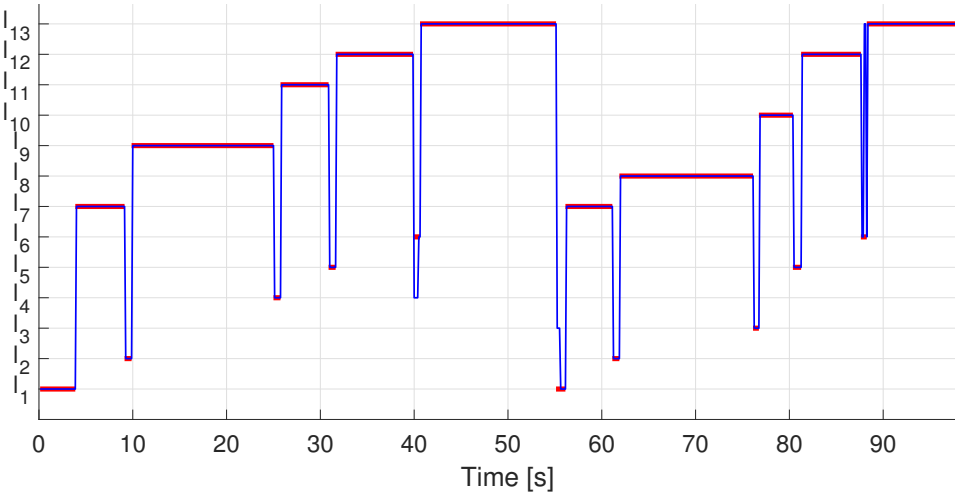
the hardware capability. Still, it shows the relative performance among the different cases. Table 2.2 reports the obtained results. Recognition rates are high in all conditions, with the classifier based on mean trajectories performing slightly worse at 97.5% of human actions correctly classified. Moreover, the computational load is consistent with expectations. The time required to process each new input sample is proportional to the concurrent DTW instances and the average length of the reference sequences. Thus, it is larger when the classifier works on the complete dataset of raw trajectories (69 ms per sample). When the simplified trajectories are considered, the computational time drops according to the attained compression ratio to 15 ms. Instead, about 1 ms is needed to update the classification output when the algorithm compares the input with only one average trajectory per class. Ensuring small processing time is crucial for real-time applications, especially when computational power is also needed to make decisions about how to exploit the information obtained from the monitoring system. Therefore, one can say that the third condition provides the best trade-off between performance and load.

Figure 2.11 shows two examples of real-time parsing of the human activity, one for each task variant. The red lines refer to the ground-truth segmentation of human actions, while the blue lines describe the output of the classifier based on simplified average trajectories. Form the plots, one can appreciate that the monitoring algorithm is able to correctly parse the activity with few local errors. Nevertheless, in the very few cases when the classifier is not able to correctly classify the ongoing reaching motion by itself, the response is rectified when the action is completed by assessing which goal station has been reached by the hand of the operator. This step is embedded in the online segmentation procedure.

A fundamental aspect of real-time performance is the delay with which the algorithm is able to correctly classify the ongoing human action. Indeed, early recognition is the key factor that can improve agents' coordination. From the experimental results, we found out that the recognition delay is independent of the duration of the human action. Instead, the algorithm



(a)



(b)

Figure 2.11: *Example of monitoring performance for the two variants of the task: left wheel first (a) and right wheel first (b). The blue line shows the real-time classification, the red one the ground truth. Segment labels refer to Table 2.1.*

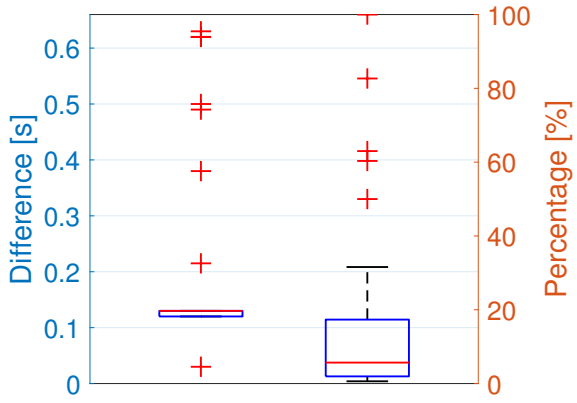


Figure 2.12: *Recognition delay in terms of elapsed time from the start of the action (left) and in terms of percentage with respect to the total action duration (right). Boxplots indicate median (red bars), quartiles (blue boxes), minimum/maximum values (whiskers), and outliers (red crosses).*

consistently provides the correct classification after about 0.13 s from the start of the segment. Exceptions to this are the sporadic cases when the classification is more difficult, for instance due to peculiar motion of the operator during the execution of the specific action, and the algorithm has to correct itself after an initial wrong response. The left boxplot in Figure 2.12 shows the recognition delay for the 120 performed classifications, where one can note the tight interquartile range. A constant recognition delay means that the correct classification happens at a different point of the action execution, which depends on the total segment duration. The right boxplot in Figure 2.12 reports the recognition delay as a percentage of the duration of the action. One can notice that most classifications are completed before 20% of progress of the ongoing action, thus supporting the claim on the early recognition capabilities of the proposed strategy.

The importance of early classification has been proved with a simple collaborative task, during which the operator performs the product assembly, while the robot must provide the boxes that contain the left or right wheels, which are not directly accessible by the human. The correct parsing of the activity and the early recognition of the variant being performed by the operator allows the robot to determine the best time and the correct part to supply to the operator. In this view, the proposed approach has been compared to the one from [170], which is based on reaching motions only. The expected results have been already outlined discussing the contents of

Figure 2.10, which shows the task models employed by the two strategies. Specifically, our approach anticipates the discrimination between the two variants by exploiting the difference in the assembly actions following the grasping of the swivel fork (l_8 and l_9). Instead, the baseline method recognises the variants based on the subsequent reaching motions (l_3 and l_4). Since the robot can start its assistive action only after determining which is the correct variant to provide the correct part, not considering the assembly actions in the monitoring of the human activity entails the presence of operator's idle time, who must wait for the robot after concluding the assembly of the swivel fork. Instead, using the proposed approach, the robot can start his task early and the human finds the required part ready to be picked.

We gathered experimental data from 4 executions of the assembly task for each of the two employed monitoring algorithms. Figure 2.13 shows the side-to-side comparison between screenshots taken from videos of the operator performing V_2 in the two cases. Initially, the two executions evolve in parallel, until the operator reaches for the swivel fork the first time (action l_2, Figure 2.13a and 2.13b). Then, the human starts assembling the fork onto the right hole. With the proposed approach, the robot promptly recognises the performed action and starts the assistive operation (Figure 2.13c). After the completion of l_9, the operator finds the correct box already in place and can start to assemble the wheel (Figure 2.13e). Instead, the robot remains still in the other case (Figure 2.13d) until the human finishes the assembly of the fork and tries to reach the wheel box (Figure 2.13f). Therefore, the operator has to wait for the completion of the robot operation (Figure 2.13g). Meanwhile, with the proposed monitoring the worker has almost finished to assemble the right caster (Figure 2.13h). On average, the operator's idle time was equal to 6.75 s when the strategy in [170] was applied. The value is consistent with the time required by the robot to pick and place the correct box in front of the operator.

(a) *Pick up swivel fork.*

(b) *Pick up swivel fork.*

(c) *Assemble swivel fork.*

(d) *Assemble swivel fork.*

(e) *Start right wheel assembly.*

(f) *Wait for robot assistance.*

(g) *Complete wheel assembly.*

(h) *Pick up wheel.*

Figure 2.13: *Screenshots taken from a video of the experiments. Comparison between the proposed approach (left column) and [170] (right column).*

CHAPTER 3

Progress–based human monitoring

In the previous Chapter, we introduced a first strategy to monitor human activity based on demonstrations. Although powerful and suitable for a wide variety of collaborative assembly tasks, the proposed method lacks flexibility and strong learning capabilities, as it heavily relies on the initial offline training phase. First, training is a time-consuming process that occupies resources that could be dedicated to the production, thus lowering the plant throughput and increasing costs. Moreover, additional training is required whenever changes in human activity occur. For this reason, the applicability to dynamic environments characterised by frequent modifications is problematic. This is the case, for instance, of flexible manufacturing plants where products are regularly introduced, modified, and dismissed. Second, all variants of the task must be known a priori and demonstrated by an expert user to collect data to learn the complete structure of the task. However, it is difficult to predefine all possible ways to perform an operation, as each worker might behave in different ways that are difficult to conceive by others. Also, the model obtained at the end of the training might describe many variants that will never be executed by the human during operation. Still, the presence of such variants is considered

during monitoring, leading to increased computational load and possibly lower performance.

For these reasons, in this Chapter we present a new strategy for the real-time monitoring of human activity that does not require any offline training and has the primary objective of estimating the expected duration of the ongoing operation. Indeed, the remaining time to completion of the current activity is the information that helps to improve agent coordination the most, especially in the presence of long and complex activities that are characterised by high variability in their execution. The proposed method is based on a modified version of the Dynamic Time Warping (DTW) algorithm and learns online from previous repetitions of the same activity. Previously unseen variants are automatically recognised and added to the activity model. By doing so, it considers the fact that humans can perform the same operation in many ways and with different speeds, occasional errors, and short pauses.

As far as previous works are concerned, [62] presents a strategy to evaluate the advancement of repetitive activities, which requires the observation of task-specific features to learn motion primitives and their effect on the overall progress. [84] proposes a method that combines optimization, supervised learning, and unsupervised learning to build a Bayesian model for partial temporal sequences alignment. Pauses in task execution are explicitly taken into account. In [70] the worker is supported during assembly tasks with information on the progress and the correctness of operations, which are retrieved through gesture recognition and detection of assembly parts. Instead, [27] detects errors in human activity by monitoring object manipulations. Abnormal behaviours are defined beforehand by domain experts using first-order logic. The method developed in [96] for early classification is robust against occlusions. It leverages a probabilistic representation of motion primitives learned from demonstration to align observations to the best fitting model. The problem of parsing complex tasks is tackled in [160], where trained Bayesian networks can also handle operation variants. The one being executed is retrieved in real-time by detecting specific primitive actions. Differently from our approach, all the aforementioned works either rely on offline training or prior expert knowledge of the task. Moreover, like [96], the proposed method handles possible lack of information in human monitoring that may occur during activity execution due to occlusions or tracking errors. Though unrelated to human monitoring, [25] presents a local alignment of vehicle trajectories that works offline based on Dynamic Time Warping. Vehicles can change roads following a path that is unknown a priori. Thus, trajectories do not align entirely with

any of the known references.

In the remainder of the Chapter, Section 3.1 presents the basic concept of progress-based estimation of activity duration. Then, Section 3.2 details the DTW-based algorithm used to solve the problem. Then, Section 3.3 extends the approach to manage task variants and human errors. The proposed strategy has been tested within a realistic assembly task. Results show its ability to give accurate predictions also in case of peculiar variants, such as those associated with errors.

3.1 Progress–based estimation of task duration

In most scenarios, especially in an industrial setting, the human is expected to repeat a given activity multiple times, i.e. a finite set of primitive actions needed to fulfil a task. In case of low variability, one could predict the duration of the ongoing human activity using data collected from past repetitions of the same task. Let $\mathcal{T}$ be the set of past durations and T_e the elapsed time from the start of the current task, then an estimate $\widehat{T}$ of its duration is the conditional expectation of past durations that are longer than T_e:

$$\widehat{T}(T_e) = \mathbb{E}_{\mathcal{T}}[\tau \,|\, \tau > T_e] \tag{3.1}$$

However, the elapsed time is representative of the actual activity advancement only under the assumption of low pace variability. Instead, human task execution speed may vary considerably, also due to distraction or fatigue. Thus, data from past executions are insufficient for accurate predictions, but additional information on the present activity is needed. Given a real-time estimate of the advancement $adv(T_e)$ of the task, one can extrapolate a better prediction from the average rate of progress as:

$$\widehat{T}(T_e) = \frac{T_e}{adv(T_e)} \tag{3.2}$$

To do so, a prototypical action, as well as a way to compare it with the current execution, are needed, which are described in the following. Figure 3.1 exemplifies the behaviour of the two approaches in case of high variability in the operation duration. Based on the previous available data, equation (3.1) predicts the current activity to last the same (1 s). Instead, it is clear that the task is being executed at a reduced pace ($adv(T_e) \approx 0.25$), which leads to a more likely duration of 2 s using equation (3.2).

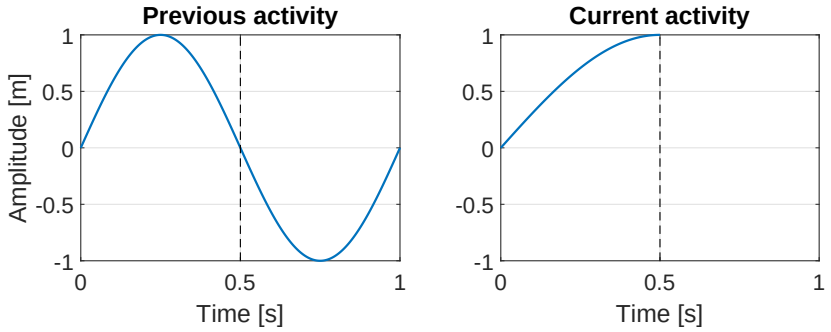


Figure 3.1: *Effects of high pace variability: based on the previous execution (left) the duration of the current activity (right) is predicted to be 1 s using equation* (3.1) *and 2 s using equation* (3.2)*.*

3.2 Dynamic Time Warping-based algorithm

This Section provides details on the modified version of the DTW algorithm used to compute a real-time estimate of the activity advancement. In particular, Open-ended DTW can be used to compare an incomplete sequence to a complete reference regardless of the changes in their speeds and obtaining as output the fraction of the reference that is matched by the input up to the current time instant. The proposed approach builds on the existing algorithms to obtain a method suitable for the problem of monitoring the human activity. Particularly, the developed algorithm must cope with the presence of occlusions and with asynchronous and uninformative human movements during task execution in order to obtain a more robust estimate of activity progress. Moreover, a method to update online the reference activity template is suggested.

Similar to Chapter 2, data on the movements of the operator's hands are required to monitor the evolution of the ongoing activity. Such information is obtained by means of the same tracking algorithm outlined in Section 2.1.1, which provides the 3D Cartesian coordinates of the human wrists and middle fingers. To compare different repetitions of the same activity, it would be beneficial to have normalised signals. However, the proposed algorithm runs in real-time, so that information on the signals as a whole are not available. One could express the motion of the features in terms of velocity, in order to remove the position bias. However, different execution speeds would result in different amplitudes. Moreover, a wavering behaviour of the operator would easily introduce undesired spikes in the velocity profile that would compromise time series comparison. On the

other hand, in industrial settings the workspace is usually well structured, so that the human motion is constrained by the position of product parts and assembly areas. For this reason, a description of task evolution in terms of feature positions attains better performance.

As already explained in Section 2.3.1, it is advisable to run two separate instances of DTW to account for the asynchronous motion of the two hands of the human. One receives as input the time series that describes the movements of the right hand, the other the time series related to the left hand. For each instance of he DTW algorithm, we recall that, given the input sequence $X = (x_1, \dots x_{|X|})$ that describes the partial execution of the ongoing activity and the reference template sequence $Y = (y_1, \dots y_{|Y|})$, the elements of the DTW matrix are computed as:

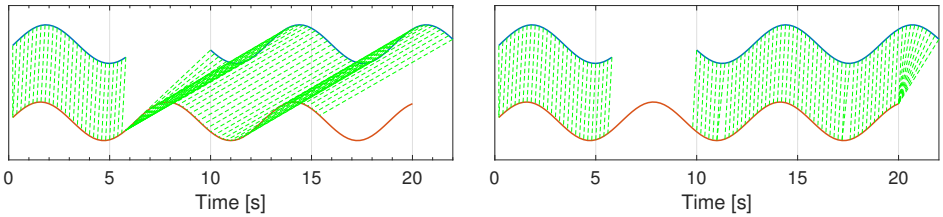
$$\begin{aligned} D(i,j) &= \delta + \|x_i - y_j\| \\ \delta &= \min\{D(i-1,j), D(i,j-1), D(i-1,j-1)\} \end{aligned} \tag{3.3}$$

from which the expression of the indexes of optimal truncation j_i^* and the warping path $\phi(i)$ for $i = 1, \dots |X|$ are:

$$j_i^* = \underset{j=1,\dots|Y|}{\arg\min}\, D(i,j) \qquad \phi(i) = \max\{j_i^*, \phi(i-1)\}$$

3.2.1 Occlusions handling

In order to achieve good performance, DTW algorithms require a consistent stream of measurements at run-time. However, the loss of data during the monitoring of human activity may occur due to tracking errors or occlusions of features. Two cases are possible: either only some of the signals are missing or all features are unavailable at the same time. Usually, tracking errors translate to isolated lost samples of all signals while occlusions are related to a subset of features but last for an indefinite time interval. If a partial sample is available, the algorithm can continue to work in the standard way, provided that the Euclidean distance in equation (3.3) is computed using only the non-occluded features. Conversely, the presence of full occlusions is critical for the DTW algorithm, as shown in Figure 3.2a. Standard DTW updates the cumulative distance $D(i,j)$ using equation (3.3) by searching for the minimum distance d among the neighbours of the point (i,j). When an occlusion occurs, input samples are discarded so that the first points before and after the occlusion are considered to be consecutive, which leads to a wrong warping of the signals. An occlusion handling mechanism has been designed that takes into account the occlusion length and is able to retrieve the correct alignment as depicted in Figure 3.2b. The



(a) (b)

Figure 3.2: *Behaviour of DTW in presence of occlusions without (a) and with (b) occlusion handling mechanism. The reference signal is shown in red and the input in blue; green lines show point-to-point associations. The modified algorithm accounts for the occlusion length and is able to retrieve the correct alignment, where some of the reference samples are not associated with any input sample.*

idea is to allow for many feasible matches during the occlusion and retrieve the correct one when new measurements are available. In fact, it is possible to extend the exploration space according to the occlusion length L_{occ} to take into account the possibility for the points of the reference to be associated with occluded input points. The modified rule that substitutes (3.3) reads as follows:

$$\begin{aligned} &\delta = \min\{D(i-1,j), D(i,j-1), D(i-k,j-1)\} \\ &k = 1, \ldots, L_{occ} \end{aligned} \tag{3.4}$$

In this way, all feasible alignments are taken into account as equally probable during the occlusion. The principle is illustrated in Figure 3.3, which shows an example of DTW matrix and optimal warping path in the presence of an occlusion. Exploiting the extended exploration space the algorithm is able to detect a missing peak in the input sequence and points 4 and 5 of the reference are associated with the occlusion.

3.2.2 Management of low-information template sections

During the execution of the task, there may exist parts during which one or both the operator's hands only perform small movements that are hardly captured by the feature tracking algorithm. An example is when one hand is used to hold the product, while the other reaches for a new part. In this case, both the reference template and the input sequence that describe the trajectory of the stationary hand will contain a section where the samples are essentially equal with each other. This results in a portion of the DTW matrix where all the elements tend to assume values close to each others.

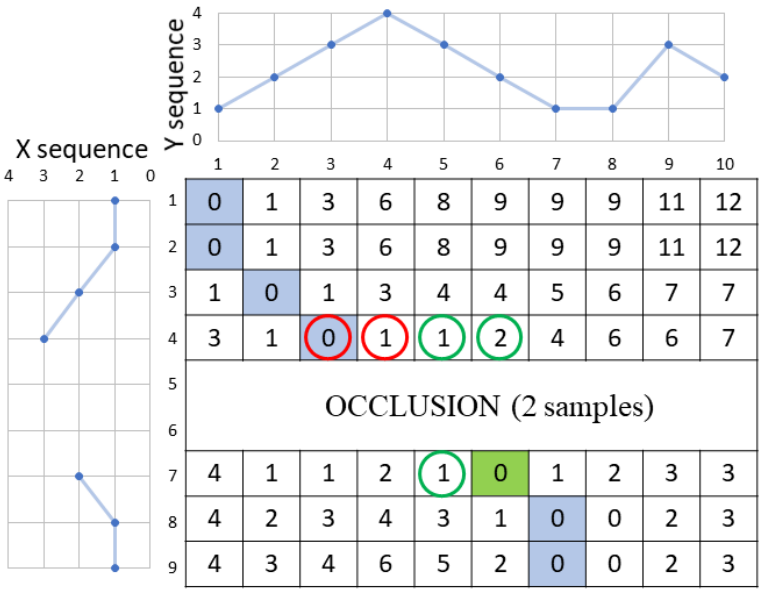


Figure 3.3: *DTW matrix with occlusion: to compute the first row after an occlusion, the standard exploration space in equation* (3.3) *(green circles) is extended according to* (3.4) *(red circles), accounting for the occlusion length.*

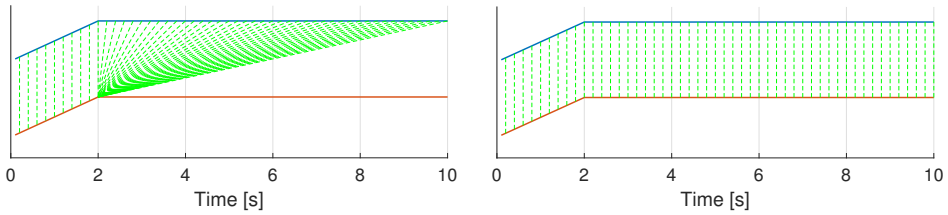
In fact, the Euclidean distance between any pairing of input points with reference points belonging to the low-information section will be similar. In this situation, the search of the optimal truncation index tends to underestimate the real advancement of the task and is easily prone to the presence of local minima. This is a problem for the correct alignment of the sequences, as exemplified in Figure 3.4a. To retrieve the desired behaviour shown in Figure 3.4b, which better describes the actual progress of the task, an additional contribution to the DTW cost is included, which is described in the following.

First of all, low-information sections of the template are detected as those during which the average speed of motion of the human hand is below a given threshold $\bar{v}$. Specifically, the i-th template sample belongs to a low-information section if:

$$\|y_i - y_{i-1}\| < \bar{v}$$

Then, a mode filter is exploited to remove very short sections and merge those that are close to each other.

Second, an additional term $\psi(i, j)$, named anti-pause contribution, is added to the computation of the elements of the DTW matrix to favour the



(a) **(b)**

Figure 3.4: *Behaviour of DTW in presence of low-information template sections (a) and desired behaviour (b). The reference signal is shown in red and the input in blue; green lines show point-to-point associations. In case of flat template, the modified algorithm computes an alignment that is more consistent with the actual advancement of the task.*

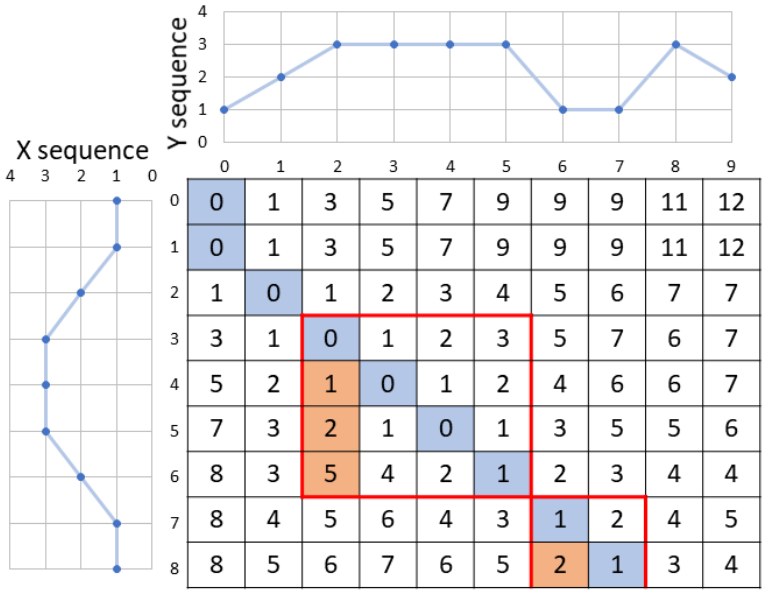
advancement of the warping path during low-information sections of the template. Specifically, the new rule that updates (3.3) reads as follows:

$$\begin{gathered} D(i,j) = \delta + \|x_i - y_j\| + \psi(i,j) \\ \delta = \min\{D(i-1,j), D(i,j-1), D(i-k,j-1)\} \\ k = 1, \ldots, L_{occ} \end{gathered} \tag{3.5}$$

The additional term should be different from zero only when aligning the input to the low-information section of the template. In fact, we do not want to force the advancement in other conditions, such as when the operator slows down or makes small pauses during execution, which must be considered to obtain the correct task progress.

To give a formal expression of $\psi(i,j)$, it is useful to define the following:

- A function $p(i)$ that returns *true* if the i-th template sample belongs to a low-information section and *false* otherwise;
- A variable that identifies the nearest start point of a low-information section:
$$j_P = \max_k \{k \leq \phi(i) | p(k) \wedge \neg p(k-1)\}$$
- A variable $i_P : j_P = \phi(i_P)$, which identifies the input sample that optimally aligns with j_P;
- A variable that identifies the nearest end point of a low-information section:
$$j_{\neg P} = \min_k \{k \geq \phi(i) | p(k) \wedge \neg p(k+1)\}$$
- A parameter ψ_0 that weights the anti-pause contribution.



	0	1	2	3	4	5	6	7	8	9
0	0	1	3	5	7	9	9	9	11	12
1	0	1	3	5	7	9	9	9	11	12
2	1	0	1	2	3	4	5	6	7	7
3	3	1	0	1	2	3	5	7	6	7
4	5	2	1	0	1	2	4	6	6	7
5	7	3	2	1	0	1	3	5	5	6
6	8	3	5	4	2	1	2	3	4	4
7	8	4	5	6	4	3	1	2	4	5
8	8	5	6	7	6	5	2	1	3	4

Figure 3.5: *DTW matrix with low-information section in the template sequence: optimal warping path obtained with (blue) and without (red) the anti-pause contribution in equation* (3.5)*. Red squares highlight the region of the DTW matrix where the anti-pause contribution is active. The anti-pause contribution allows for a better alignment that avoids pauses and jumps.*

Then, one has:

$$\psi(i,j) = \begin{cases} \psi_0 \left|(j - j_P) - (i - i_P)\right| & p(\phi(i-1)) \wedge (j > j_P) \wedge (j > j_{\neg P}) \\ 0 & \text{otherwie} \end{cases}$$

That is, a linear alignment is assumed during the low-information section and alignments that deviate from this assumptions are penalized proportionally to the magnitude of the deviation. Figure 3.5 depicts the effects of the anti-pause contribution, which allows for a better alignment that avoids pauses and jumps in the warping path.

3.2.3 Warping paths merging

As already said, two instances of DTW run concurrently to monitor the two operator's hands separately. Therefore, two warping paths are obtained, which we can call $\phi^{dx}(i)$ and $\phi^{sx}(i)$. In order to compute a single value for the progress of the activity, the warping paths are merged together accord-

ing to the following weighted average:

$$\begin{cases} \bar{\phi}(k) = \frac{\phi^{dx}(k)+\varepsilon(k)\phi^{sx}(k)}{1+\varepsilon(k)} & \phi^{dx}(k) \geq \phi^{sx}(k) \\ \bar{\phi}(k) = \frac{\phi^{sx}(k)+\varepsilon(k)\phi^{dx}(k)}{1+\varepsilon(k)} & \phi^{dx}(k) < \phi^{sx}(k) \end{cases} \tag{3.6}$$

where $\bar{\phi}(k)$ is the average path and ε a varying weighting coefficient, which is chosen so to have:

$$\begin{aligned} \varepsilon(k) &= 1 \quad \text{if} \quad |\phi^{dx}(k) - \phi^{sx}(k)| = 0 \\ \varepsilon(k) &\to 0 \quad \text{if} \quad |\phi^{dx}(k) - \phi^{sx}(k)| \to \infty \end{aligned}$$

For instance, one can choose:

$$\varepsilon(k) = e^{-\xi|\phi^{dx}(k)-\phi^{sx}(k)|}$$

with ξ a design parameter used to determine the decreasing speed. The idea of equation (3.6) is to assign more weight to the path that estimates the highest advancement. This proves to be of fundamental importance when one of the two instances of the algorithm temporarily attains low performance due to uninformative human movements or the presence of occlusions. The effect is a reduced advancement rate of such instance. Therefore, giving more weight to the highest advancement, the overall estimate progresses without interruptions and is less affected by the local error.

3.2.4 Activity duration estimate

Given the optimal warping path, one can compute the progress of the activity at time T_e, with k the number of available input samples at T_e, as:

$$adv(T_e) = \frac{\phi(k)}{|Y|} \tag{3.7}$$

Then, equation (3.2) can be exploited to obtain an estimate of the expected duration. However, it is not convenient to directly use the output of the DTW-based algorithm in the equation, as it suffers from a couple of problems. Firstly, at the beginning of the activity, only a few data are available and the progress estimate is poor. Moreover, a small difference in the advancement leads to significant variations in the predicted duration, as it is computed by extrapolating the average rate of progress. Overall, the effect is that the raw DTW estimate shows a bad initial transient that is exemplified by the yellow line in Figure 3.7a. Secondly, during a low-information section of the activity, e.g. when the operator is mostly stationary, DTW

results typically exhibit a temporary stop in the advancement followed by a sudden jump when motion is again informative. The problem has been already discussed and mitigated with the introduction of the anti-pause contribution to the DTW cost, but its effects might still be significant (an example is shown in Figure 3.7b at $t \approx 10s$). The consequence is an increase in the duration estimate, followed by a sharp correction. These two behaviours of the prediction of the expected duration do not reflect the true task execution, but are due to bad estimates retrieved by the DTW. To get rid of the initial transient produced by the DTW-based estimate, it is possible to use the elapsed time-based prediction returned by equation (3.1) until a certain level of activity progress is reached. In fact, when only little information is available on the present task, an estimate based on past durations is more accurate. Instead, a modification to the optimal warping path is introduced to handle low-information sections with the aim of smoothening flat parts and jumps and obtain a more stable duration prediction, which is described in the following.

Pauses and jumps in the advancement can be characterised with respect to the warping path. Specifically, pauses occur when $\bar{\phi}(k) = \bar{\phi}(k-1)$. As long as the warping path is increasing, the modified path $\breve{\phi}$ is taken equal to the DTW output $\bar{\phi}$. When a pause is detected, a linear growth is imposed with speed equal to the average rate of advancement of the current activity. When DTW output resumes to grow or there is a jump, linear growth is stopped and $\breve{\phi}$ is saturated until the DTW estimate becomes greater than the saturation value. This ensures that the non-decreasing constraint on the warping path is satisfied. The computation of the modified warping path can be modelled using the finite state automaton in Figure 3.6, with three states that describe the possible situations: N for *Normal behaviour*, L for *Linear growth* and S for *Saturated*. In particular, we compute $\breve{\phi}(k)$ as:

$$\begin{aligned} N: &\quad \breve{\phi}(k) = \max\{\bar{\phi}(k), \breve{\phi}(k-1)\} \\ L: &\quad \breve{\phi}(k) = \frac{\breve{\phi}(k-1)}{k-1}k \\ S: &\quad \breve{\phi}(k) = \breve{\phi}(k-1) \end{aligned} \tag{3.8}$$

where $\breve{\phi}(k-1)/(k-1)$ represents the activity average rate of progress up to sample $k-1$. Transitions between states are triggered by:

$$\begin{aligned} N \to L: &\quad \bar{\phi}(k) = \bar{\phi}(k-1) \\ L \to S: &\quad \bar{\phi}(k) > \bar{\phi}(k-1) \vee \breve{\phi}(k-1) - \bar{\phi}(k) > \delta_M \\ S \to N: &\quad \bar{\phi}(k) \geq \breve{\phi}(k-1) \end{aligned}$$

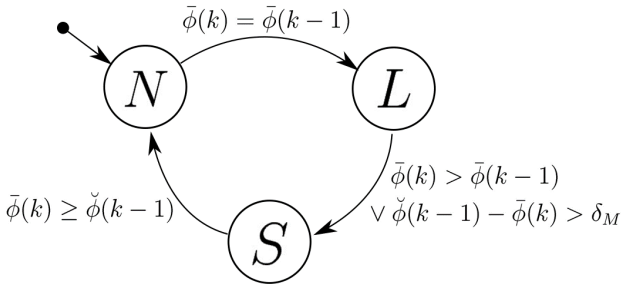


Figure 3.6: *Finite state automaton that describes the warping path modification algorithm. In each state the modified path $\breve{\phi}$ is computed according to equations* (3.8)*.*

The additional second condition on the $L \rightarrow S$ transition is used to put a limit to the linear growth in order to take into account cases when the pause in the activity advancement is not due to a DTW algorithm error, but reflects a real stop in the execution performed by the human operator. Complete stops are rare events that usually occur in the presence of human mistakes and are not followed by a corrective jump in the DTW output. A linear growth would lead to move away from the actual activity evolution and would deteriorate the algorithm performance. Therefore, the modified path saturates if its distance from the estimated one becomes greater than a limit threshold δ_M.

Overall, the expected duration is estimated as:

$$\widehat{T}(T_e) = \begin{cases} \mathbb{E}_{\mathcal{T}}[\tau \mid \tau > T_e] & adv(T_e) \leq \alpha \\ \frac{T_e}{adv(T_e)} & adv(T_e) > \alpha \end{cases} \tag{3.9}$$

where the advancement is computed using the modified warping path $\breve{\phi}$ in equation (3.7) and equation (3.1) is used to neglect the initial transient, when little information on the ongoing execution is available, until the advancement exceeds the threshold α.

3.2.5 Selection of the reference

The DTW-based algorithm works by comparing the current execution with a reference template. However, the proposed method does not require prior knowledge of tasks or learning phase to increase ease of use and flexibility. Instead, the algorithm selects the best template among the temporal sequences it has collected from past executions of the monitored task. Clearly, the first instance of a given activity cannot be monitored and is taken as the first reference. However, it is expected that training of the op-

erator increases by repeating the same tasks, whereas isolated uncertainties and errors may still occur. The correct rule for activity template update should address both issues. Several possibilities were tested on different assembly operations and compared with each other. Changing the template whenever a new one is available benefits from better operator's training, but is prone to execution errors. Computing an average template from all the available ones is difficult, since series alignment must be considered, and worsens the performance, since it distorts the shape of the time series. The best results have been obtained changing the reference only when the last execution is shorter than the current template. The underlying idea is that a short activity is more likely to be free from pauses and errors. Furthermore, improved operator training generally leads to faster executions.

3.2.6 Experiments

Warping path modifications

The effectiveness of the proposed algorithm has been tested on realistic assembly operations. First of all, the advantages of path modification in terms of better activity duration estimate have been investigated. Figure 3.7 reports the results of 3 different assembly operations repeated 40 times each. A Microsoft Kinect is used to monitor the human operator's activity, as described in Section 2.1.1. Box-plots in Figure 3.7c show the average prediction error obtained with or without modifying the DTW output, which is computed with respect to the actual operation duration T^* as:

$$e_m = \int_0^{T^*} \frac{|\hat{T}(\tau) - T^*|}{T^*} d\tau \tag{3.10}$$

Filtering the output of the DTW-based algorithm and excluding the initial transient significantly reduces both mean and error variance, achieving an average error of less than 1s on activities lasting 15-45s. Figures 3.7a and 3.7b give an example of the improved performance. Figure 3.7b compares a profile of advancement obtained from the DTW with the corresponding modified one: at 10s a pause-jump event occurs and the modified path stays closer to the ideal activity advancement. Moreover, it is apparent from Figure 3.7a that most of the error using the DTW algorithm concentrates during the initial transient (yellow and red lines), whereas the elapsed time-based estimation provides a stable prediction of the duration (the initial flat part of the blue line).

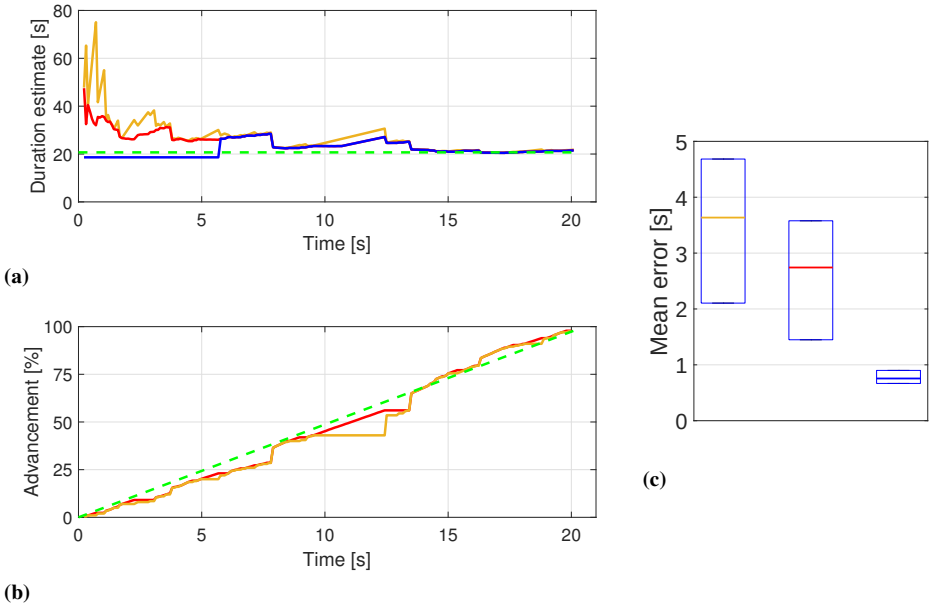


Figure 3.7: *Effects of DTW path modification: example of duration estimate evolution (a), activity advancement (b) and mean estimation error with quartiles (c). In all plots yellow lines refers to the unmodified DTW output, red to the filtered one, blue also neglects initial transient, green indicates the actual task duration and the ideal advancement.*

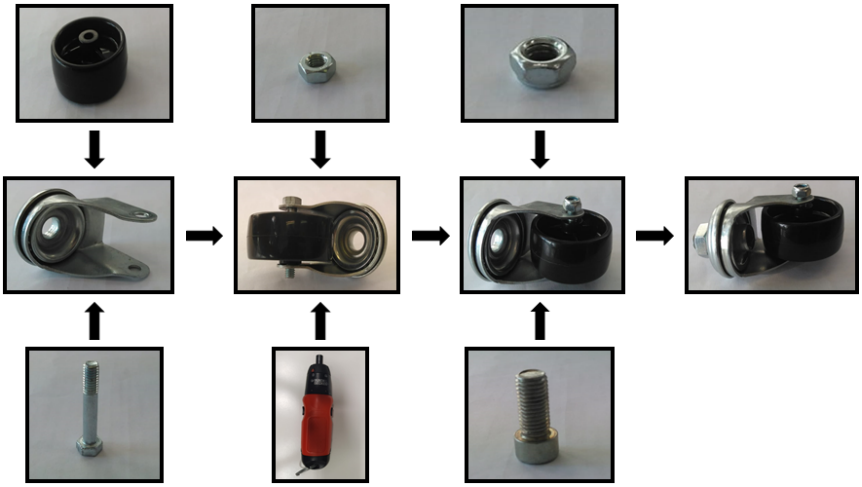


Figure 3.8: *Diagram of the assembly operation of a caster wheel. The operator picks the main body and positions the rubber wheel. Then, the wheel is fixed using a screw with the help of an electric tool. Another screw is inserted and fastened to allow for wheel mounting.*

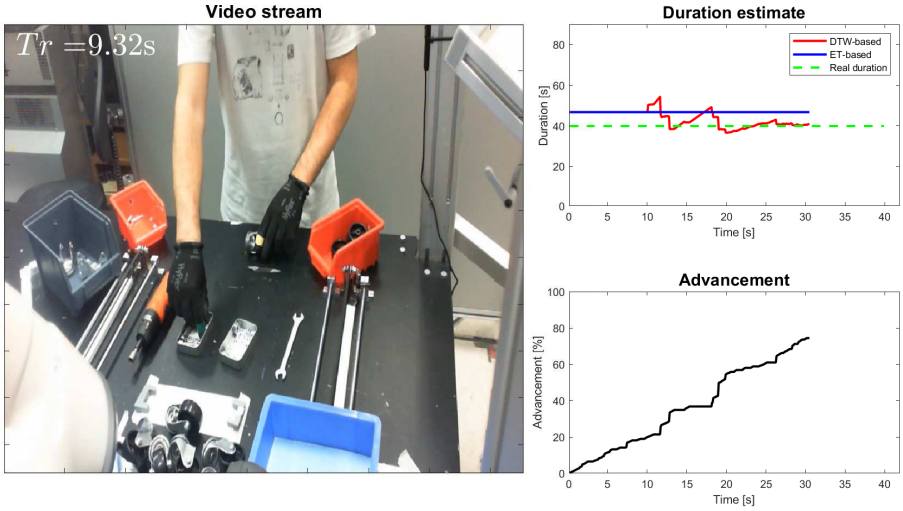


Figure 3.9: *Snapshot of the operator performing the assembly task. Activity advancement profile (bottom-right) and duration estimation (top-right) as given by the algorithms based on the elapsed time (blue) and DTW (red).*

Algorithm performance

To test the overall performance of the proposed algorithm, multiple assemblies of a caster wheel have been performed, which comprises a wide variety of actions – such as reaching motions, peg insertions, manual and automatic screwing. The scheme of the assembly steps is shown in Figure 3.8. First, the operator picks the main body and positions the rubber wheel. Then, the wheel is fixed using a screw with the help of an electric tool. Finally, a second screw is inserted and fastened to allow for wheel mounting. Data were collected from a total of 68 assembly operations grouped in 8 runs, each one composed of 8 or 9 repetitions of the task. The human activity has been monitored using the proposed DTW algorithm. The estimate based on past durations and the elapsed-time, given by equation (3.1), has been also computed for comparison, and referred to as the *ET prediction*. In order to test the ability of the proposed algorithm to learn the reference template of the activity, it has been reset to start with a blank template at the beginning of each new run. The mean duration of the assembly operation was of 40.8s, with a minimum of 31.0s and a maximum of 59.6s, thus exhibiting high variability that is mainly due to the presence of insertion and screwing tasks. Figure 3.9 presents a snapshot of the operator performing the assembly task. The associated activity advancement curve and duration estimates obtained using the ET and DTW algorithms up to the current time

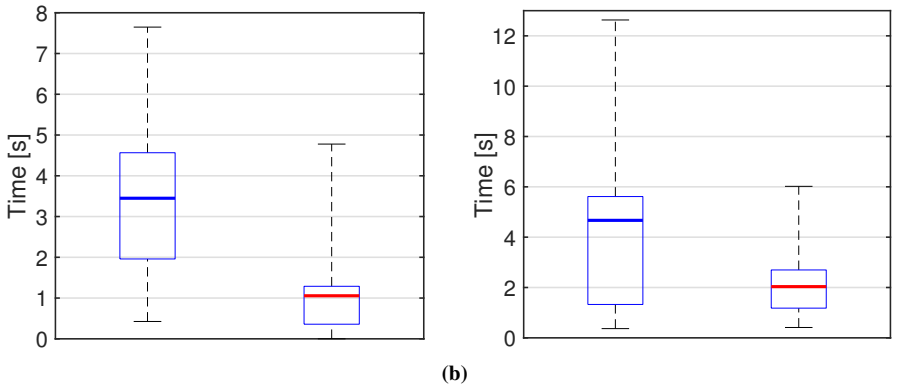


Figure 3.10: *Final duration error e_f (a) and mean duration estimate error e_m (b) for ET prediction (blue) and DTW-based algorithm (red). Boxes indicate the 25-75 percentiles, whiskers the extreme points. DTW shows a strong reduction in both mean end variance of the errors.*

instant are also included.

Two performance indexes have been used to evaluate the average estimation errors of activity duration and discuss the overall performance of the proposed algorithm. Results are reported in Figure 3.10. Specifically, Figure 3.10a refers to the error at the time instant in which the activity ends, that is:

$$e_f = |\hat{T}(T^*) - T^*|$$

This indicates whether the algorithm converges to the correct estimate. Instead, Figure 3.10b gives the average prediction error e_m during task execution according to equation (3.10), which is linked to the ability to give early correct estimates. In both cases, the proposed algorithm attains superior performance: the average final error drops from 3.5s to about 1s while e_m reduces from 4.7s to 2s. Also the error variance is significantly reduced with respect to the ET prediction.

To highlight the learning capabilities of the algorithm, it is possible to break down the average prediction error e_m according to the position of each activity inside its run. The error value exhibits a decreasing trend, as shown in Figure 3.11: as the algorithm is able to choose the best template among past executions, the prediction becomes more accurate. When 8 past repetitions are available DTW attains an error reduction of about 25% with respect to the case when only one past execution is available as the reference template. This also confirms the intuition that a shorter template generally describes a better execution of the task.

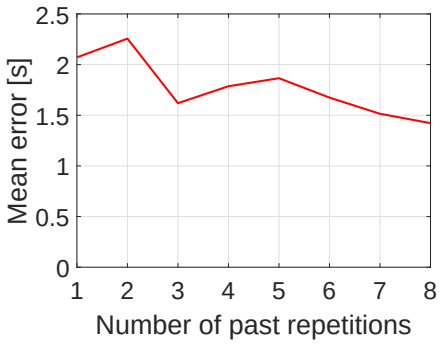
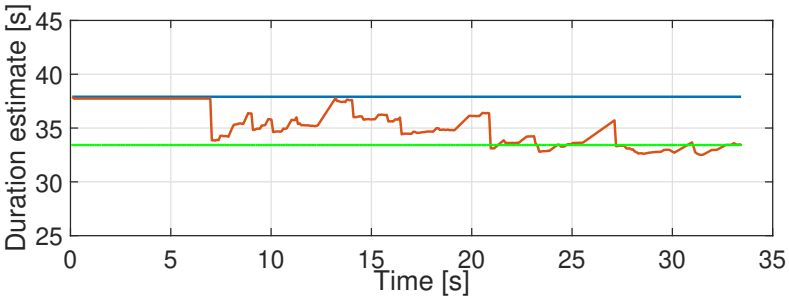


Figure 3.11: *Template learning leads to a decreasing trend in the value of the average prediction error e_m as the number of repeated executions of the activity increases.*

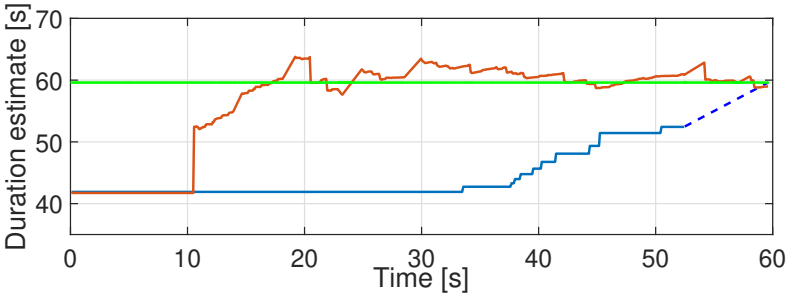
Finally, Figure 3.12 gives insight on the better performance of the proposed algorithm showing results for notable critical cases. Firstly, whereas the ET prediction benefits of a large past activity duration database, the DTW algorithm is capable of good monitoring performance starting from the first repetition after learning an activity reference. Figure 3.12a presents a comparison of the typical behaviour at the second iteration of the activity: the ET prediction is constant and equal to the only past duration available, while the DTW algorithm is already able to follow the advancement of the ongoing task. Figures 3.12b and 3.12c show what happens in case of particularly long or short duration of the activity, respectively. In both cases, the ET prediction gives a wrong early estimate and it does not have enough information to converge to the correct final duration. Instead, both situations are not critical for the DTW-based algorithm, which is designed to inherently take into account high variability in the speed of execution of the operation. The case in Figure 3.12b is of special importance, as uncertainties and difficulties in the execution of the activity by the human operator mostly translate in a reduction of the rate of advancement. On the other hand, faster executions appear as the operator training increases.

3.3 Robust monitoring with task variants and errors

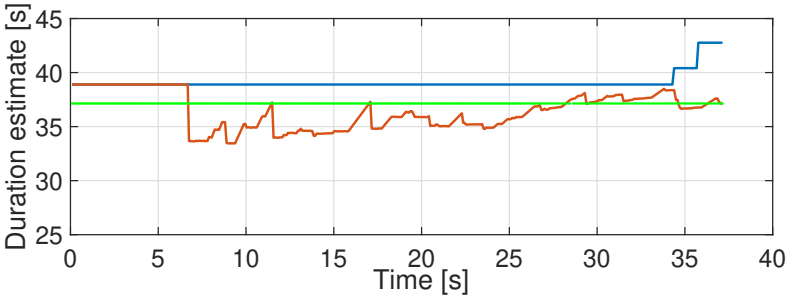
In general, the human can perform the same task in multiple ways and the movements and execution times among the different variants can change considerably. In this case, the algorithm presented in Section 3.2 offers poor performance, as it is not possible to determine a unique template that is consistent with the operator's movements for all variants. Instead of handling each variant separately, i.e. having one reference for each variant



(a)



(b)



(c)

Figure 3.12: *Examples of critical results for activity monitoring: second execution (a), long execution (b) and short execution (c). Comparison among DTW algorithm (red), ET prediction (blue) and actual activity duration (green).*

and running multiple instances of the DTW-based algorithm concurrently, a richer template must be considered, able to efficiently describe the structure of the task with all its known variants. Besides significantly reducing the computational load to preserve real-time performance, this allows improving the estimate of task progress, recognition of variants, and template learning. However, a method to identify in real-time the task variant being performed and to compare each execution with the correct reference is needed.

With this aim, the task is segmented online relying on the automatic detection of a set of features $F = \{f_1, \dots f_{|F|}\}$. A variant $V_i \in V$ of the task is then defined as the ordered sequence of features that occur from its start to its end. Features identify specific events during human activity (i.e. the picking of a specific part or tool during assembly operations) and are defined a priori based on general knowledge of the task. The definition of the features is guided by two main considerations:

- They should allow capturing all possible behaviours of the operator;
- They should allow applying the algorithm presented in Section 3.2 at the segment level, i.e. the prototypical execution of each segment can be described by a single temporal sequence.

Even so, there is no need to explicitly know and define all variants beforehand, as the template is built at run-time, adding previously unseen variants and learning better references for the known ones.

In the following, Section 3.3.1 describes how to build and update the activity template, Section 3.3.2 addresses the online recognition of variants, and Section 3.3.3 deals with the estimate of the advancement and the duration of the task. Finally, Section 3.3.4 discusses the experimental results and performance of the complete algorithm.

3.3.1 Reference template structure

From the definition of variant, each execution of the task is described by a sequence of segments, which are delimited by the detection of two features. Each segment is a time series that describes the human motion by means of notable features as in Section 2.1.1. The set T collects all known reference sequences t_{ij} composed of $|t_{ij}|$ samples that describe the human's motion along the segment that goes from f_i to f_j. Conventionally, the start of the activity is associated with a fictitious feature f_0 common to all variants. Then, variants differ from each other after a possible common initial part,

defined by the same sequence of features, during which the operator performs the actions in the same way. As a result, the structure of the task can be represented as a tree, with nodes associated with segments and branches marking the presence of variants at that point of the execution.

More formally, let $\Theta = (N, A)$ be the template tree of the task, with N the set of nodes and A the set of arcs. A node $n_i \in N$ is defined as a 5-tuple $n_i = (o_i, d_i, p_i, C_i, r_i)$ where o_i and d_i are the origin and destination features of the segment, respectively (for the root node $o_0 = d_0 = f_0$). Each node is associated with the reference template t_{o_i,d_i} that describes the execution of the segment from o_i to d_i. For complex tasks, more than one node with the same origin and destination may exist, i.e. $\exists i \neq j : o_i = o_j \wedge d_i = d_j$, which refers to the same common template. p_i is the parent node, which has the destination equal to the child's origin. C_i is the set of children that collects all nodes whose parent is n_i. Arcs in A link parents to children, i.e. $a_{ij} \in A \Leftrightarrow n_i = p_j$. Finally, r_i is the number of times the segment has been already executed during past repetitions of the activity. Given the tree, a variant $V_x = (f_0, \dots f_N)$ is equivalently defined as the ordered set of N-1 nodes connecting the root to a leaf that refer to the segments that compose the task.

As already stated, when the monitored task is performed for the first time, the template tree is empty. In fact, the proposed method does not rely on offline training but learns online from past repetitions of the same activity following the concept initiated in Section 3.2. This increases applicability in dynamic environments, such as present-day manufacturing, which is characterised by fast changes in production and, consequently, in human activities. As a result, the first execution of the task cannot be monitored but is taken as the first reference. Each time the occurrence of a feature is detected, i.e. a segment of the task is completed, a new node is added to the tree and the temporal sequence that describe the human's movements is taken as the reference template for such segment in the set T.

To better understand the behaviour of the algorithm for subsequent repetitions of the same task one can refer to Figure 3.13, which shows an example of template tree for a task with three variants $V_1 = (f_0, f_1, f_2, f_3)$, $V_2 = (f_0, f_2, f_1, f_3)$ and $V_3 = (f_0, f_2, f_3)$ after eight repetitions of the activity. The set of reference sequences is $T = \{t_{01}, t_{02}, t_{12}, t_{13}, t_{21}, t_{23}\}$, noting that nodes n_3 and n_7 refer to the same segment. Initially, the algorithm starts to monitor the first segment. To do so, it runs one instance of the DTW-based algorithm for each child of the root node (n_1 and n_4 in the Figure). In this way, the ongoing human operation is compared to all the known variants that depart from the root node. The algorithm under-

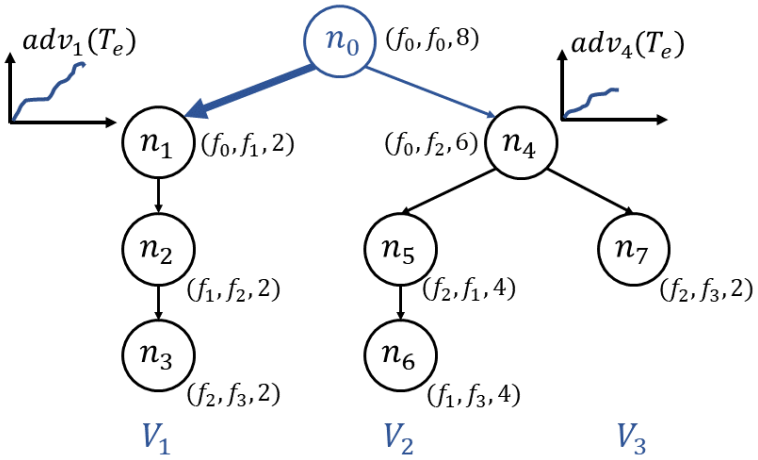


Figure 3.13: *Example of template tree of a task with three variants. Label for the i-th node is (o_i, d_i, r_i). After n_0, two DTWs run concurrently, one for each branch. The width of the blue arrows is proportional to their current probability.*

stands which is the most probable variant being performed and evaluates the advancement and the expected duration of the task accordingly. When the next feature is detected (suppose f_1), marking the end of the current segment, the algorithm proceeds to monitor the next segment.

The current node becomes the appropriate child of the root node (which is n_1 in case of the detection of f_1) and new instances of DTW are run for each child n_i of the current node to monitor the next part of the task. The input is the sequence of the human hand positions collected from the start of the segment (i.e. the last feature detection) to the current time, which is compared to the reference sequence $t_{o_i,d_i} \in T$. Any time a segment is completed, if the temporal sequence that describes its last execution is shorter than the current template in T, the former is taken as the new reference for such segment, in the same way as described in Section 3.2.5. Conversely, if none of the known children corresponds to the performed segment, i.e. the human is following a new, previously unseen, variant of the task, a new branch is added to the template tree. From this point, the progress of the activity cannot be estimated, as the future evolution of the ongoing variant is unknown. However, human motion is still monitored to segment the remaining part of the activity and update the template accordingly in order to include the new variant in the task structure.

Noting that multiple nodes in the tree can refer to the same task segment, such as n_3 and n_7 in Figure 3.13, one might wonder whether a graph-like structure of the activity could improve representation efficiency

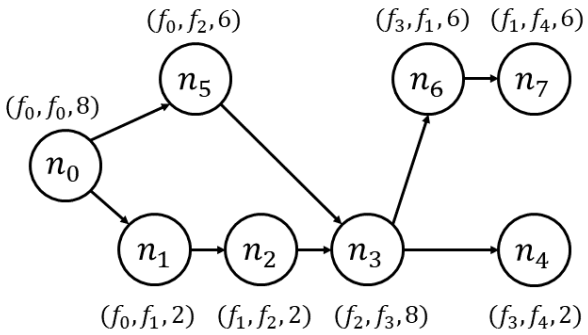


Figure 3.14: *Example of graph-like activity structure that does not allow to properly recognise variants.*

by eliminating unnecessary nodes. However, a graph model would make it impossible to correctly distinguish the variants that compose the activity due to the presence of joint-fork pairs and/or loops. For instance, two variants $V_1 = (f_0, f_1, f_2, f_3, f_4)$ and $V_2 = (f_0, f_2, f_3, f_1, f_4)$ would produce the graph shown in Figure 3.14, where a joint-fork connection is present. However, looking at the graph, one could say that also variants $(f_0, f_1, f_2, f_3, f_1, f_4)$ and (f_0, f_2, f_3, f_4) are possible. This would make it impossible to correctly recognise the variant being performed and estimate the advancement of the activity as described in the following. In the case of loops, the problem would be even worse, as the graph would contain paths of infinite length that could be considered as feasible variants. On the other hand, although the same segment can appear in the tree multiple times as different nodes, only one reference template is kept in memory in the set T.

3.3.2 Early recognition of task variants

During the monitoring of a task, the exact segment being executed is unknown. While the origin feature is known, the destination is determined only after the completion of such a segment. However, early information on which variant is the one being performed is crucial to estimate the progress of the task. This is true both when there is a fork in the tree (and multiple DTWs are running) as well as when the current node has a single child, since the human might always follow a new variant.

The proposed method for the early recognition of task variants computes a probability value for each of the branches departing from the current node $\bar{n}$, related to the last completed segment. To do so, it exploits a similarity measure that derives from the cumulative distances provided by the DTW

applied at the segment level. For each $n_i \in \overline{C}$, two DTW matrices are present to monitor the movements of the right and the left hand, respectively. A unique cost $\overline{D}_i$ is taken as:

$$\overline{D}_i(k) = \min\{D_i^{sx}(k), D_i^{dx}(k)\}$$
$$D_i^{sx}(k) = \min_j \; D_i^{sx}(k,j) \qquad\qquad D_i^{dx}(k) = \min_j \; D_i^{dx}(k,j)$$

where k is the number of samples in the input sequence, and $D_i^{sx}(k,j)$ and $D_i^{dx}(k,j)$, $\forall j$, are the last rows of the two DTW matrices computed according to equation (3.5). Then, the similarity measure $\widetilde{D}_i(k)$ is given by the convex combination of two different contributions with respect to a design parameter $\gamma \in [0,1]$:

$$\widetilde{D}_i(k) = \gamma D_i'(k) + (1-\gamma) D_i''(k)$$
$$D_i'(k) = \frac{\overline{D}_i(k)}{k} \qquad\qquad D_i''(k) = \overline{D}_i(k) - \overline{D}_i(k-1)$$

In particular, D_i' is the cumulative distance normalised over the input length, which is a global measurement of the similarity between the input and reference sequences and evolves smoothly during the evolution of the task. Instead, D_i'' considers the local rate of growth of the DTW cost and is more reactive in detecting changes in the operator's movements with respect to the reference template.

Given the similarity measure, the current probability P_i of each branch is computed using a recursive Bayesian classifier (similar to the one presented in [170]). The prior probability of each child n_i of $\bar{n}$ is based on historical data and given by:

$$P_i(0) = \frac{r_i}{\bar{r}}$$

with $\bar{r}$ the past executions of $\bar{n}$. The value is updated with each new input sample according to the following recursive rule:

$$\widetilde{P}_i(k) = P_i(k-1) \cdot f(\widetilde{D}_i(k)) \tag{3.11}$$

where f is the probability density function of a Gaussian distribution with zero mean and standard deviation $\sigma = 0.27$, whose value has been determined as the one minimizing the classification error during preliminary tests. As the cost $\widetilde{D}_i(k)$ increases, i.e. the sequences are more dissimilar, the probability tends to decrease.

The sum of all probabilities returned by equation (3.11) might be greater than 1. Thus, a normalization is introduced as:

$$P_i(k) = \frac{\widetilde{P}_i(k)}{\max\{1, \sum_i \widetilde{P}_i(k)\}}$$

Notice that values are normalised only when their sum is greater than 1. Otherwise, if the probability of all known variants is low, the algorithm assumes the existence of a new variant. Specifically, $1 - \sum_i P_i(k)$ is taken as the probability that the operator is performing an unknown variant of the task.

3.3.3 Estimate of task advancement and expected duration

Since the DTW-based algorithm is applied at the segment level, the equation (3.7) used in Section 3.2 to monitor the operation with a single template would return the advancement of the current segment instead of the one of the whole task. As variants might differ in length, the advancement of the task depends on which variant is considered as the one being executed. Still, we aim to find a single value that best describes the overall progress of the activity and leads to the most accurate prediction of its duration. To do so, all the variants that are feasible at the current time instant must be considered, as we cannot know the future human behaviour.

Let $\bar{n}$ be the current node, $n_i \in \overline{C}$ one of its children, and $\mathcal{V}_i = \{V_x \in V | n_i \in V_x\}$ the set of all variants that contain n_i. $\mathcal{V}_i$ identifies the i-th branch that departs from the current node of the template. Moreover, let consider a partition $V_x = (V^{pre}, V_x^{post}) : V^{pre} = (n_0, \dots \bar{n})$, $V_i^{post} = (n_i, \dots)$ and note that V^{pre} is common among all $V_x \in \mathcal{V}_i$. Then, the advancement of the task assuming the i-th branch that departs from $\bar{n}$ to be the correct one is computed as:

$$adv_i(T_e) = \frac{L^{pre} + \breve{\phi}_i(k)}{L^{pre} + L_i^{post}} \tag{3.12}$$

with k the number of available input samples at time T_e. Moreover, L^{pre} is the length of the fraction of the task template that has been already executed, equal to:

$$L^{pre} = \sum_{n_i \in V^{pre}} |t_{o_i,d_i}|$$

Instead, L_i^{post} is an estimate of the length of the remaining part of the task template to be performed, which is computed considering all variants be-

longing to the i-th branch $\mathcal{V}_i$. Let l_{V_x} be the length of the part of the reference template of V_x that remains to be performed and π_{V_x} the probability to be the variant performed by the human based on past data, that is:

$$l_{V_x} = \sum_{n_j \in V_x^{post}} |t_{o_j,d_j}| \qquad \pi_{V_x} = \prod_{n_j \in V_x^{post}} \frac{r_j}{r_{p_j}}$$

Then, L_i^{post} is computed as the value that minimises the average error with respect to the length L^* of the ground-truth variant:

$$L_i^{post} = \arg\min_{L^*} \sum_{V_x \in \mathcal{V}_i} \pi_{V_x}(L^* - l_{V_x})^2 = \sum_{V_x \in \mathcal{V}_i} \pi_{V_x} l_{V_x}$$

Which is to say that the estimated length of the remaining part of the template is the convex combination of the remainder of the template of all variants that belong to the branch, weighted with respect to their historical probability.

Referring again to Figure 3.13, when a new repetition of the activity starts, the current node is set to $\bar{n} = n_0$. Since two branches are present ($\mathcal{V}_1 = \{V_1\}$ and $\mathcal{V}_4 = \{V_2, V_3\}$), two instances of the DTW-based algorithm monitor the ongoing task with templates t_{01} and t_{02} associated with $\mathcal{V}_1$ and $\mathcal{V}_4$, respectively. Since no segment has been already completed $L^{pre} = 0$. Besides, $L_1^{post} = l_{V_1}$ and $L_4^{post} = 0.67 l_{V_2} + 0.33 l_{V_3}$, being for instance $l_{V_1} = |t_{01}||t_{12}||t_{23}|$.

Assuming that the i-th branch is the one being executed, one can predict the duration $\widehat{T}_i$ of the activity with the progess-based estimate in (3.9) using the task advancement computed with equation (3.12). Then, the final prediction $\widehat{T}(T_e)$ of the duration of the task is obtained by weighting the values obtained for each branch according to their current probability to be the one being performed, as given by the method explained in Section 3.3.2:

$$\widehat{T}(T_e) = \sum_{n_i \in \overline{C}} P_i(k) \widehat{T}_i(T_e) + \left(1 - \sum_{n_i \in \overline{C}} P_i(k)\right) \overline{T}(T_e)$$

The last term accounts for the possibility that the operator is following an unknown variant and $\overline{T}(T_e)$ is the average duration of the task computed with the elapsed time-based prediction (3.1). As the algorithm becomes more certain about the variant being performed, the influence of unlikely branches vanishes and the prediction converges to the value associated with the correct variant. In case a new node is added to the template at the end of a segment, i.e. the human is following a new variant, the remaining part

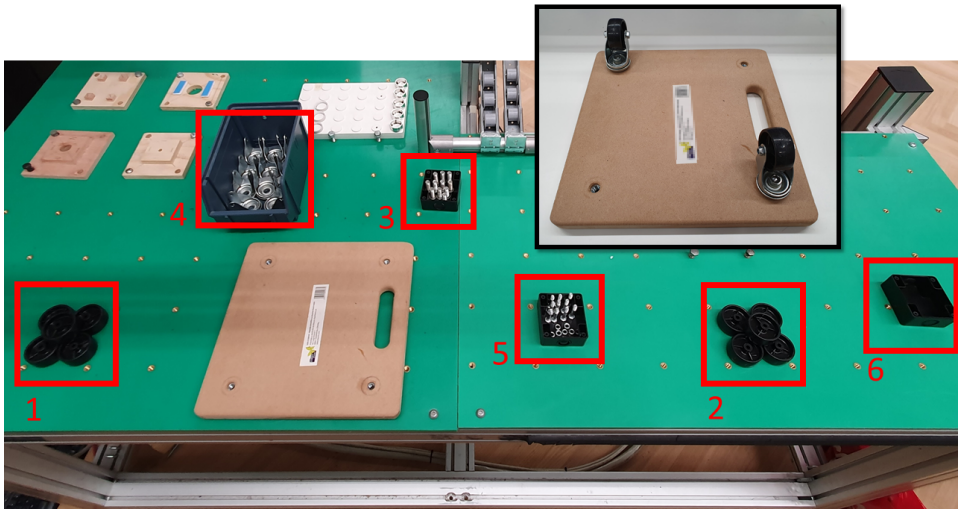


Figure 3.15: *Picture of the workspace and of the final product.*

of the activity cannot be monitored. Instead, the next segments are added to the tree and $\overline{T}(T_e)$ is returned as the best prediction for the duration of the ongoing task.

3.3.4 Experiments

The experimental campaign aimed to verify the variant recognition mechanism, the construction of the template tree, and to assess the performance of the proposed algorithm in terms of prediction errors of the activity duration. Similar to the experiments in Section 3.2.6, the movements of the human have been tracked in terms of wrists and index fingers trajectories following the approach outlined in Section 2.1.1. The task consisted in the partial assembly of a wheeled base, during which only two out of four wheels were mounted (see box in Figure 3.15). The product parts were stored in boxes on the worktable as shown in Figure 3.15: wheels for the left (1) and right (2) casters, screws to fix the casters to the base (3), swivel forks (4), and screws and nuts to fix the wheel to the fork (5). Besides, zones are present to store defective parts (6), the bases, and the completed products (not shown in the picture). For the purpose of task segmentation, features have been defined as the operator reaching specific locations in the workspace, described as spheres, with one index finger. In addition to f_0 that marks the start of the activity, f_1 is associated with location 1 shown in Figure 3.15, f_2 with location 2, f_3 with location 3, f_4 with location 6, and f_5 with the output zone, which marks the end of the task.

Out of all variants that can be described with the defined features, we have considered six, which lead to the template tree in Figure 3.16, namely:

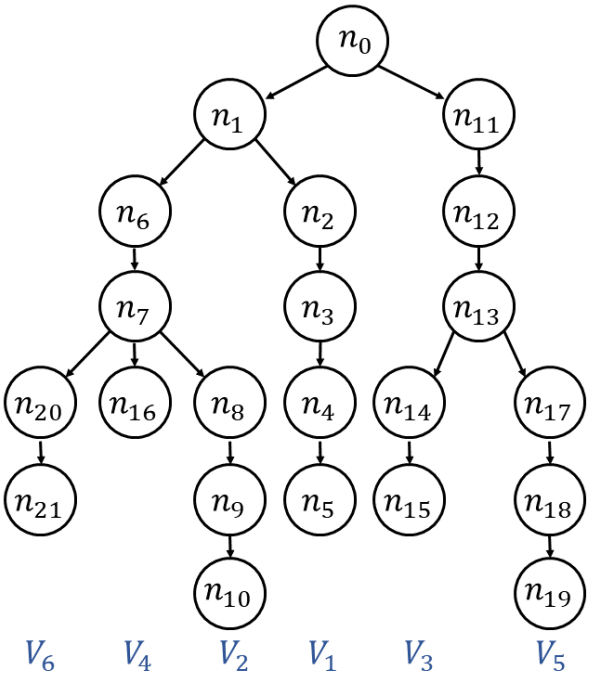
- $V_1 = (f_0, f_3, f_1, f_3, f_2, f_5)$;
- $V_2 = (f_0, f_3, f_3, f_2, f_4, f_1, f_5)$;
- $V_3 = (f_0, f_1, f_3, f_2, f_3, f_5)$;
- $V_4 = (f_0, f_3, f_3, f_2, f_5)$;
- $V_5 = (f_0, f_1, f_3, f_2, f_4, f_3, f_5)$;
- $V_6 = (f_0, f_3, f_3, f_2, f_1, f_5)$.

The variants differ in the assembly sequence used to complete the product. In V_1 the human mounts the left fork on the base, completes the caster fixing the wheel to the fork, mounts the right fork, and finally the right wheel. In V_3 operations are inverted: the human first assembles the left caster (fork plus wheel), fixes it to the base, then moves to the right caster which is mounted in the same way. During V_6 the operator fixes both forks, then assembles the wheels. Variants that describe error cases are also present. During V_2 and V_5, the operator finds a defective screw while fixing the right wheel. Thus, he/she has to stop and throw the screw in the waste zone before continuing. In V_4 the human forgets to mount the left wheel and delivers an incomplete product.

Variant recognition and template construction

The process of building the template from scratch has been repeated three times, assembling each time 15 products following a random sequence of variants. The first execution is saved as the initial template. Subsequent repetitions of the task are either monitored (if the variant is known) or added to the template (if unknown). Overall, the algorithm always recognises the correct variant as soon as the actions of the human differ from the known templates.

To exemplify its behaviour, we can focus on the three-way branch that stems from node n_7 in Figure 3.16, composed of variants V_2, V_4, and V_6, as it represents the most critical point for variant recognition. Let us consider the case when V_6 and V_4 are already known, i.e. are included in the template tree, and V_2 is performed for the first time by the operator. The template tree in such a situation is shown in Figure 3.17. The first segments (f_0, f_3), (f_3, f_3), (f_3, f_2) are common among the three variants. Thus, the algorithm is not aware of the new variant and the task is monitored as if it was V_4 or



(o_i, d_i)	n_i
(f_0, f_1)	n_{11}
(f_0, f_3)	n_1
(f_1, f_3)	n_3, n_{12}
(f_1, f_5)	n_{21}
(f_2, f_1)	n_{20}
(f_2, f_3)	n_{14}
(f_2, f_4)	n_8, n_{17}
(f_2, f_5)	n_5, n_{16}
(f_3, f_1)	n_2
(f_3, f_2)	n_4, n_7, n_{13}
(f_3, f_3)	n_6
(f_3, f_5)	n_{15}, n_{19}
(f_4, f_1)	n_9
(f_4, f_3)	n_{18}

Figure 3.16: *Template tree of the assembly task performed during experiments. Note that many nodes link to the same segment.*

V_6. After node n_7, a fork is already present: one instance of the DTW-based algorithm runs for each of the two branches and their probabilities are updated with each new sample according to the recursive Bayesian classifier of Section 3.3.2. At first, the algorithm cannot decide which is the correct variant, since the initial part of segments (f_2, f_1), (f_2, f_5) and (f_2, f_4) is common, as the operator mounts the first wheel. Then, either he/she fixes the second (V_6), or delivers an incomplete product (V_4), or finds a defective screw (V_2). As soon as the human movements differ from both known templates (≈ 8.5 s after the start of the current segment, see the top graph of Figure 3.17), the algorithm recognises the presence of a new variant: the costs returned by both DTWs increase and probabilities drop to zero. It can be also noticed that the advancement computed for the two wrong variants remains below 0.5 at the end of the segment, highlighting the fact that the new trajectories do not align well with the known ones.

Figure 3.18 shows a full example of one execution of V_4 when all variants are already known. V_4 represents an error variant that is rarely performed and lasts less than average. Initially, the expected duration is overestimated, as the prediction favours more usual variants, such as V_6. As

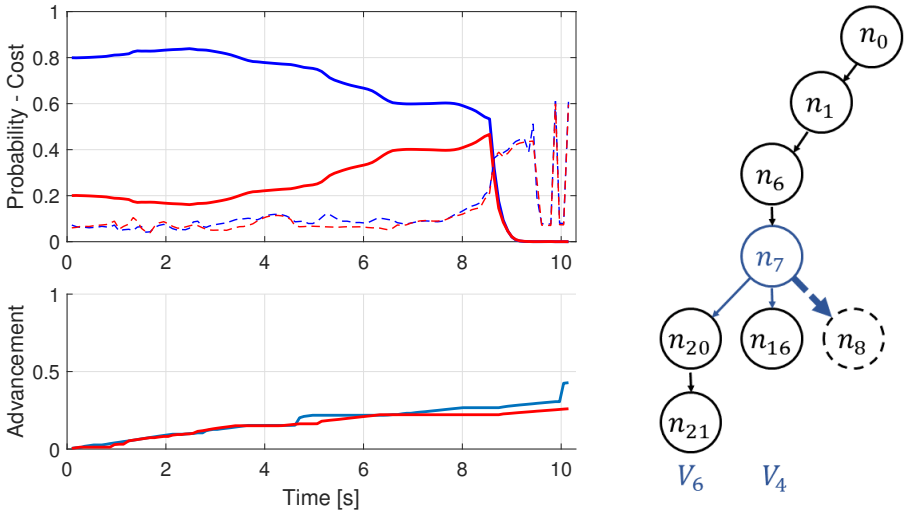


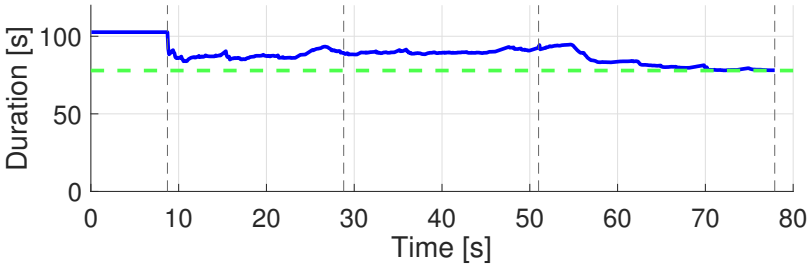
Figure 3.17: *Recognition of variant V_2, known V_4 and V_6. Both children of n_7 are deemed as wrong (≈ 8.5 s) and node n_8 is added to the tree. Graphs show P_i (top, solid), $\widetilde{D}_i$ (top, dashed) and adv_i (bottom) for segments (f_2, f_1) (red) and (f_2, f_5) (blue).*

information becomes available, the prediction converges to the actual task duration. Notice that the correct branch is recognised even if it starts with the lowest prior probability.

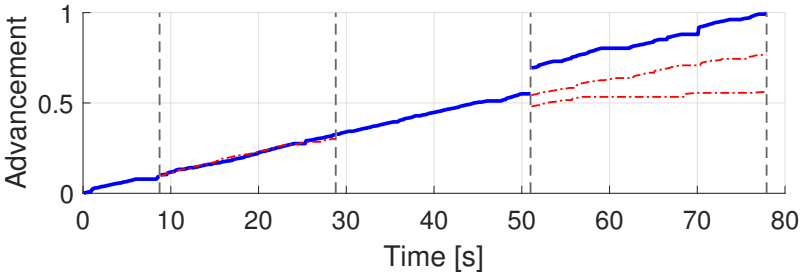
Performance evaluation

To evaluate performance, the proposed *Multi-variant* (MV) algorithm is compared to other two methods. The first one is the *Single-Variant* (SV) algorithm presented in Section 3.2, which describes the prototypical execution of the task with a single template sequence. The second is an ideal method, referred to as the *Prophet algorithm* (PA), which has perfect knowledge of the variant being performed. It behaves as an SV algorithm trained only on past executions of the variant to monitor.

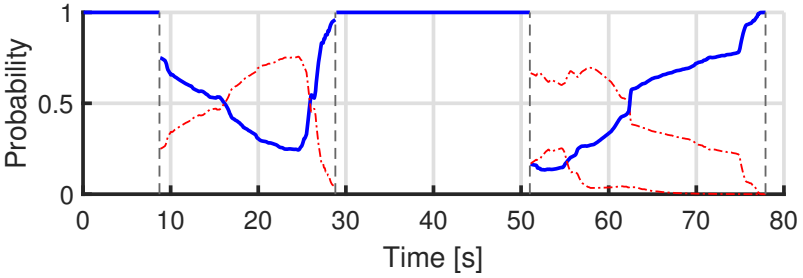
Results come from 33 repetitions of the assembly task. Respectively, the six variants have been repeated 12, 3, 6, 2, 3, and 7 times, reflecting the fact that error ones are infrequent in a real case. To neglect the learning transient, data from previous experiments have been used as a training set to build the template. The average duration of the task has been 105.0 s, from a minimum of 72.9 s to a maximum of 131.0 s, thus showing high variability. The performance index is the average error in the prediction of



(a) *Expected (blue) and actual (green) duration.*



(b) *adv of correct (blue) and wrong (red) segments.*



(c) P_i *of correct (blue) and wrong (red) segments.*

Figure 3.18: *Results for one execution of* V_4*: as the correct variant is recognised, the duration estimate converges to the real one.*

e_m [%]	Total			Last 20 s		
	q_{25}	q_{50}	q_{75}	q_{25}	q_{50}	q_{75}
SV	5.22	17.99	30.57	3.13	11.35	22.35
MV	1.95	4.28	8.79	0.63	1.62	3.52
PA	1.93	3.98	7.84	0.97	2.46	5.49
MV–PA	-2.65	0.30	2.93	-3.66	-0.52	0.55

Table 3.1: *25, 50, and 75 percentiles of the average prediction errors computed along the entire duration of the activity (1^{st} column) and the last 20s only (2^{nd} column).*

the duration, normalised over the actual duration of the task T^*:

$$e_m = \frac{1}{T^*} \int_0^{T^*} \frac{|\hat{T}(\tau) - T^*|}{T^*} d\tau$$

Figure 3.19 shows examples of the results obtained by the three methods following different task variants. MV consistently gives more accurate predictions than SV, while PA attains the best results. SV fails, as the only template sequence it considers is compared to the input regardless of the variant being performed. Since the algorithm takes the shortest past execution of the activity as the reference, this is likely to be an instance of V_4, i.e. when an incomplete product is delivered. Table 3.1 reports the errors obtained for the different methods. While SV attains a median error (q_{50}) of 17.99% of the task duration, MV gives accurate predictions, with a median error of 4.28%, which means less than 5 s for the average activity. Furthermore, the prediction becomes more precise as the task progresses, and the error reduces to 1.62% if we consider only the last 20 s of each execution. The error dispersion is also significantly reduced, with the interquartile range that halves from 6.84% to 2.89%. The improvement mainly comes from the fact that the availability of more data reduces the uncertainty of unknown quantities, such as the rate of advancement of the task (from which the expected duration is extrapolated) or the estimate of L_i^{post} used in equation (3.12). Also, L^{pre}, whose value is certain, grows with the number of completed segments, while L_i^{post} lowers and becomes less important. This kind of uncertainty may produce peaks or valleys in the duration estimate in the first part of the task, similar to the one shown in Figure 3.19a, even if the advancement is correctly tracked (see Figure 3.19b).

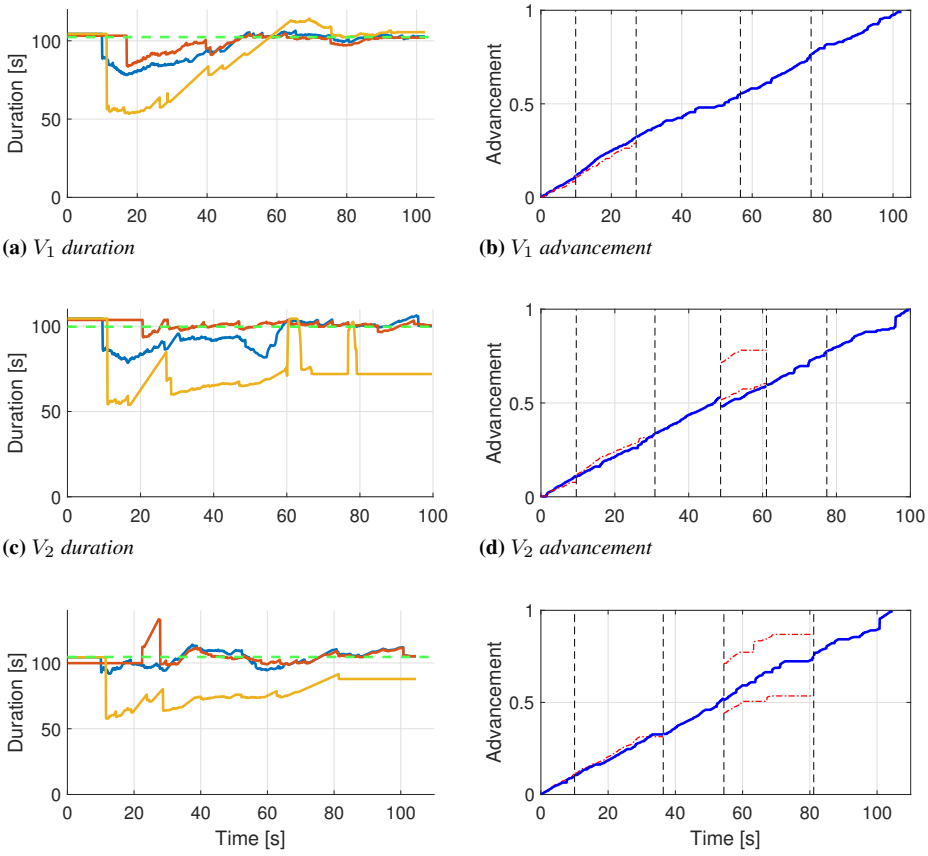
When an infrequent variant that lasts more (less) than average is performed by the operator, the algorithm overestimates (underestimates) the length of the activity until the correct variant is detected. Figure 3.19c reports the results of one execution of V_2, which is an infrequent variant and

lasts longer than average. The presence of the more probable variant V_6 in the same part of the template tree leads to an underestimation of the duration of the ongoing task. When the operator performs segment (f_2, f_5), the probability of the correct variant raises and the prediction converges to the actual operation duration. This is apparent from the negative jump in the advancement associated with the correct variant shown in Figure 3.19d.

From the last row of Table 3.1, we can also notice that the performance of MV and PA are not significantly different, with the median of the error difference in the range of $\pm 0.5\%$. This is a major result, which means that the presence of several variants of the task does not worsen the estimate of its duration. An example is shown in Figure 3.19e for one instance of V_6. Despite the monitoring of the task requires the evaluation of three forks in the template, one of which composed of three branches, the predictions of MV and PA can be mostly superimposed. Note that the slightly better performance attained by MV in the last 20 s may be due to the more data it had available for template learning.

Concluding remarks

Overall, the proposed approach can monitor a wide variety of tasks and task variants while achieving good performance. Its main limitation lies in the requirement of a priori knowledge of the set of segmentation features, which limits the number of task segment types that can be handled by the algorithm. Future works should investigate how to improve task segmentation through the automatic definition of features from the operator's motion data. This would also allow optimizing segmentation to minimize the similarity among different segments, possibly facilitating variant recognition.



(a) V_1 *duration*

(b) V_1 *advancement*

(c) V_2 *duration*

(d) V_2 *advancement*

(e) V_6 *duration*

(f) V_6 *advancement*

Figure 3.19: *Examples of results in the monitoring of different variants of the task. In the left column: prediction of $\widehat{T}(T_e)$ obtained with MV (blue), PA (red) and SV (yellow) algorithms. Actual duration in green. In the right column: advancement obtained with MV algorithm (blue), with the advancement of wrong branches reported in red.*

Part II

Dynamic scheduling of flexible collaborative cells

CHAPTER 4

Control system architecture

THE main objective of this thesis is to control the production of collaborative manufacturing cells, focusing on flexible assembly tasks where multiple humans and robots cooperate to complete different products according to a time-varying mix. In Part I, we addressed the monitoring of the human activity in order to gather more information on the current state of the process. In Part II, we deal with the task planning problem, which means to assign operations to all the agents that work in the cell in order to satisfy the product demand. Scheduling algorithms are essential elements to manage the production of industrial plants since the usage of the limited available resources must be carefully planned to maximise productivity and minimize costs. However, optimal task allocation and scheduling are usually made difficult by the large size of the problem, the presence of many constraints, and large uncertainty in the system. For these reasons, a lot of research effort has been put on the topic for many years both in manufacturing and other fields, such as electricity grids [22], traffic management [48], and CPU scheduling [162].

In our application, the presence of the human adds flexibility to the system thanks to his/her strong adaptability and high cognitive skills. On

the other hand, it also introduces large uncertainty, since the workers' behaviour is only weakly controllable. As already discussed in Part I, the human can, in general, exhibit an erratic behaviour that is difficult to model and predict. Moreover, present-day flexible manufacturing is characterised by frequent changes in production. As a consequence, the layout of the workspace, the types of products, the agents available in the plant, and the operations they must perform can all be subject to modifications. For instance, robots and workers can be reallocated to different cells, new products introduced, and others discontinued. In this scenario, it is crucial to implement control architectures that are capable of quick adaptation and reconfiguration to minimise set-up times. Also, dynamic scheduling algorithms must fully exploit the available flexibility and output plans that are robust against the system variability. To do so, both task allocation and scheduling must be solved online and concurrently.

One way to attain a certain degree of performance and robustness is to apply dispatching rules [39], which are based on predetermined heuristics. The main advantage is the minimal computational load at run-time that favours their use for very large problems, such as job-shop scheduling with many workstations. However, decisions are based only on local information and cannot lead to optimality. Hence, the need for more complex dynamic schedulers that must work in real-time. To reduce computational load, decentralised auction-based approaches are presented in [23, 113]. Also, [29] proposes to decompose a global task into local subtasks that are assigned to individual agents according to their capabilities. Instead, Lou et al. [93] describe a two-stage algorithm where the initial schedule, computed offline, is repaired at run-time using heuristics in case it becomes unfeasible. Centralized strategies are also proposed. For instance, in [49] the authors present a framework for task scheduling in multi-robot teams that combines Mixed Integer Linear Programming (MILP) optimization and Earliest Deadline First policy. The proposed algorithm is not optimal, but only search for a feasible solution. Similarly, [150] combines MILP optimization with heuristic rules for job-shop order scheduling. In [112], the best task allocation is found by evaluating possible alternatives offline. The plan is then adjusted at run-time in case of abnormal events following predetermined rules. Finally, [59] analyses two iterative heuristic algorithms for cycle time optimization of deterministic marked graphs, which are able to model cyclic scheduling problems in manufacturing plants.

Restricting the scope to human-robot collaboration, [21] presents a strategy based on genetic algorithms that minimises production time and cost. Johannsmeier et al. [68] propose a hierarchical framework that leverages

A* search on AND/OR graphs for optimal task planning in human-robot assembly. Although task assignment and sequencing are solved offline, the method includes a certain degree of real-time adaptation at the motion planning level. In [19, 73], Petri Net models are exploited to schedule robot tasks. In particular, [19] solves the problem by predicting human intentions, which are assumed to be uncontrollable. [137] considers the human intention to be partially observable and proposes a framework for task allocation based on Partially Observable Markov Decision Processes. In [165], the robot schedule switches between precomputed alternatives to adapt to the preferences of the human operator, while in [128] task allocation is guided by a model of bilateral trust dynamics between the human and the robot partner. Instead, systematic ways to optimally assign tasks based on agents' skills are presented in [11, 130].

In the following, the control system architecture proposed in this work for the dynamic scheduling of collaborative tasks is detailed in Section 4.1. Then, Section 4.2 presents a general method to build the Digital Twin of flexible manufacturing cells, which constitutes the pivotal element of the proposed framework. Finally, Section 4.3 focuses on how to include robot faults and human errors in the model.

4.1 Proposed control system architecture

In recent years, cyber-physical systems are becoming key components of the emerging smart factories of Industry 4.0 [85, 111]. A virtual copy of the physical system, called Digital Twin (DT), collects data in real-time and tracks the state of the process at a high level of detail [138]. On the one hand, information is locally available to make decentralized decisions and achieve a high degree of autonomy. On the other hand, connection with the enterprise information system allows for better production management and tighter integration with other processes, such as logistics and maintenance.

As far as task planning is considered, the presence of a DT provides a tool to simulate feasible system evolutions, explore what-if scenarios, and predict the performance of alternative production plans. Obviously, the quality of the information provided by the DT depends on how well the virtual copy models the real system. In particular, critical issues are given by the inherent variability of manufacturing systems that stems, for instance, from stochastic processing times, the occurrence of unpredictable faults, and uncertainty in the future production demand. Modelling the manufacturing process becomes even more difficult when the human is also present,

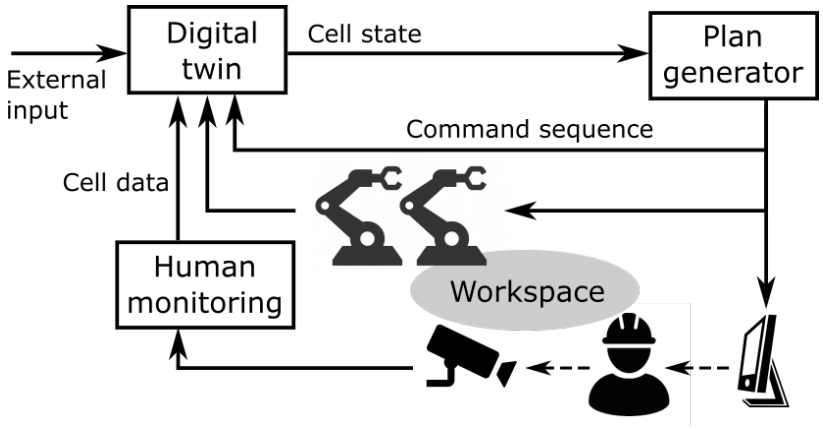


Figure 4.1: *Control system architecture.*

as in human-robot collaborative tasks. As already highlighted, the human is a major source of uncertainty. Even in case that they are provided with instructions on the next operation to perform, they can make mistakes, accomplish the same task in several ways, or refuse to follow the instructions altogether. For this reason, although the literature available on applications of DTs to manufacturing is abundant and diverse [77], limited contributions consider human–robot interaction [9, 99].

The control system architecture proposed in this work is depicted in Figure 4.1. The physical system consists of multiple robots and humans that collaborate in a shared workspace to the assembly of different products. The pivotal element is constituted by the Digital Twin of the collaborative cell, which models both the physical structure of the workspace and the evolution of the assembly process. For instance, it describes the production demand, the state of agents and other resources, the material flow, the occurrence of faults, and the associated recovery actions. A detailed description of the implementation of the DT, which is based on Petri Nets, is provided in Section 4.2.

The current state of the process is tracked in real-time based on data collected from the physical system and the enterprise information system. In particular, information on the target production mix comes from the company's ERP system, while robots provide data on their status through wired or wireless communication between the controllers and the DT. More critically, a dedicated *monitoring unit* is needed to gather information regarding the activity of human workers. The ideal monitoring unit should be able to identify the operation that the operator is performing, provide an estimate of the expected duration of the task, and notify the system of the occur-

rence of errors. For this purpose, the algorithms presented in Part I can be exploited in combination with a task identification algorithm. Anyway, simpler monitoring strategies can also be employed, at the price of reduced performance. For instance, a simpler prediction of task duration would not cause any problems in the functioning of the system, but would achieve suboptimal agents' coordination. Instead, in case of information unavailability, such as the lack of error detection algorithms, the system would still work as intended as long as the DT is able to correctly track the state of the system with the available data. Instead, manual intervention would be required to recover when the state of the DT and the one of the physical system diverge, leading to increased downtime and reduced flexibility.

The current state of the process, as given by the DT, is taken as the initial condition by the scheduling algorithm to predict the future evolution of the system and determine the optimal control action with a receding horizon approach. This is done by exploring the Reachability Tree of the Petri Net model, to which a temporal characterization has been applied. When the plan of future operations that the agents must perform has been determined, only the first command is dispatched. Then, the plan is re-scheduled starting from the updated status of the system. Both task allocation and sequencing are dynamically solved on-line for multiple concurrent products to increase productivity and flexibility. Also, many sequences are possible to complete the assembly, and the optimal one for each product is selected in real-time by the scheduler, whose choice can change during operation. This allows adapting the plan in real-time, responding on-line to the occurrence of robot faults, and taking into account how quickly the human operator completes a task. The proposed dynamic scheduling algorithm is thoroughly presented in Chapter 5.

Commands for the robots are sent directly to the robot controllers. Instead, human workers are informed of the next operation to perform through intuitive user interfaces. Usually, graphical interfaces on grounded screens are used to this aim, although other strategies have been already employed in the literature [159]. Advanced human interfaces can improve performance in complex scenarios. In this view, in Section 5.2 we propose to combine graphical instructions with wearable vibrotactile interfaces.

As a final remark, we recall that the short product lifecycles and the large number of product variants that characterise present-day flexible manufacturing require frequent re-designs of workstations. For this reason, various benefits come from the possibility to analyse new production concepts prior to implement them on the real plant. Appendix A discusses a way to integrate the proposed framework into a commercial Product Lifecy-

cle Management software for automatic manufacturing process verification and commissioning, which allow companies to reduce costs and improve productivity. Notably, the variability introduced in the system by the presence of the human and the occurrence of faults is taken into account.

4.2 Digital twin for flexible collaborative cells

As stated in Section 4.1, the DT aims to track the state of the assembly process in real-time and allow simulating future evolutions of the system. To do so, it must describe both the physical structure of the workspace and the assembly tasks. Specifically, the DT models the state of all the agents that are present in the cell (humans and robots), the operations they can perform, and the tools and other resources that are needed for their execution. Precedence constraints among operations and the material flow inside the workspace stem from the structure of the assembly products. Moreover, the presence and finite capacity of buffers that store intermediate Work In Process (WIP) are also taken into account, as they influence scheduling. Finally, the occurrence of robot faults and human errors, and the associated recovery actions, are modelled to increase robustness.

In the literature, several methods have been employed to model assembly sequences. In [145], the authors review existing formalisms and propose a novel approach derived from SFC diagrams. Instead, AND/OR trees are used in [30], while [134] exploits precedence graphs. The aforementioned works focus on the representation of the assembly structure, while neglecting other aspects of the assembly process. Casalino et al. [19] recognise Petri Net as a powerful and general tool to describe human-robot assembly thoroughly. However, human actions are not included in the model, as well as faults and errors.

In this thesis, the Digital Twin of the cell is described in terms of Petri Nets, as they are capable of modelling all the information required to obtain a comprehensive DT. Moreover, they provide an efficient framework for simulation and scheduling purposes. However, as the complexity of the manufacturing process increases, the abundance of details embedded in the DT obfuscates the high-level structure of the process and leads to a rather complex and unintuitive model. Therefore, a simpler and more user-friendly way to provide data to the DT is needed, which can be easily understood by process experts. Here, we propose to automatically generate the Petri Net-based DT from the definition of assembly products via Augmented AND/OR Graphs (AAOG). Readability, modularity, and automatic generation are even more crucial features when flexible manufacturing is

concerned. Indeed, frequent changes in the production reflect in equal modifications of the DT, which must therefore be fast and straightforward.

In the following, Section 4.2.1 presents the formal definition of the high-level assembly jobs and the rules to reconfigure the assembly process in the face of changes. Then, the Petri Net-based digital twin is described in Section 4.2.2 together with the procedure for the automatic generation from AAOGs.

4.2.1 High-level job definition

This Section aims to provide a user-friendly and easily modifiable way to define the assembly jobs that compose the complete manufacturing process. A user that is knowledgeable in the task, but is not a robotics/IT expert should be able to specify even complex use-cases from a high-level perspective. Then, the software is in charge of the automatic generation of the DT. For the scope of this thesis, a collaborative robotic cell consists of a set of resources R used to repeatedly complete a set of jobs. Resources can be working agents (i.e. humans and robots), shared workspace, tools, or other equipment. A job is a finite set of tasks (or operations) $\Omega = \{o_1, \ldots, o_{n_o}\}$ that are required to assemble a product $P = \{p_1, \ldots, p_{n_p}\}$ composed of n_p parts. In general, a given product can be completed following different sequences of tasks. To give a compact representation of all viable assembly plans, one can use AND/OR trees [30]. Starting from AND/OR trees, we propose AAOGs, which are expanded to include information on the feasible assembly operations, resource requirements, and workspace layout.

An AAOG is a hypergraph $H = (N, E)$, where N is the set of nodes and E the set of (hyper)edges. Each node in the graph represents a step in the assembly process. It is uniquely associated with an intermediate assembly (WIP part) and the physical buffer where it is stored between operations. A node is thus described by $n = (W, c_B)$, where $W \subseteq P$ defines the WIP part and c_B the buffer capacity, i.e. the maximum number of products that can contain simultaneously. Starting with all parts $p_i \in P$ disconnected from each other, we join them to form the final assembled product P. Thus, leaves of the tree are n_p nodes with $W_i = p_i$ for $i = 1, ..., n_p$, while the root node has $W_{|N|} = P$.

The graph has an edge for each task. A task is described by $o = (d, R_O, R_F)$, where d contains information about its expected duration and R_X stands for a set (possibly empty) of resources. Namely, R_O collects resources that must be available at the beginning of the task and are needed for its execution, whereas R_F collects resources that are released at the end

of it. The two sets can be different, so to account for cases where a resource is kept occupied along multiple tasks and is not released at the end of each one. An assembly operation connects a single parent node $f \in N$ to a set of children nodes $C \subset N$. Each edge is thus defined by $e = (o, C, f)$. For the AAOG to be well-defined, each edge must be such that:

$$W_f = \bigcup_{c \in C} W_c \quad \wedge \quad W_i \cap W_j = \emptyset, \, \forall i \neq j \in C$$

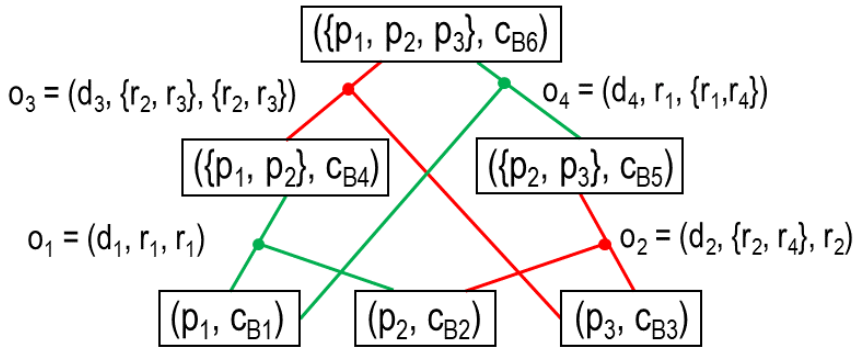
That is, the set describing the WIP part associated with the parent node is the union of disjoint sets of all WIP parts coming from children nodes. Non-assembly tasks, such as painting or greasing, can be modelled by connecting a single child to a parent node, so that both WIP are made of the same parts. Yet, this allows considering resource requirements, task duration, and the possible relocation of the WIP product inside the workspace.

A feasible assembly sequence for the job Ω is the set $\Omega_S \subseteq \Omega$ of tasks related to the edges of a minimal sub-graph that includes the root node and all the leaves of the complete AAOG, i.e. a graph describing a set of operations sufficient to obtain the complete product from its base parts. The job is considered to be completed after the execution of all and only the tasks belonging to one of its feasible assembly sequences. Usually, the optimal Ω_S is determined offline based on nominal behaviour. In this work, the scheduler dynamically makes this choice for each product to improve flexibility and adaptation to the current status of the production.

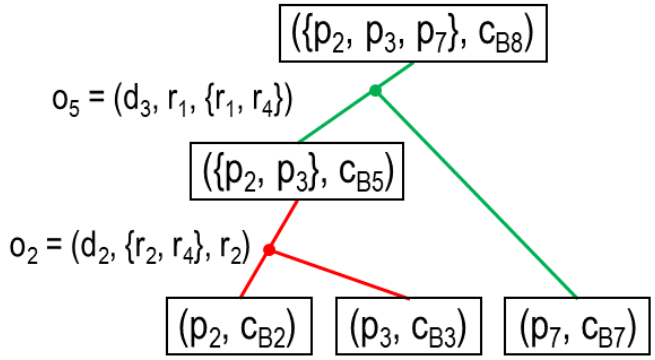
Figure 4.2a shows an explanatory example of AAOG for a product composed of three parts $P = \{p_1, p_2, p_3\}$, which are specified in the leaves with the relative buffer capacities. The root represents the complete product $W_6 = \{p_1, p_2, p_3\}$, and the intermediate nodes contain the WIP parts $W_4 = \{p_1, p_2\}$ and $W_5 = \{p_2, p_3\}$. The resource set is $R = \{r_1, r_2, r_3, r_4\}$ and we can suppose r_1 to be a human operator, r_2 a robot, r_3 a tool, and r_4 a fixture device. Two assembly sequences are possible, namely (o_1, o_3) and (o_2, o_4). The human can perform operations o_1 and o_4, while the robot o_2 and o_3. In addition, operation o_3 can be completed by the robot only with the help of the tool r_3. A fixture r_4 is required to hold the WIP product throughout the entire assembly if the sequence (o_2, o_4) is followed. Therefore, r_4 is occupied at the beginning of o_2 and released only after the completion of o_4.

High-level model of the complete assembly process

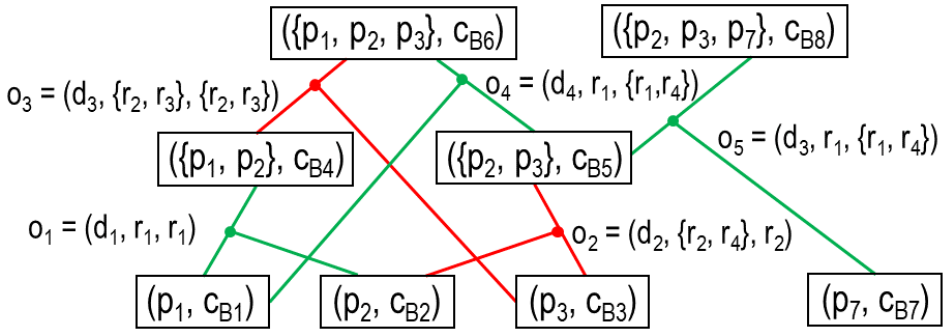
When multiple product types are assembled in the same collaborative cell, the expert user defines one AAOG per product. Then, all graphs must be



(a)



(b)



(c)

Figure 4.2: *Examples of Augmented AND/OR Graphs for two different assembly jobs (a)-(b) and Augmented AND/OR Graph of the complete assembly process obtained by merging those related to the single products (c).*

merged together to obtain the AAOG that describes the complete assembly process. In doing so, one must pay attention to the fact that different jobs may share part of their assembly sequences. This is the case, for instance, when the manufacturing unit is devoted to the assembly of variants of the same product. On the one hand, AAOGs of single jobs shows all possible assembly sequences for the same product. On the other hand, merging the individual AAOGs highlights the interrelations that exist among different products. Both pieces of information are transferred to the final DT model to allows for higher flexibility in the scheduling stage. In fact, the scheduler dynamically chooses the assembly sequence for each product and dynamically associates intermediate WIPs to the type of final product.

Suppose, without loss of generality, to have two AAOGs, $H_1 = (N_1, E_1)$ and $H_2 = (N_2, E_2)$, and let $H = (N, E)$ be the graph obtained by merging H_1 and H_2. Then, H is obtained as the union of the two AAOGs, that is $N = N_1 \cup N_2$ and $E = E_1 \cup E_2$, considering that the following rules applies to determine when elements belonging to different AAOGs are equal:

- Two edges are equal if they describe the same operation:

$$e_i = e_j \iff C_i = C_j \wedge o_i = o_j \qquad \forall e_i \in E_1, e_j \in E_2$$

- Two nodes are equal if they describe the same intermediate product:

$$n_i = n_j \Leftrightarrow W_i = W_j \wedge \exists e_k \in E_1, e_l \in E_2 : e_k = e_l \wedge n_i = f_k \wedge n_j = f_l$$
$$\forall n_i \in N_1, \forall n_j \in N_2$$

For leaf nodes, the rule reduces to check whether the nodes refer to the same base part:

$$n_i = n_j \iff W_i = W_j$$

An example of the procedure is given in Figure 4.2, where the graphs in Figure 4.2a and 4.2b are merged to obtain the AAOG in Figure 4.2c. In particular, the sub-assembly composed of parts p_2 and p_3 is common to both products and it is not replicated in the final AAOG. This situation implies that, during production, a WIP $W_5 = \{p_2, p_3\}$ can be either completed as a product $W_6 = \{p_1, p_2, p_3\}$ or as $W_8 = \{p_2, p_3, p_7\}$. The optimal choice is made online by the scheduling algorithm according to the current process state and product demand.

Reconfiguration of the assembly process

The high-level description based on AAOGs is used to automatically generate the DT of the assembly process, as described in Section 4.2.2. Furthermore, it allows for simple and fast adaptation in the face of changes, which

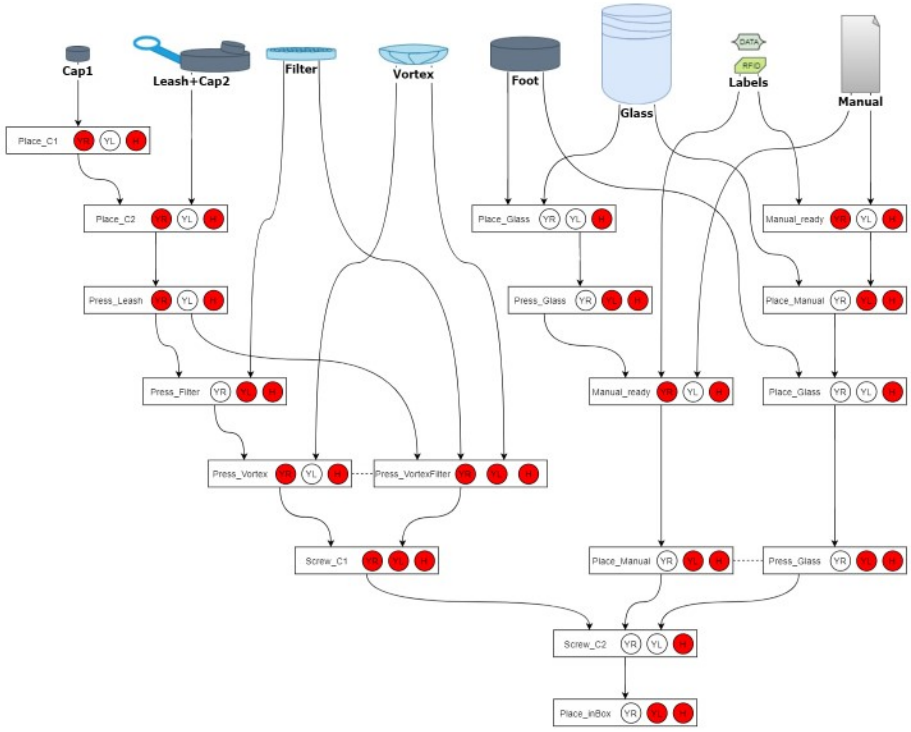


Figure 4.3: *Example of intuitive graphical interface for the definition of Augmented AND/OR Graphs.*

is a crucial feature for flexible manufacturing applications. Typical modifications to the process can concern the production mix, the agents working in the cell, or the workspace layout.

If the process to assemble an existing product type is modified, it is sufficient to act on the AAOG related to the specific type of product accordingly. To introduce a new type of product, one must simply define the associated AAOG. Conversely, if a product is discontinued, the corresponding AAOG is removed. Then, such modifications reflect into the AAOG that describes the complete process. Instead, the addition or removal of an agent within the cell can be managed directly from the complete AAOG. Specifically, if a new agent is added, the user can define the new operations on the complete AAOG, which in turn are added to the product AAOGs as appropriate. Vice versa, if an agent is removed, all operations it performs are no longer available and are removed from all AAOGs where they appear. Finally, alterations of the physical workspace can be, for instance, permanent changes in the available equipment (tools, fixtures, etc.), which are handled

similarly to agents. Also, modifications to the buffer capacity imply new values of the parameter c_B of corresponding nodes. All the aforementioned functionalities can be coded into a software application with an intuitive graphical interface, like the one shown in Figure 4.3.

4.2.2 Digital twin model

The AAOG allows us to define the assembly process in a simple, yet rigorous, way that is suitable also for non-expert users. However, it consists in a static representation that lacks the concepts of state and time. Thus, it cannot be used to track the evolution of the robotic cell and the production. To this purpose, it is possible to automatically build a partially controllable Petri Net (PN) from the AAOG, which can be seen as the DT of the complete collaborative cell. The PN describes both the physical structure of the workspace and the evolution of the assembly process, considering the state of all resources and concurrent WIP products, as well as situations originating from human errors and robot failures.

A PN is a bipartite graph defined by a tuple $\mathcal{M} = (\Pi, \Theta, I, \boldsymbol{m}_0)$, where Π is the set of places, Θ the set of transitions, I the $|\Pi| \times |\Theta|$ incidence matrix and $\boldsymbol{m}_0$ the $|\Pi|$-dimensional initial marking vector. The marking vector indicates the number of tokens that are present inside each place of the PN and gives a compact representation of the state of the model. The evolution of the state of the cell is tracked in real-time through subsequent transition firings that represent the occurrence of events. Given the $|\Theta|$-dimensional transition vector $\boldsymbol{\theta}_k$ having in the j-th position the number of times transition j has fired, the updated marking vector is computed as:

$$\boldsymbol{m}_{k+1} = \boldsymbol{m}_k + I\boldsymbol{\theta}_k \tag{4.1}$$

Transitions can fire only if enabled, i.e. their firing leads to a feasible state of the system such that $\boldsymbol{m}_{k+1} \geq 0$.

The set Θ is partitioned as $\Theta = \Theta_C \cup \Theta_U$, $\Theta_C \cap \Theta_U = \emptyset$, with Θ_C the set of controllable transitions, whose firing is decided by the scheduler, and Θ_U that of uncontrollable transitions that are triggered by exogenous events. In this work, controllable transitions determine the start of new tasks, whereas uncontrollable ones relate to tasks completion and the occurrence of faults. In the following, uncontrollable transitions are labelled with an overbar.

Automatic generation from Augmented AND/OR Graphs

AAOGs define four main elements that must be translated into their PN equivalent, namely resources, buffers, operations, and the connections be-

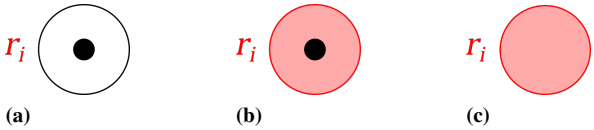


Figure 4.4: *Resources are modelled in the Petri Net as a single place. Free generic resource (a), free agent (b), and busy/unavailable agent (c).*

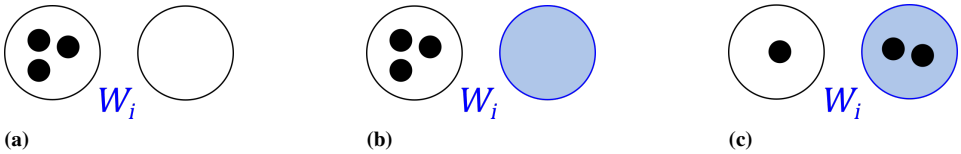


Figure 4.5: *Buffers are modelled in the Petri Net with a pair of places. Example of free buffer for base parts (a) and WIP (b), and partially full WIP buffer (c).*

tween these components. In the PN, a resource is modelled by a place that is marked only when the resource is available (see Figure 4.4). Differently from those that describe generic resources, such as tools, places related to humans and robots are highlighted with a coloured background in the graphical representation. If marked, they contribute to the cost in determining the optimal schedule, as they indicate an idle agent. More details on the scheduling algorithm and the cost function are given in Chapter 5.

Each node of the AAOG is transformed into a pair of places to track free and occupied slots of the physical buffer that stores the corresponding intermediate WIP. The sum of the markings is equal to the buffer capacity. Figure 4.5a and Figure 4.5b show an example of free buffers that can contain three products. Instead, in Figure 4.5c the buffer is partially full, with two occupied spaces and one more available. Similarly to what happens for agents, places that mark the presence of intermediate WIP, i.e. those associated with nodes of the original AAOG that are neither leaves nor the root, are coloured to highlight the fact that they influence the scheduling cost when marked. Specifically, they indicate a WIP product which is stored and not being processed.

The basic structure of operations is composed of a controllable transition that models the start of the task, a place that is marked as long as the operation is in progress, and an uncontrollable transition that triggers upon task completion (see Figure 4.6). However, operations require resources to be performed and assemble parts together to obtain new WIP or final products. Therefore, elements of the PN are connected to obtain the complete

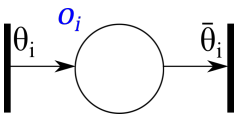


Figure 4.6: *Operations are modelled in the Petri Net with a sequence of controllable transition θ_i, place, and uncontrollable transition $\bar{\theta}_i$.*

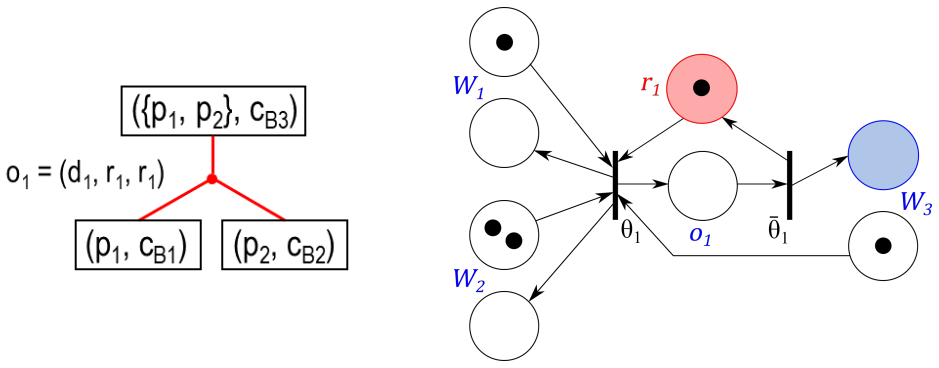


Figure 4.7: *Equivalent model of an operation with parts and resource requirements using AAOG (a) and Petri Net (b).*

description of the task. Figure 4.7 shows one operation with parts and resource dependencies described as an AAOG and the equivalent PN model. The task is performed by one agent and consists of the assembly of two base parts into an intermediate WIP ready for further processing.

The complete PN is obtained by interconnecting operations according to the structure of the assembly process defined by the AAOG. A simple example is provided in Figure 4.8, which shows the equivalent PN obtained from the AAOG in Figure 4.2b. The job comprises two operations (o_2 and o_5) performed by two agents (r_1 and r_2, with red background). During the first operation, performed by r_2, two base parts W_2 and W_3 are joined to obtain W_5. Then, r_1 complete the product adding part W_7 to the intermediate WIP. Notice that an additional resource r_4, which can be a fixture, is needed along the complete assembly and is not freed at the end of operation o_2. For each buffer W_i, tokens in the upper place indicate available parts, tokens in the lower place indicate the number of free slots, and their sum is equal to the buffer capacity (e.g. W_2 has $c_{B2} = 2$).

Based on the initial marking (Figure 4.8a), only the controllable transitions θ_2 is enabled: at least one part is available for each input buffer, and the required resources are free (marked places). After that θ_2 fires (Fig-

ure 4.8b), a marked place indicates that the operation o_2 is ongoing. Also, one slot of the input buffers is freed and resources r_2 and r_4 become busy. The uncontrollable transition $\bar{\theta_2}$ fires upon task completion (Figure 4.8c): the operation place empties, the agent r_2 returns available and the output buffer contains the W_3 part. At this point, transition θ_5 is enabled and operation o_5 can start. Instead, θ_2 cannot fire although both the base parts and the agents are available, since r_4 is still occupied.

As a final remark, PNs are a general and powerful tool that can describe additional cases that are not explicitly considered in this work, but can be inserted with low effort in the formalism. For instance, one can think of including a transport delay for buffers (e.g. to model conveyor belts) or the presence of mobile robots. Furthermore, Section 4.3 details how to model a comprehensive variety of robot faults, human errors, and associated recovery actions in the DT.

4.3 Management of robot faults and human errors

A DT that only describes the nominal operating conditions limits the productivity of the plant. Whenever an abnormal event occurs, the DT would not be able to properly track the state of the system, production would stop, and manual intervention would be required for recovery. The presence of the human increases the potential sources of errors, but also enhances the recovery capabilities of the system thanks to his/her cognitive skills. To allow the scheduling algorithm to optimally handle unexpected events and continue production uninterruptedly, the DT must be modified to take into account the different types of failure cases and the appropriate recovery actions. Specifically, abnormal situations can arise from robot faults, human errors, defective products, or problems with the equipment. On the one hand, hardware faults are usually infrequent, but can grow with equipment wear. On the other hand, human errors depend on several factors, such as attention, fatigue, and cognitive load.

The complexity of the collaborative manufacturing process makes it impossible to predefine all possible causes and effects of failure conditions. Nevertheless, there is no need to explicitly model every detail. First of all, the control system architecture is designed to inherently adapt to variations in the duration of operations. Therefore, we can avoid including in the DT all the small accidents whose only effect is to delay task completion. Instead, the monitoring unit is in charge of estimating the expected duration of the ongoing activity taking them into account. For instance, the algorithm presented in Chapter 3 was able to correct the duration of the human

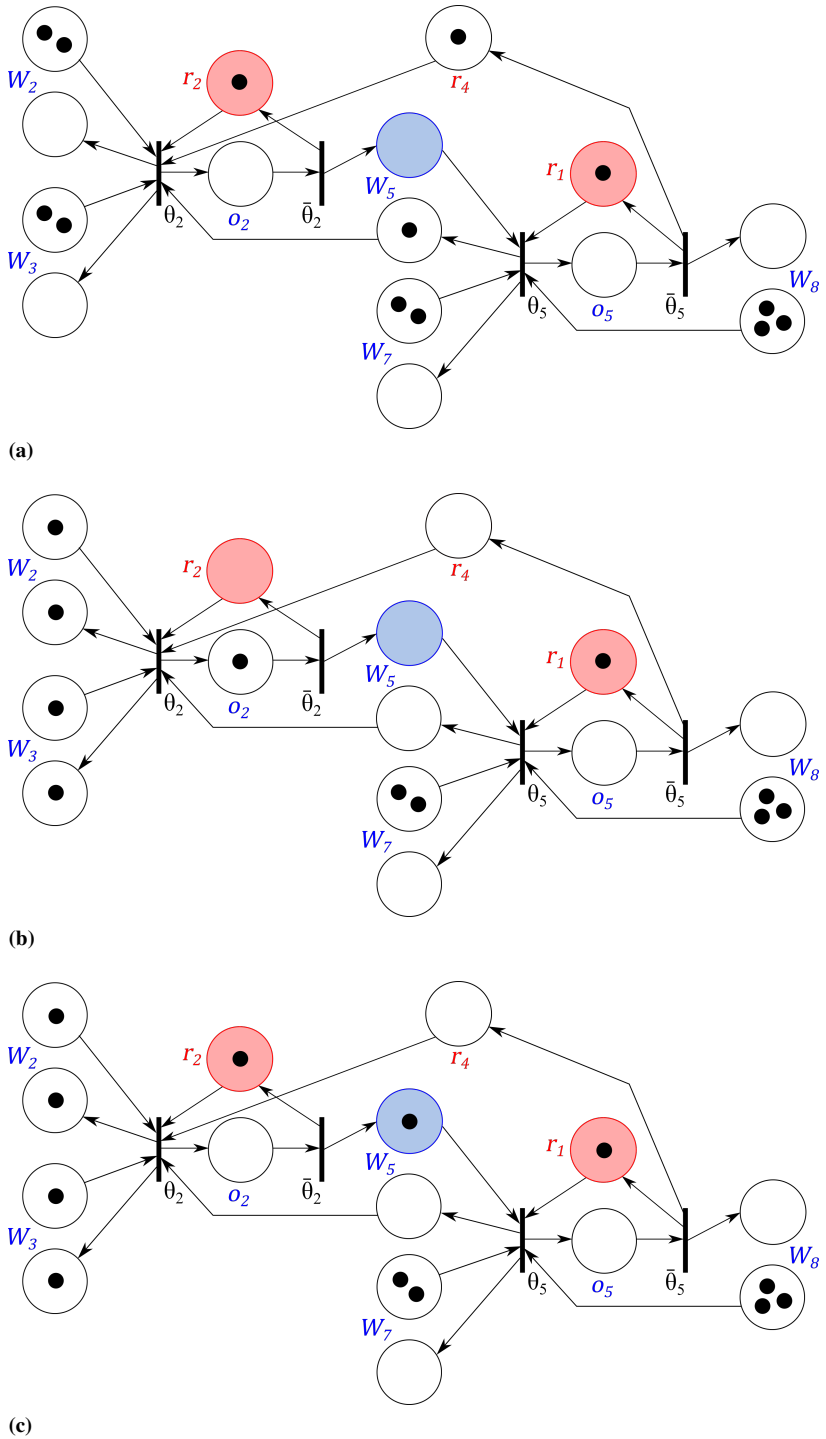


Figure 4.8: *Petri Net equivalent of the Augmented AND/OR Graph in Figure 4.2b and example of evolution. Initial state (a), state after θ_2 fires (b) and state after $\bar{\theta}_2$ fires (c).*

activity in the case a defective part was present and had to be substituted. On the other hand, a robot fault produces two effects: it makes the agent unavailable and requires human intervention for recovery. The occurrence of such fault has a significant impact on the state of the cell and future task scheduling. Therefore, the PN state must be modified accordingly.

Overall, the guiding principle in the definition of abnormal behaviours was to balance the search for completeness in the description of possible failure cases and the need to limit the resulting increasing complexity. Thus, similar faults and recovery actions are grouped together and we leave a certain degree of freedom to the human to choose how to better respond to the specific situation. When the operator takes the leading role, the DT tracks the new state with the help of the monitoring unit.

The remainder of the Chapter provides detail on the leading causes of abnormal behaviour and how they have been modelled in the PN-based DT. The detection of some error types is demanded from the monitoring unit. In the following, we suppose that the monitoring unit implements the real-time strategy presented in Chapter 3. However, simpler algorithms can be used as well. The minimum requirements to manage all failure cases that are described in the following is the capability to estimate the expected duration of the ongoing human task and the detection of notable positions reached by the operator's hands. If the aforementioned requirements are not met, the control architecture could still be exploited with limited capabilities to respond to unexpected events.

4.3.1 Robot faults

In this Section, we describe error cases due to robot failures, which have been grouped into three main classes based on the effects on the agent's state and the product being processed:

1. The robot is still available, but the product is unavailable (e.g. the robot makes a mistake during task execution that ruins the product or asks the human intervention after a failed quality check);

2. The robot is unavailable, but the product is still available (e.g. the gripper of the robot gets stuck and it is not able to perform the task);

3. Both the robot and the product are unavailable (e.g. the robot senses a collision and stops while holding the product).

Errors are signalled directly by the robot controller or detected by sensing the workspace. In all cases, human intervention is needed to recover the

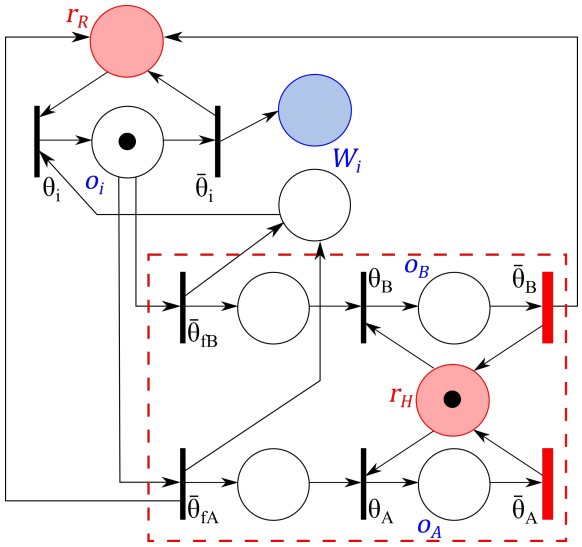


Figure 4.9: *Management of robot faults in the Petri Net model.*

robot and/or the product. Specifically, for each robot operation two recovery actions are added to the PN to model situations where the agent is faulty or not, respectively.

Figure 4.9 shows the new structure of robot tasks. Transitions identified with the letter *A* refer to the first type of fault when the robot is still available. Instead, the second and third error cases are difficult to distinguish automatically and they are modelled in the same way in the PN (*B* label). When the fault occurs, either transition $\bar{\theta}_{fA}$ or $\bar{\theta}_{fB}$ fires depending on the received signal. The token is removed from the place marking the ongoing task and is placed upstream of the recovery action so that θ_A or θ_B is enabled when the human operator is free. In this way, the scheduler can insert the recovery action in the plan. Then, the human is in charge of restoring the robot and the product as needed. In particular, he/she can choose to complete the interrupted operation, discard a ruined product, or partially disassemble it and place the obtained WIPs in the correct buffers. At the end of the recovery action, i.e. when transitions $\bar{\theta}_A$ or $\bar{\theta}_B$ fire, the monitoring system communicates which buffers had been filled and the PN marking is updated accordingly. This also allows discriminating between the two error types that share the same recovery action. If some of the buffers are not reachable by the operator, WIPs can be stored outside the workspace and reintroduced into the process at a later stage, e.g. during shift changes.

4.3.2 Human errors

The present Section investigates error cases that stem from wrong human behaviour. Namely, three significant cases have been identified, which are detected by the human monitoring unit:

1. The operator performs a different operation from the required one (either on purposes or not) that is possible to complete, i.e. all WIPs and resources are available;

2. The operator performs a different operation from the required one (either on purposes or not) that is impossible to complete, i.e. some WIPs or resources are unavailable;

3. The operator makes a mistake during the execution of the correct operation (e.g. he/she forgets a step of the assembly procedure).

In the first case, we conform to human behaviour and the state of the DT is corrected according to the task that the operator is actually performing. Then, a rescheduling is needed to update the future task plan. Conversely, in the second case we cannot comply with the intention of the human. Thus, the error is notified to the operator with the request to start the correct action. The state of the PN evolves as planned, but the expected time for the completion of the operation will increase to account for the wasted time. The reason for distinguishing the two conditions is twofold. On the one hand, we reduce interruptions to the operator's workflow that can lead to greater stress and frustration. On the other hand, the additional time required by the human to become aware of and correct the error often makes the current optimal schedule outdated and justifies adhering to a nominally sub-optimal plan.

The third error condition calls for a corrective action that depends on the specific error and whether the ongoing operation is reversible. Reversible operations are composed of actions that can be undone and repeated without constraints, such as screwing parts together. In this case, the operation can be corrected and completed by the human. Instead, irreversible operations cannot be simply corrected by repeating the same actions, but require more processing. Examples are failed welding or painting operations, that permanently modify the WIP product. In this case, the product must be either discarded or partially disassembled.

The human is notified when the monitoring unit detects the error and starts the corrective action. In the PN, the recovery operation is modelled as exemplified in Figure 4.10. Error detection triggers transition $\bar{\theta}_f$ and a

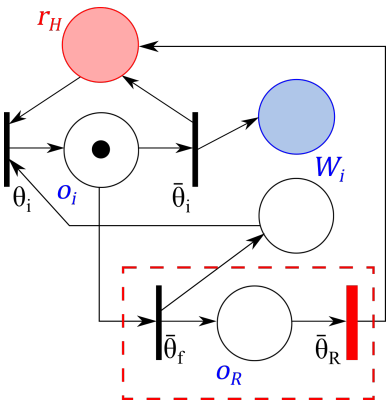


Figure 4.10: *Management of human errors and defective products in the Petri Net model.*

token is moved from the place marking the ongoing operation to a place that identifies the ongoing recovery. Similar to what has been described for robot faults, the human has complete freedom to understand which is the best corrective action for the specific situation. Therefore, his/her actions are monitored by the monitoring unit to correctly update the PN by detecting which WIP products result from the recovery and are placed in the corresponding buffers.

4.3.3 Other failure cases

Two further error conditions are discussed in the following, caused by defective products and failed resources. The presence of defective products cannot be avoided completely in the manufacturing processes, especially in flexible systems with high mix and frequent production changes. Therefore, the operator may find a defective WIP and cannot proceed with the assembly. For instance, it may contain a flawed part that needs to be replaced, or it may be assembled incorrectly during previous steps without the fault being identified. Although conceptually different, this kind of error is equivalent to human mistakes and is managed identically (see Figure 4.10). The only difference is that the detection signal is not provided by the monitoring unit, but it is the operator who communicates the problem to the control system through an appropriate interface, such as a grounded or wearable button. Then, during recovery he/she can repair the WIP, partially disassemble it to a safe configuration, or discard the product.

The last case that has been considered is the failure of resources. For instance, tools might wear or discharge (if powered) and become impossible to use. The operator signal that a resource is currently unavailable

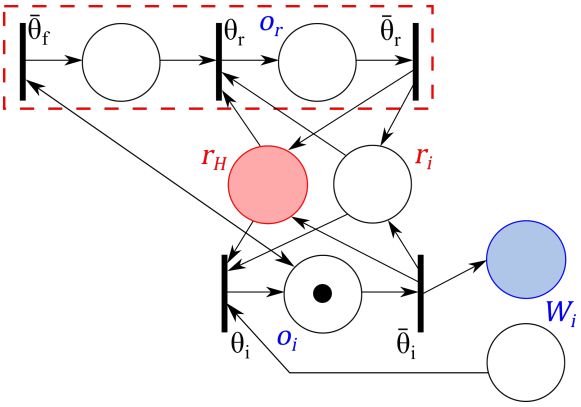


Figure 4.11: *Management of non-necessary resource failure in the Petri Net model.*

through the user interface. Also, workspace sensing can help in determining the state of equipment used by humans or robots. The effect on the assembly process depends on whether the faulty resource is necessary to perform the operation. An example of a non-necessary tool is an electrical screwdriver, which helps the task but can be replaced with a manual one without compromising its outcome. Instead, a welding gun is necessary for task completion. In the case of unnecessary resources, the only effect on the DT is a change in the expected duration of the operations that involve faulty resources until they are repaired. Therefore, the ongoing operation can continue uninterruptedly, but a token is placed upstream of a recovery action that can be scheduled to restore the equipment (see Figure 4.11). Differently, in the case of a necessary resource, the ongoing operation must be interrupted and none of the tasks that require such resource can start until it is repaired (see Figure 4.12). In both Figures 4.11 and 4.12, transition $\bar{\theta}_f$ fires when the fault is detected, then θ_r marks the start of the recovery action and $\bar{\theta}_r$ its conclusion.

Table 4.1 recaps the modelled error cases and associated recovery actions, whereas experimental results are reported in Chapter 5 after that the scheduling algorithm has been introduced.

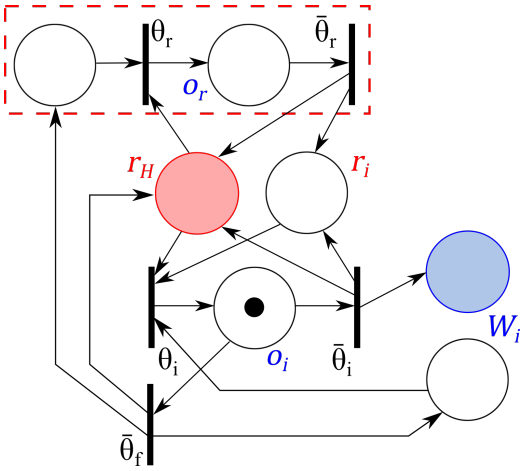


Figure 4.12: *Management of necessary resource failure in the Petri Net model.*

Table 4.1: *List of modelled error cases.*

Cause	Error description	Recovery action
Robot	Robot available, product unavailable	Schedule recovery
Robot	Robot unavailable, product available	Schedule recovery
Robot	Robot unavailable, product unavailable	Schedule recovery
Human	Human starts a wrong but feasible operation	Indulge human
Human	Human starts a wrong and unfeasible operation	Request correct task
Human	Human performs the correct operation wrongly	Immediate recovery
Product	Defective intermediate product	Immediate recovery
Resource	Non-necessary resource unavailable	Schedule recovery
Resource	Necessary resource unavailable	Schedule recovery

CHAPTER 5

Dynamic scheduling algorithm

ONCE that the DT and the monitoring unit are in place, a dynamic scheduling algorithm is needed to determine the optimal instructions for the humans and the robots working in the cell based on information about the current state of the assembly process. As outlined in Chapter 4, the scheduler introduced in this work leverages the DT to predict the future evolution of the system and determine the optimal control action with a receding horizon approach. In particular, the proposed method:

1. Dynamically solves on-line both task allocation and task sequencing to adapt the plan to the process variability;
2. Accounts for the variability in the target production mix, the duration of human tasks, the occurrence of robot faults, and other abnormal system behaviours;
3. Allows for the concurrent assembly of multiple products to increase productivity;
4. Considers many feasible sequences to complete the assembly. The optimal one for each product is selected in real-time and can change

during operation.

To the best of our knowledge, no other work in human-robot collaboration embeds all these features to allow the highest degree of flexibility, which results in the smart use of resources and the minimization of the overall cell idle time. In the following, Section 5.1 details the proposed strategy. Then, Section 5.2 presents a novel visuo-haptic interface to give instructions to human operators. Finally, Sections 5.3 and 5.4 discuss the performance and shows simulation and experimental results, respectively.

5.1 Receding horizon scheduling

During production, the agents in the cell work to repeatedly complete the available jobs according to the target mix. The scheduling problem consists in planning the sequence of operations to perform for each robot and human. Therefore, the output of the algorithm is the optimal set $S = \{(o_i, t_i), (o_j, t_j), \dots\}$ of future tasks and associated start times. The solution to the scheduling problem is based on a Model Predictive Control-like approach. The underlying idea is to predict the evolution of the process over the planning horizon from the current state to determine the control input sequence that minimises a cost function J. The cost function favours productivity and tracking of the target mix. When the optimal plan of future operations has been determined, only the first commands are dispatched to the agents that are currently free, which starts their task. Then, the state of the DT is updated and the schedule recomputed. The receding horizon paradigm allows the schedule to adapt in real-time to the natural variability of the process and unforeseen events, such as faults.

Constraints among tasks arise from the structure of the product, resource requirements, and the workspace layout. For the highest productivity and flexibility, multiple products are assembled concurrently, and it is the scheduler that decides when and which product to start next. Also, the scheduler dynamically chooses the assembly sequence for each product and dynamically associates intermediate WIPs to the type of final product. On the other hand, when a fault occurs, the leading role is handed over to the human, who is in charge of determining the best recovery action according to the specific situation, since it is impossible to model all possible error cases in the DT (see Section 4.3). In this view, the PN-based DT represents the best tool to allow such a high degree of flexibility while providing a computationally efficient framework. The connections between places and transitions inherently embed constraints among operations. A token in the appropriate buffer identifies each concurrent WIP. Moreover, tokens are not

rigidly associated with any product type or assembly sequence. To give an example, we can refer to the assembly process defined by the AAOG in Figure 4.2c. A WIP product $W_5 = \{p_2, p_3\}$ can be completed either as $W_6 = \{p_1, p_2, p_3\}$ or $W_8 = \{p_2, p_3, p_7\}$ depending on which choice optimise production, also considering the production demand.

5.1.1 Digital Twin for simulation purpose

To simulate the future evolution of the robotic cell, and the firing of uncontrollable transitions in particular, a temporal characterization must be included in the model. Therefore, a Timed Petri Net (TPN) [86] description is adopted for the prediction phase. In a TPN, each transition $\theta_i \in \Theta$ can only fire with a delay of at least d_i seconds after it has been enabled. If the firing of a transition disables a previously enabled transition, the delay of the latter is reset when it is enabled again. In our model, controllable transitions have zero delay, as their firing is directly controlled by the scheduler. Instead, uncontrollable transitions linked to task completion have delay equal to the expected duration of the operation, which is computed with the help of a database that records past task durations and is updated each time a new one is completed. Finally, uncontrollable transitions that mark the detection of abnormal system behaviour have infinite delay, i.e. are prevented to fire. This means that only nominal evolutions of the process are simulated, while we manage the occurrence of failures with a reactive approach, leveraging the receding horizon adaptation.

The initial state of the DT that is given as input to the scheduler is composed of the current target production mix $\boldsymbol{y}^0(t)$, the current marking $\boldsymbol{m}_0$ of the TPN, and the remaining time to completion of ongoing operations. The latter is provided by the monitoring unit and determines the initial delay of the associated uncontrollable transitions. Instead, the target production mix is computed starting from the orders that are present in the company's ERP system. Specifically, let $Q_{o,p}$ be the quantity of products of type p requested for the o-th order and a_o the deadline within order Q_o must be completed. Then, the desired throughput for each product type at time t is given by:

$$\omega_p^0(t) = \sum_{i=0}^{n_o} \frac{Q_{i,p} - q_{i,p}(t)}{a_i - t}$$

where n_o is the amount of pending orders and $q_{o,p}(t)$ is the number of products of type p already produced for order o at time t. Orders are fulfilled earliest deadline first. Therefore, when a new product is completed, the produced quantity associated to the corresponding type is in-

creased for the order with highest priority that still requests such product type, i.e. $o^* = \arg\min_o\{a_o | Q_{o,p} > q_{o,p}(t)\}$. When an order is fulfilled, i.e. $Q_{o,p} = q_{o,p}(t)\ \forall p$, or when it expires, i.e. $a_o < t$, it is removed from the pending queue. As a final step, the production mix setpoint is normalised to describe a relative priority among product types:

$$y_p^0(t) = \frac{\omega_p^0(t)}{\sum_{i=0}^{n_p} \omega_i^0(t)} \tag{5.1}$$

5.1.2 Evaluation of feasible future evolutions

When simulating the evolution of the process, the decisions of the scheduler are limited to the firing of controllable transitions. Instead, uncontrollable transitions always fire with the minimum delay. In addition, since an unnecessary delay of controllable transitions increases the idle time of resources, the only meaningful scheduling decision at each step is either to fire one enabled controllable transition immediately or wait for the firing of an uncontrollable one. The latter choice means to wait for the end of an ongoing operation, that can enable new alternatives. If several uncontrollable transitions are enabled at the same time, the scheduler can only choose to wait for the one with minimum delay, which is the first to fire. Otherwise, the uncontrollable transitions with smaller delay are no longer able to fire at the correct time. Overall, the scheduling problem reduces to determining the optimal firing sequence of transitions.

From the initial state, feasible system evolutions are found by exploring the Reachability Tree (RT) of the TPN. The compact matrix representation of the PN model and the simple update rule given by equation (4.1) allow for efficient simulation of the system. The RT is a pair (V, A), where V is a set of nodes connected by arcs in A. Each node v_i represents a reachable state for the TPN and is described by $(\boldsymbol{m}_i, t_i, J_i)$, that is the corresponding marking vector, the arrival time to the state, and its cost. Instead, each arc a_{ij} represents the transition whose firing brings the system from v_i to v_j. A path between two nodes v_i and v_j, referred to as $\langle v_i, v_j \rangle$, can be equivalently described by the set of connected nodes, i.e. states of the PN, or the sequence connecting arcs, i.e. fired transitions.

The expansion of the RT starts from the root node $v_0 = (\boldsymbol{m}_0, 0, 0)$, where $\boldsymbol{m}_0$ is the current marking of the DT. Then, one branch per each feasible scheduling choice departs from the current node to explore all possible alternatives. For a node v_b to be the child of node v_a, the transition associated with a_{ab} must be enabled in v_a. In case of controllable transitions, the arrival time to v_b is $t_d = t_a$. Conversely, to determine the arrival

time when an uncontrollable transition triggers, it is important to know the moment when it was enabled. The farthest enabling ancestor of the node v_b, is the node $v_e \in \langle v_0, v_b \rangle$, such that $\langle v_e, v_b \rangle$ is the largest sub-path that contains only nodes for which the transition leading to v_b is enabled and never fires. Then, the arrival time to the state v_b is equal to $t_b = t_e + d_{ab} \geq t_a$, where t_e is the arrival time to the farthest enabling ancestor of v_b and d_{ab} is the delay associated with the transition described by arc a_{ab}. Note that the case $t_e + d_{ab} < t_a$ describes an unfeasible system evolution where the uncontrollable transition is not able to fire with the correct delay. A general rule to compute the arrival time for any transition reads as:

$$t_b = \max\{t_e + d_{ab}, t_a\}$$

The exploration of the RT stops in one of the following two cases: either the node is a leaf where no transitions are enabled, or the planning horizon limit h_{exp} has been reached, which is expressed in terms of the number of operations in the sequence. The value of the planning horizon is guided by a trade-off between computational load and performance that is better discussed in Section 5.3.2 based on experimental data.

The optimal path in the RT is $\langle v_0, v_* \rangle$ from the root node to the leaf v_* with minimum cost J_*. Then, the final schedule S is obtained from the optimal path, as the sequence of controllable transitions along the path and their firing time. The cost function minimised in the evaluation of the optimal schedule is:

$$J(v_i) = J(\langle v_0, v_i \rangle) = k_S t_i + k_I \sum_{j=1}^{|A|} c_j t_j^I + k_W \sum_{j=1}^{|B|} t_j^W \qquad (5.2)$$

where k_S, k_I, k_W are weight coefficients and t_j^I, t_j^W are functions of the TPN marking and arrival times of nodes in $\langle v_0, v_i \rangle$, which in turn depend on the sequence of fired transitions, i.e. the control inputs. Furthermore, $A \subseteq R$ is the set of agents, that are described in the PN examples with a place with red background, and $B \subseteq N$ is the set of intermediate buffers, whose place indicating the occupied slots is highlighted in blue in the figures (see Section 4.2.2). The first term reduces the time of completion of the last operation in the plan. The second term penalises the agents' idle time, with c_j the unit idle cost for agent j and t_j^I the amount of time that agent j spends being idle in the interval $(0, t_i)$. Specifically, t_j^I is computed as the total time during which the agent's place (with red background) is marked during the simulation. Finally, the last term facilitates the product flow by minimizing the waiting time of parts in intermediate buffers. In

Section 4.2.2, the number of WIP products stored in a buffer was represented as an equivalent amount of tokens inside the place marked with blue background. Therefore, the total time t_j^W that parts wait in the j-th buffer is equal to the sum of the intervals during which the blue place is marked, weighted with respect to the number of tokens inside such place.

The definition of the cost function allows computing the cost of a child node v_c incrementally from the one of its parent node v_p as:

$$J(v_c) = J(v_p) + \left(k_S + k_I \sum_{p_j \in A} c_j m_c(p_j) + k_W \sum_{p_j \in B} m_c(p_j) \right) (t_c - t_p) \quad (5.3)$$

where $m_c(p_j)$ stands for the number of tokens in place p_j and $J(n_0) = 0$. The weighting coefficients are chosen to penalise more the production time, i.e. the first term in the equation (5.2), so to increase productivity. A higher WIP storage penalization distance allows better tracking of the reference mix and favours product flow. Conversely, a lower value of k_W tends to accumulate unfinished products in the cell, increasing flexibility. Finally, the remaining term of the cost is useful to discriminate among schedules that attain similar costs. The cost function parameters can change in time to reflect variations in production conditions.

Figure 5.1b shows an example of RT exploration starting from the PN in Figure 5.1a. The process comprises two agents that assemble two types of products performing either operation o_1 or o_2. Initially, r_2 is performing o_2 with remaining time to completion equal to $\bar{d}_2$ (it is supposed that $\bar{d}_2 < d_1 < d_2$). Two transitions are enabled, θ_1 and $\bar{\theta}_2$, therefore two branches depart from the root node. Overall, six different system evolutions are feasible to complete the production. When a controllable transition triggers, the cost of the child node is equal to the parent's one, as the arrival time is the same. Conversely, the firing of uncontrollable transitions increases the cost. Nodes with the same arrival time can have different costs depending on the path leading to the state. For instance, the cost of node v_{14} is higher than v_{12} due to higher idle time of resources in the former case.

Simulation of recovery actions

Transitions associated with the occurrence of faults are never triggered during the exploration of the RT. However, the initial condition of the DT can describe the process in a failure state. Although most of the information required for the correct management of error cases are already embedded in the PN model, particular attention must be paid in simulating the recovery from specific failure conditions described in Section 4.3, namely those re-

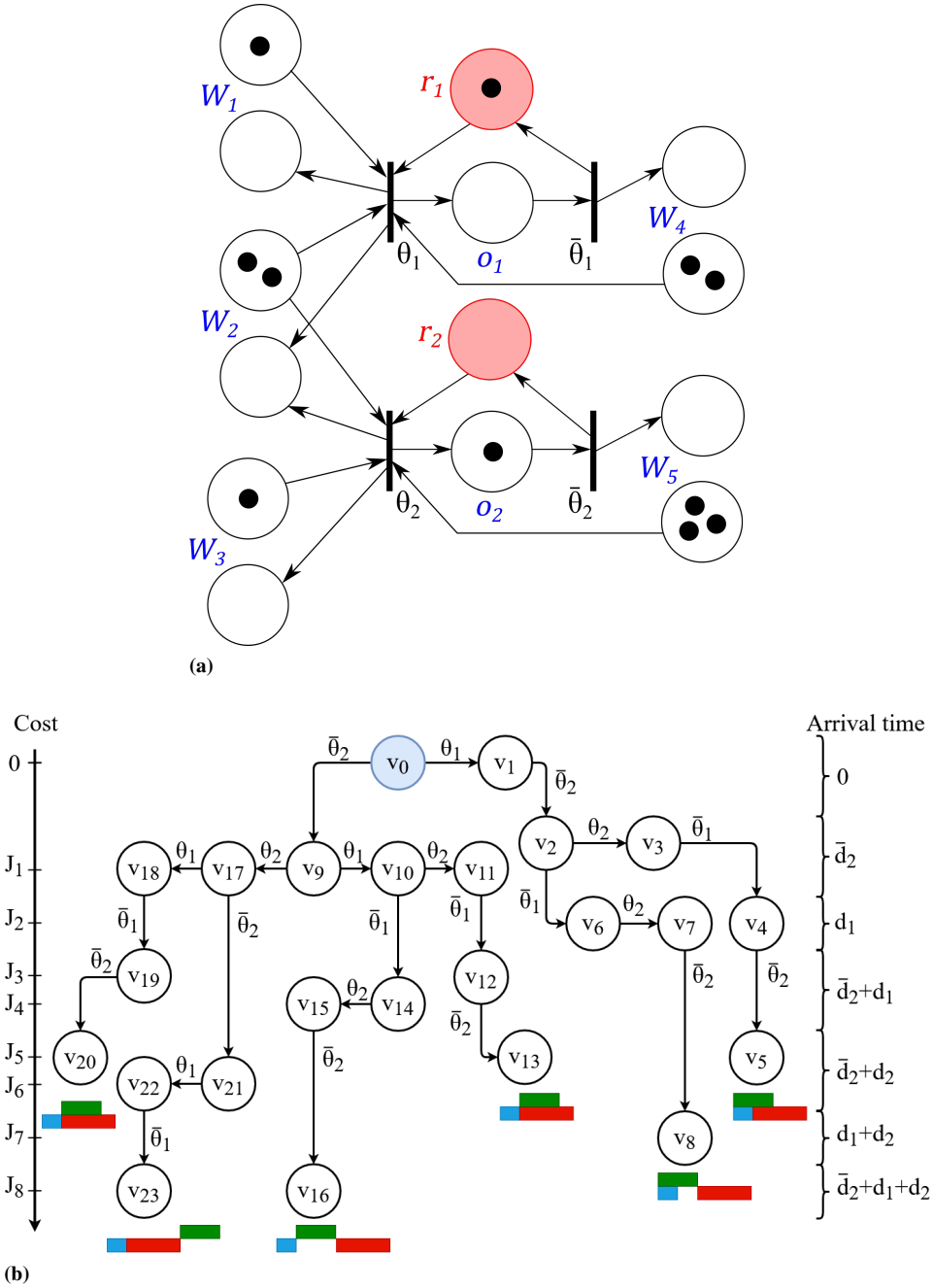


Figure 5.1: *Example of PN (a) and corresponding RT (b). Initially, agent r_2 is performing task o_2 ($\bar{d}_2$ is the remaining time to completion), while r_1 is idle. It is supposed that $\bar{d}_2 < d_1 < d_2$. Labels on arcs indicate fired transitions. Costs are computed from equation* (5.2)*. The resulting schedule is depicted for each leaf of the RT.*

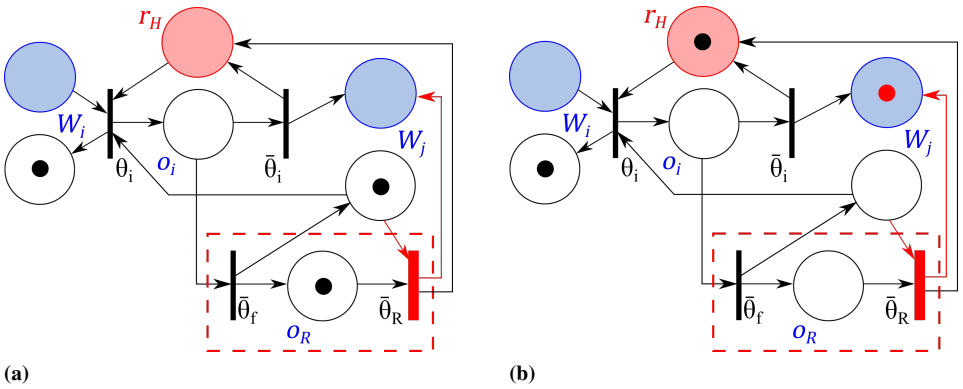


Figure 5.2: *Example of simulation of default recovery actions.*

quiring the human to choose the best action to perform. In these cases, the PN provides only a partial description of the effects of the recovery action, which must be complemented with information coming from workspace monitoring. In other words, to update the marking vector after the completion of a recovery action it is not sufficient to apply the rule in (4.1), but ad-hoc modifications are needed to correctly track the process state. The uncontrollable transitions we are referring to are those highlighted in red in Figures 4.9 and 4.10.

Obviously, data from the monitoring unit are not available when simulating future evolutions of the process. Therefore, a choice on the default effects of recovery action must be made to explore reasonable system evolutions, which depend on the specific operation and is decided by process experts. The choice must reflect the most likely outcome of the recovery action. For instance, an error during paining operations would result in the product being discarded most of the time, while a defective part during assembly could allow restoring the previous WIP. Sometimes, the simulated recovery action might differ from the actual future execution of the operation. As a consequence, the scheduler will update the plan accordingly at the next step. Figure 5.2 shows a case where the default recovery is to correct the error and complete the ongoing operation. The red arrows indicate the arcs that are added in the TPN to model the default effects of the recovery action. Specifically, Figure 5.2a depicts the initial condition after the error detection, while Figure 5.2b the TPN state after recovery.

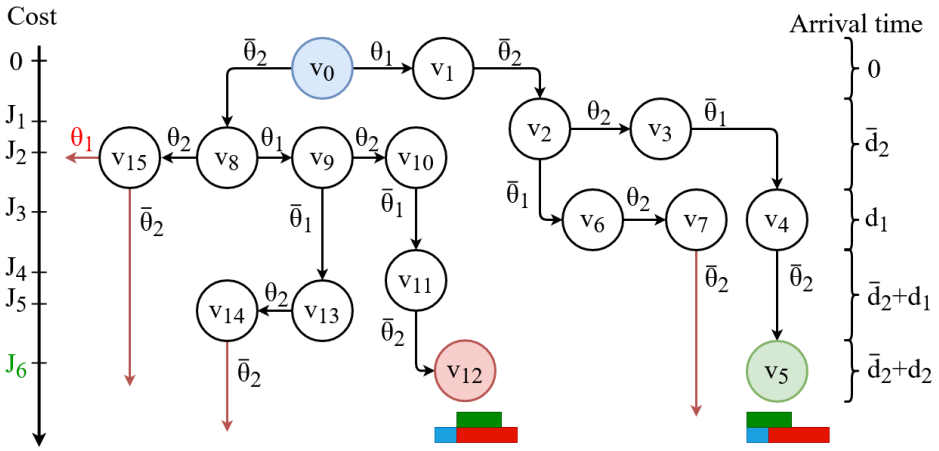


Figure 5.3: *RT from the PN in Figure 5.1a following pruning rules (except the learning-based one). Nodes are numbered in order of exploration, red arcs mark unexplored branches. θ_1 is enabled in v_{15} but would lead to the same state as v_{10}.*

5.1.3 Pruning strategies

Pruning strategies are implemented to speed up the exploration of the RT without loosing optimality. The tree is explored depth-first to rapidly find a feasible evolution of the system, then new nodes are generated only if their cost is smaller than the current optimal one. When both controllable and uncontrollable transitions are enabled at a node, we first explore controllable branches, which do not increase the cost. Also, the exploration of duplicate branches is avoided by noting that changing the order of two subsequent controllable transitions generates equivalent evolutions. Figure 5.3 shows the same RT as Figure 5.1b when pruning rules are applied. The first exploration leads to the leaf v_5, whose cost is taken as the current best. The exploration of the second branch stops at node v_7 as its child would have a cost higher than v_5. Instead, v_{12} attains the same cost as v_5, but the latter is preferable as it anticipates the start time of operations. For this reason, the best node is changed only if the cost of the new leaf is strictly smaller than the previous minimum. All other branches in the example are pruned. In particular, θ_1 is enabled in v_{15} but would lead to the same state as v_{10}.

A relevant hypothesis that limits the number of possible system evolutions is the one concerning the values assigned to the firing delays. As already stated, controllable transitions are only fired with zero delay or not triggered altogether. In principle, any value is possible and should be considered for an exhaustive search. To do so, a parametric description of

the arrival time and the cost of nodes would be necessary, which would increase complexity considerably. However, delays different from zero surely lead to sub-optimal solutions. On the other hand, we consider deterministic delays for uncontrollable transitions, equal to the expected duration of tasks. In reality, the duration of operations is a stochastic quantity, especially when human activity is considered. Thus, one operation might finish before another even if it should be concluded after in nominal conditions. A probabilistic description of task duration would be required, which in turn would lead to a probabilistic value of arrival times and the introduction of the concept of arrival probability to states. Then, the optimal scheduling choice would be the one minimizing the expected value of the final cost, similar to what is done in [19].

In other words, in this work we aim to minimise the cost in the expected condition, while in [19] the authors minimise the expected cost. Although the latter approach is more robust against the system variability, it is computationally expensive. Therefore, only approximate solutions are suitable for small collaborative processes with few agents and scheduling choices, but the load becomes prohibitive when larger manufacturing cells are considered. Exploring only the nominal evolutions of the system is a reasonable assumption that strongly reduces the size of the problem. On the one hand, the receding horizon approach reduces the importance of the single scheduling choice, since it can be corrected at the next iteration. In addition, the two strategies provide the same result in a neighbourhood of the nominal condition, which describes the most probable system evolution. In fact, from equations (5.2) and (5.3) one can see that the cost function is linear in the arrival times of nodes in the path, that is:

$$J(\langle v_0, v_i \rangle) = \sum_{v_j \in \langle v_0, v_i \rangle} j_j t_j$$

where j_j is function of the cost function parameters and the TPN markings. If the duration of tasks is considered to be stochastic, also the arrival times to the nodes is a stochastic variable. Thus, the cost in the nominal condition is given by:

$$J(\mathbb{E}\left[\langle v_0, v_i \rangle\right]) = \sum_{v_j \in \langle v_0, v_i \rangle} j_j \mathbb{E}\left[t_j\right] = \mathbb{E}\left[\sum_{v_j \in \langle v_0, v_i \rangle} j_j t_j\right] = \mathbb{E}\left[J(\langle v_0, v_i \rangle)\right]$$

where the second equivalence holds as long as there are no changes in the order of transition firings. A comparison of the performance of the two approaches, as well as with other state-of-the-art schedulers, is given in Section 5.3.4.

Learning-based heuristic pruning

To further reduce the dimension of the RT in case of large assembly processes, we developed a learning-based heuristic that learns constraints for node acceptance. Constraints are defined with respect to the increment of the arrival time $\Delta T_i = t_i - t_p \geq 0$ and cost $\Delta J_i = J_i - J_p \geq 0$ between a new candidate node v_i and its parent node v_p. The region of acceptance is limited by:

$$\begin{cases} \Delta T_i \leq \Delta \overline{T}_k \\ \Delta J_i \leq \Delta \overline{J}_k \\ \Delta J_i \leq \overline{m}_k \Delta T_i + \overline{q}_k \end{cases} \tag{5.4}$$

where $\boldsymbol{p}_k = \left[\Delta \overline{T}_k, \Delta \overline{J}_k, \overline{m}_k, \overline{q}_k\right]^T$ defines the boundaries at step k, equal to the number of previously completed schedules. During the (k+1)-th exploration of the RT, a node is accepted according to the policy:

$$\pi_k(\Delta T_i, \Delta J_i) = \begin{cases} 1 & \text{if } inside(\Delta T_i, \Delta J_i, \boldsymbol{p}_k) \\ \varepsilon_k & \text{otherwie} \end{cases}$$

where $inside(\Delta T_i, \Delta J_i, p_k)$ is satisfied if and only if the node is inside the acceptance region (5.4), and $\varepsilon_k \in [0, 1]$ is a vanishing probability value. The policy accepts all nodes inside the acceptance region (exploitation), while points outside the region are accepted with probability ε_k (exploration).

When the first scheduling takes place, $\boldsymbol{p}_0 = \left[\Delta \overline{T}_0, \Delta \overline{J}_0, \overline{m}_0, \overline{q}_0\right]^T$ must define a sufficiently large region so to accept all nodes and gather the initial information on the distribution of nodes in the plane $\Delta T - \Delta J$. Subsequently, the boundary values are iteratively approximated to reduce the dimension of the acceptance region based on the data collected in the previous RT explorations. The acceptance region should contain all the nodes belonging to the optimal paths of all explored RTs. Therefore, a vector $\boldsymbol{x}_k = [\Delta T_k, \Delta J_k, m_k, q_k]^T$ collects the worst-case observations among all the optimal nodes. After each new exploration of a RT, $\boldsymbol{x}_k$ is updated as:

$$\begin{aligned} \Delta T_{k+1} &= \max\{\Delta T_k, \Delta T_i\} \\ \Delta J_{k+1} &= \max\{\Delta J_k, \Delta J_i\} \\ m_{k+1} &= \max\left\{m_k, \frac{\Delta J_i - \overline{q}_k}{\Delta T_i}\right\} \qquad \forall n_i \in \langle v_0, v_* \rangle \\ q_{k+1} &= \max\{q_k, \Delta J_i - \overline{m}_k \Delta T_i\} \end{aligned}$$

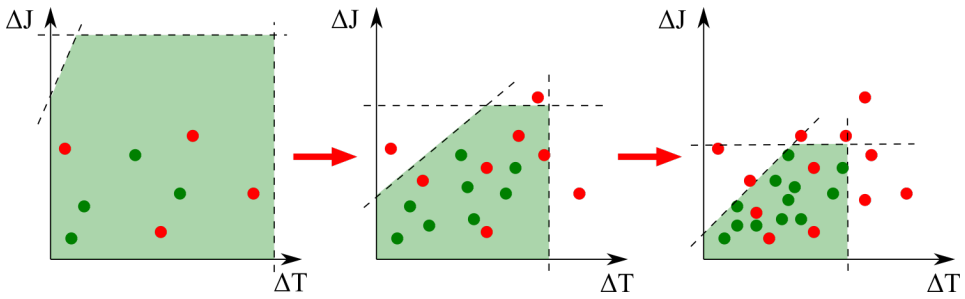


Figure 5.4: *Example of progressive reduction of the acceptance region. Green points describe nodes of the RT belonging to optimal paths, red points are sub-optimal nodes.*

Then, constraints are updated according to:

$$\boldsymbol{p}_{k+1} = \begin{cases} (1-\alpha)\boldsymbol{p}_k + \alpha\boldsymbol{x}_{k+1} & \text{if } \boldsymbol{x}_{k+1} \leq \boldsymbol{p}_k \\ \boldsymbol{x}_{k+1} + \beta\boldsymbol{p}_0 & \text{if } \boldsymbol{x}_{k+1} > \boldsymbol{p}_k \vee \nexists v_* \end{cases} \tag{5.5}$$

where α and β are positive parameters in the interval $(0, 1)$. The update rule (5.5) imposes a low-pass filter dynamics on the distance between the current boundaries and worst-case observation as long as $\boldsymbol{x}_k$ lies inside the acceptance region. Conversely, the constraints are relaxed when a node belonging to an optimal path lies outside the current limits or no solutions to the scheduling problem are found altogether, which means that good nodes have been discarded. In this case, also the value of ε is reset to favour further exploration. Adaptation must be slow enough to gather sufficient data prior to the convergence of the acceptance region, so that the probability of discarding good nodes is reduced as much as possible. Figure 5.4 exemplifies the evolution of the heuristic pruning constraints from the initial loose boundaries until an acceptance region that fits the historical optimal points. The computational load and performance reduction resulting from the application of the proposed strategy are discussed in Section 5.3.1.

5.1.4 Dispatching and replanning

Once that the optimal schedule has been determined, the first commands are sent to the free agents, then the DT continues to monitor the cell and the plan recomputed from the updated state. Specifically, replanning is triggered any time an operation ends in order to adapt to the actual duration of tasks, and when a fault occurs to readily plan the best management strategy.

At the beginning of Section 5.1, we said that the schedule is also in charge of deciding when and which product to start next. As long as there

are pending orders, it is always advisable to work at full capacity and fulfil orders earlier, instead of just in time, since new production orders can arrive at any time and saturate the plant. Therefore, we start the assembly of a new product every time the last computed schedule is composed of a number of operations smaller than a predefined threshold h_{new}. The value of h_{new} must be strictly smaller than, but close to, the exploration horizon h_{exp} so that the pending operations are always enough to guarantee a rich and informative exploration of the RT.

The choice of the product type to start is determined by the highest product priority $\boldsymbol{y}_p^0(t)$, which measures the current distance from the target mix. The insertion of a new product is managed by governing the content of the buffers that store the base parts. First, the logistic process that is in charge of replenishing such buffers is not modelled, but it is assumed that they are never empty. Conversely, they are initially marked as empty in the TPN. Then, when a new product must be introduced in the process, it is sufficient to add one token for each base part that composes the product in the corresponding buffer. In this way, transitions that model the start of the first assembly step becomes enabled and can fire in the exploration of the RT. An important remark is that the order with which products are added to the assembly cell does not necessarily equal the order with which their production actually starts or is completed. In fact, the scheduler is in full charge of determining the production sequence. For instance, in case the operations needed to assemble a product with lower priority fit the current state of the process better and lead to more efficient schedules, they might start and finish before those related to the highest priority product, although the latter has been introduced earlier in the cell.

5.2 Haptic interfaces

After a suitable scheduling algorithm has determined the optimal assembly plan, a bilateral communication is needed to communicate with the agents that are working in the cell. In particular, the scheduler outputs the commands for the humans and the robots, associated with the operations to start. Conversely, agents communicate their status to the control system, notifying operation completion and the occurrence of other events, such as robot faults. With this information, the DT can correctly track the state of the process and the scheduler can plan the next operations as explained in Section 4.1. Figure 5.5 shows the general principles behind the proposed paradigm. On the one hand, robots are in direct communication with the CPU where the scheduler runs by means of an electrical or wireless

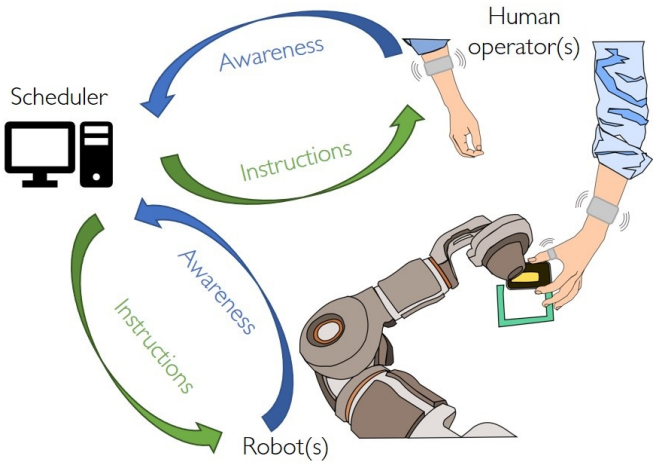


Figure 5.5: *Communication paradigm: the scheduler communicates instructions to the human(s), through haptic interfaces and/or a screen, and to the robot(s). Agents make the scheduling algorithm aware of their status, communicating operation completion and other information, like robot faults.*

connection. On the other hand, intuitive user interfaces are needed to communicate with the human worker, informing him/her of the next operation to perform. In this Section, we present a novel way of communicating instructions to the human operators through wearable vibrotactile interfaces, which has been designed in collaboration with the SIRSLab of the University of Siena.

Works on human-robot interaction and scheduling algorithms usually rely on screens to display instructions to users [68, 142]. Whenever human sight is impaired or needs to be free, other sensory channels could be used as well. Moreover, multimodal communication interfaces become essential to allow for a rich exchange of information between the agents [5]. Methods to improve human situation awareness in HRC through visual, auditory, and tactile feedback have been proposed in the literature [18, 32, 159]. To improve robot awareness of human operations, different strategies have been used, from human intention prediction algorithms [170] to flexible scheduling based on visual monitoring of human and robot actions [19]. Previous work showed that tactile feedback outperforms visual and auditory signals for sensory substitution [105], while visuo-haptic feedback leads to a reduction in reaction times and a better attention allocation than visual feedback alone [13, 152]. It has been shown that human-robot interaction can benefit from wearable haptics by exploiting tactile signals for spatial guidance [146], pace suggestion [90], task awareness [18], and command

Figure 5.6: *Haptic devices: ring with a vibrating motor and three buttons and bracelet containing the controller box and two vibrating motors.*

acknowledgement [44]. The wearability of interfaces allows human body parts to move freely and perform the assigned task without difficulty. Also, the worker is not constrained to a predetermined position in the workspace, where the grounded interface is at hand.

In this work, communication with the human is enabled by wearable devices like those shown in Figure 5.6, which are composed of a ring and a bracelet. The ring contains three push-buttons and a 3 mm vibration motor, which are controlled through an Arduino Pro Mini[1]. The same Arduino controls the two 25 mm vibration motors of the bracelet. The communication between the devices and the scheduler is wireless, thanks to two XBee® RF modules[2].

5.2.1 Human input to the scheduler

During operation, the human must communicate the completion of his/her current task to the control system. This allows the Digital Twin to keep track of the current process state, as described in Section 4.2. The signal fires the uncontrollable transition that marks the end of the ongoing task so to update the availability of WIP parts and resources for the next operation. Also, the change in the cell status triggers the re-computation of the schedule, which can adapt to the actual duration of the task just finished (see Section 5.1). One way to obtain the required information is to monitor the activity of the human in real-time, e.g. by exploiting one of the algorithms described in Chapter 3. Otherwise, one can use simpler solutions that do not require continuous monitoring of the operator, especially in cases when the human behaviour variability is low, or when monitoring is difficult due

[1] `store.arduino.cc/arduino-pro-mini`
[2] `www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Datasheet.pdf`

to characteristics of the environment or the task.

Using the proposed wearable device, the human can explicitly communicate the completion of his/her current task by pressing a button on the ring. Other buttons have been used to signal abnormal conditions, e.g. the presence of defective products, to the system (see Section 5.4.3). As the human presses the button, a vibration burst (duration = 150 ms, frequency = 200 Hz, amplitude = 0.6g) from the motor of the ring acknowledges that the pressure was correctly recognised. The importance of the acknowledgement was recognised by the users during the experiments and the wearability of the interface allowed for prompter user input than using a push-button located on the workbench.

5.2.2 Instructions from the scheduler to the human

To transmit instructions to the human worker, we propose to combine visual and tactile feedback, introducing a new method to display instructions through wearable vibrotactile interfaces. Visual instructions are static images on a fixed screen that display the name and a picture of the main stages of the operation to be performed. Different background colours emphasise waiting and fault-related commands to ease identification. Thanks to two vibrotactile bracelets worn on the two arms, the operator can also receive haptic signals indicating his/her next task. The approach we use to send haptic signals depends on the type of task to perform and mixes typical features of spatial guidance paradigms [146] with techniques to send complex messages through *tactons* [69].

For failures, emergencies, or other high-priority operations that require particular attention or immediate intervention, we use a fast sequence of several vibrations (e.g., in the experiments we used 10 vibrations lasting 50 ms with an interval of 50 ms between them). This type of signal was found to be suitable to communicate the importance of an event [56]. Depending on which bracelet is activated and for how long the train of vibrations lasts, different messages can be conveyed.

For assembly operations, i.e. those that are part of the productive cycle and are not responses to unexpected events, the approach sketched in Figure 5.7 is followed. When the left (right) bracelet vibrates, the human has to work in the left (right) part of the workspace. The number of vibration bursts indicates the sector associated with the starting point of the task to be performed, where the user should move the hands. In practice, a sector may indicate a buffer that contains a part or a tool required for the assembly operation. In our experiments, vibration bursts lasted 100 ms, had a frequency

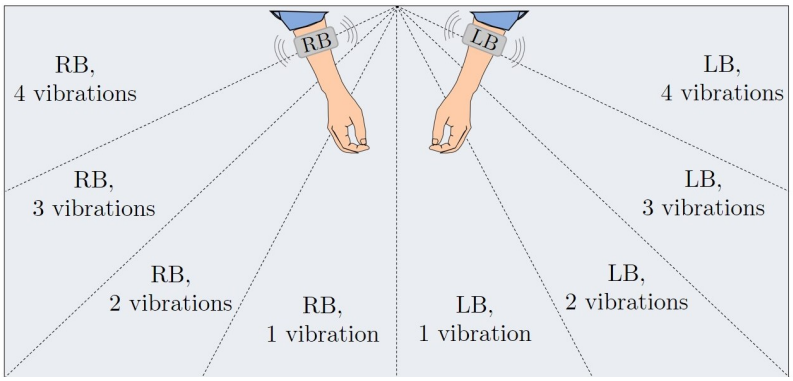


Figure 5.7: *Haptic communication principle: the operator is guided towards the sector of the workspace associated with the next operation that he/she must perform. The actuated bracelet indicates the direction from the human viewpoint (LB = left bracelet, RB = right bracelet), the number of vibrations indicates the sector.*

of 220 Hz, an amplitude of 0.7g, and were equally spaced of 100 ms.

To the best of the author's knowledge, this is the first attempt to combine visual and tactile cues to communicate instructions during a human-robot collaborative task supervised by a dynamic scheduler. The proposed paradigm is easy to learn and general, as it unequivocally associates each operation to a specific tactile cue using a simple rule, and it can be applied to different set-ups, provided that each portion of the workspace corresponds to at most one operation. Although this might seem a strict constraint, it is well suited to most structured activities, such as industrial assembly tasks, as the workspace layout can be designed accordingly.

5.3 Simulations

Before proceeding with the experimental campaign, simulations have been exploited to give better insight into the behaviour of the control architecture and assess the performance. To obtain realistic results, simulations have been run with reference to the same use-case implemented for the experiments, which is better detailed in Section 5.4. The duration of operations has been modelled as Gaussian variables with mean and variance computed from real data collected from the collaborative cell.

5.3.1 Heuristic pruning performance

First, the effects of the heuristic pruning strategy presented in Section 5.1.3 are analysed in terms of computational load and system productivity. The

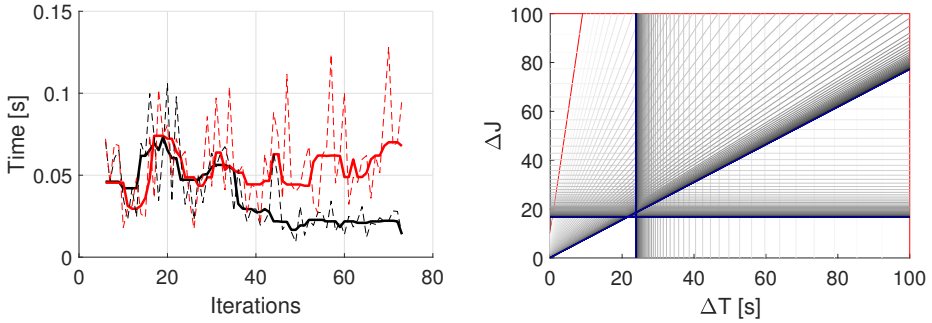


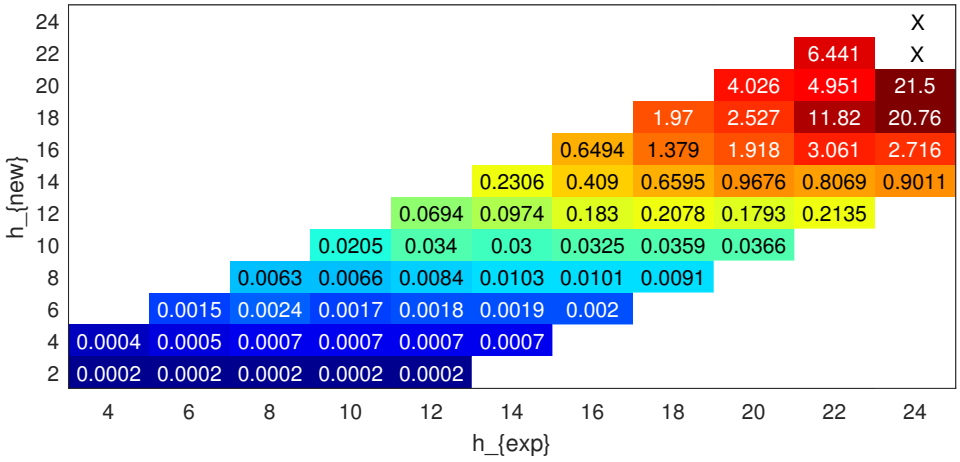
Figure 5.8: *Performance of the heuristic pruning strategy. (a) Evolution of the average scheduling time with (black) and without (red) heuristic pruning. Solid lines report median filtered values. (b) Example of evolution of the constraints boundaries from the initial (red) to the final (blue) ones.*

latter is quantified as the average cycle time, i.e. the time between the completion of two subsequent products. The initial and final parts of each simulation have been neglected to remove transient behaviour.

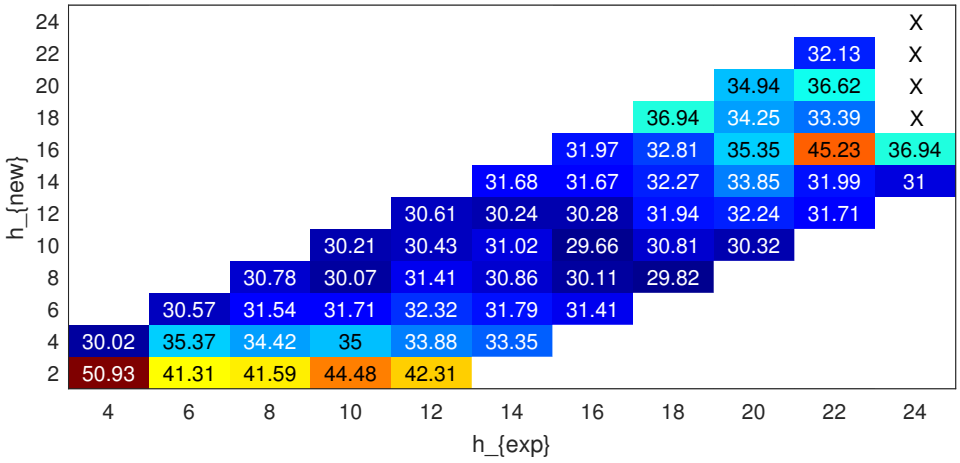
The algorithm parameters have been set to $p_0 = [100, 100, 10, 10]^T$ and $\alpha = \beta = 0.1$. Ten simulations have been run, each one consisting of the assembly of 30 products. Five with the heuristic pruning active and five without. Results show that the optimality of the scheduling is preserved, as the cycle time obtained with heuristic pruning does not significantly differs from the one attain without. However, the time required to solve the scheduling problem strongly reduces after the initial learning transient. Figure 5.8a shows the evolution of the average scheduling time as the simulation progresses. When pruning is disabled, the computational load remains constant throughout the simulation (red line). When pruning is active (black line), the scheduling time does not differ as long as the acceptance region is still large, but rapidly drops when the constraints start discarding unpromising nodes. At steady state, the median scheduling time without heuristic pruning is 0.0618 s against a value of 0.0218 s with pruning (64.7% reduction). An example of the evolution of the boundaries of the acceptance region is provided in Figure 5.8b.

5.3.2 Sensitivity analysis

A critical point in obtaining good results is the selection of the values of parameters h_{exp} and h_{new}. Therefore, a sensitivity analysis has been per-



(a)



(b)

Figure 5.9: *Heatmaps reporting the average scheduling time (a) and production cycle time (b) varying h_{exp} and h_{new}.*

Table 5.1: *Production orders.*

Arrival time	Deadline	P_1	P_2	P_3
0 s	550 s	5	5	0
100 s	750 s	0	5	10
300 s	800 s	0	5	0

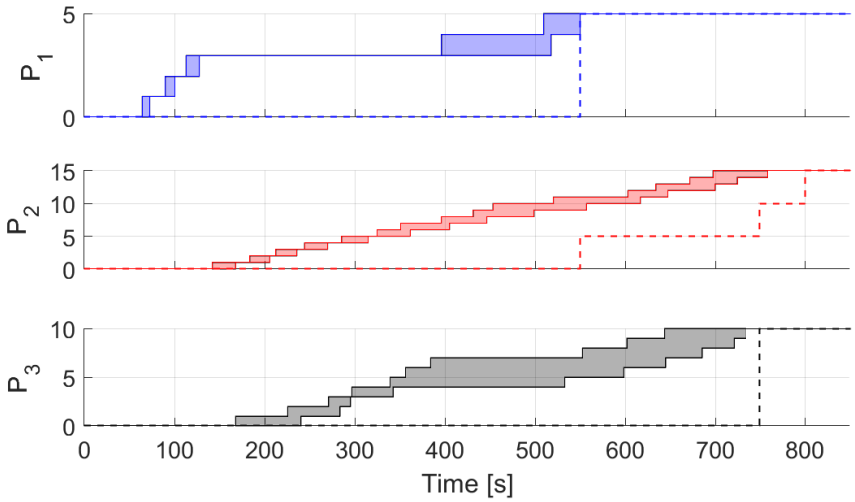


Figure 5.10: *Target mix tracking: variability of the actual production in 10 simulations (shaded areas) and production orders to satisfy (dashed).*

formed to assess how this choice influences the performance and the computational load. Figure 5.9 reports the results obtained using an off-the-shelf laptop (Intel Core i7-8550U CPU 1.80GHz, 8 GB RAM) by varying h_{exp} from 4 to 24 in steps of 2 and h_{new} ranging from 2 to h_{exp}. For each combination of the parameters, the results are the average among ten simulations. As expected, the computational load, expressed as the average scheduling time, increases as the parameter values increases, as shown in Figure 5.9a (missing values refers to times greater than 50 s). A higher planning horizon h_{exp} means that the RT is explored to a greater depth; a higher h_{new} increases the number of concurrent products in the cell and, consequently, the branching factor of the RT. On the other hand, Figure 5.9b shows that performance is bad for smaller values of the parameters, since the scheduler is not able to gather enough information to determine the best control action. As the planning horizon increases, the cycle time reduces until it reaches the optimum. Then, it starts worsening again due to the higher scheduling times that result in the agents waiting for the next instruction from the system. Moreover, planning past a certain depth provides little added value, as predictions lose accuracy due to the high variability of the system. Missing data in Figure 5.9b refer to cases where the cycle time was greater than 100 s. In the remainder of the Chapter, we set $h_{exp} = 16$ and $h_{new} = 10$, which attained the best cycle time with reasonable computational load.

5.3.3 Target mix tracking

To test mix tracking capabilities of the scheduling approach, we extended the use-case to comprise three different types of products. Two of them (P_1 and P_2) are variants of the same product that differentiate half-way in the assembly sequence, while the third one (P_3) is a separate product. We run ten simulations to show the robustness of the plan against the system variability. The production orders are reported in Table 5.1, while Figure 5.10 shows the variability of the actual production. Overall, the scheduler was able to meet the production requirement every time adapting the mix to the pending orders. For instance, the production of P_1 is put on hold after the arrival of the second order, as P_2 and P_3 gain higher priority. Then, the remaining P_1 products needed to complete the first order are assembled when the deadline approaches.

5.3.4 Comparison with other schedulers

To conclude, we performed a simulation campaign to compare the performance of the proposed algorithm with other schedulers that are available in the literature, which were introduced in the introduction to Chapter 4. None of the reviewed approaches present all the features that allow our control system to achieve the maximum degree of flexibility, namely dynamic task assignment and sequencing, online fault management, and robustness to the system variability (especially in the duration of the operations).

Lou et al. [93] propose a decentralised algorithm that combines offline proactive scheduling with online reactive repair according to predetermined rules. Known and fixed Gaussian distributions for task durations are supposed to compute the robust offline plan. Instead, Nikolakis et al. [112] describe a similar centralised approach where multiple alternatives are evaluated offline to find the best nominal plan, which is then adapted online in case of abnormal events. However, task duration is considered to be deterministic. The fixed duration of the operations is also assumed in [49]. Although the approach is suitable for online task scheduling and fault management, it does not aim to find an optimal solution, but only one that satisfies constraints. Instead, [59] and [150] propose heuristic strategies for job-shop order scheduling and cycle time optimization in manufacturing plants, respectively. However, both works are difficult to apply to the use-case considered in this thesis. In [21], genetic algorithms are employed to minimise production time and cost. The authors consider exponential distributions for task durations and do not account for errors and faults. Johannsmeier et al. [68] propose a hierarchical framework for optimal task planning in

Table 5.2: *Comparison with other schedulers.*

Scheduling algorithm	Online assignment	Online sequencing	Fault management	Uncertain duration
Proposed approach	✓	✓	✓	✓
Casalino *et al.* [19]	✗	✓	✗	✓
Chen *et al.* [21]	✓	✗	✗	✓
Gombolay *et al.* [49]	✓	✓	✓	✗
He *et al.* [59]	✗	✗	✗	✗
Johannsmeier *et al.* [68]	✗	✗	✗	✗
Lou *et al.* [93]	✗	✓	✓	✓
Nikolakis *et al.* [112]	✓	✗	✓	✗
Shi *et al.* [150]	✗	✓	✗	✗
Wilcox *et al.* [165]	✗	✓	✗	✗

human-robot assembly. However, both task allocation and sequencing are solved offline with respect to the nominal task durations. The work of Wilcox et al. aims to adapt the robot plan to the operator's preferences without considering productivity. Finally, Casalino et al. [19] uses Timed Petri Nets to schedule robot tasks in collaborative assembly by predicting human intentions, which are assumed to be uncontrollable. Section 5.1.3 already provided some insight into the different approaches to optimization and the exploration of the feasible system evolutions between [19] and the strategy proposed in this thesis. Table 5.2 summarises the characteristics of the schedulers available in the literature as compared to the main features of the proposed approach.

To carry out the performance comparison, we selected one decentralised and one centralised method, namely [93] and [112]. The two works have been chosen as they are representative of two main classes of scheduling algorithms, suit the use-case considered in this thesis, aim at optimising productivity, and provide fault management capabilities. To apply the two approaches, the distributions associated with the task durations are assumed to be Gaussian and known a priori. Also, the mean duration is used in [112], as it is deterministic. Conversely, the our method does not require such information. Moreover, the work by Casalino et al. [19] has been considered to compare the two optimisation approaches. To do so, humans are not treated as uncontrollable agents as in the original work, but follows the instructions provided by the scheduler.

For each scheduling algorithm, 20 simulations have been performed, each one consisting of the assembly of 10 products. Figure 5.11 reports the average cycle times obtained by the different methods. Results from [93]

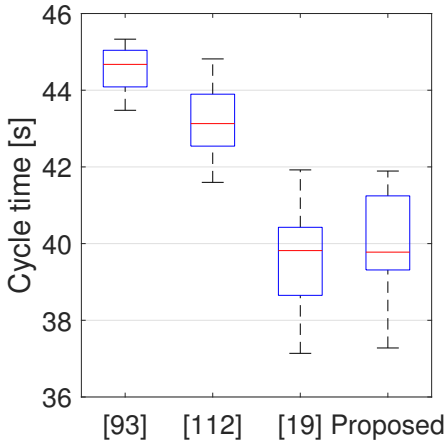


Figure 5.11: *Average cycle time of different scheduling algorithms.*

and [112] are worse, as they allow for less flexibility in the face of the variability of operation durations. On the other hand, the proposed approach and the one from [19] attain comparable performance with similar scheduling times. The latter result is important, as our approach implements a simpler exploration rule that is more suitable for large problems. A further note is that simulations have been run with stationary duration distributions and fixed production mix. Therefore, the performance of [93] and [112] in a real case is expected to be even worse. In particular, any time the production demand changes, the offline optimal plan must be recomputed, which takes a lot of time. Moreover, differently from the other works, the one proposed in this thesis is capable to optimally manage a wide range of abnormal behaviours of the system, described in Section 4.3, that allows for higher robustness and performance in real applications.

5.4 Experiments

The scheduler proposed in Section 5.1 and the communication paradigm described in Section 5.2 have been tested on a complex assembly task involving three agents with an experimental campaign involving 16 subjects. Results have been compared to those obtained by employing the algorithm in [93], taken as an example of state-of-the-art approach. In the following, Section 5.4.1 presents the experimental setup and protocol, while Section 5.4.2 discusses the results. Then, further experiments are included in Section 5.4.3 with the aim of assessing the system behaviour in the man-

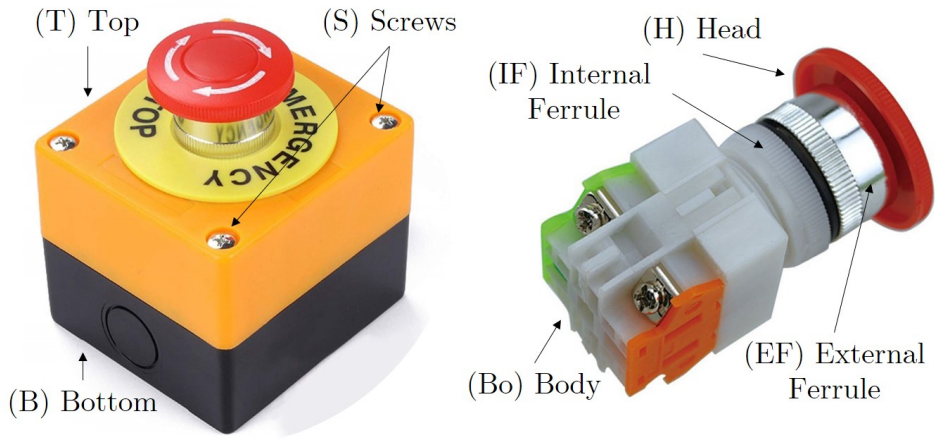


Figure 5.12: *Emergency button and its parts.*

agement of the error cases described in Section 4.3.

5.4.1 Experimental setup and protocol

The collaborative job consists in the assembly of an emergency button (Figure 5.12) executed by a human operator with the help of two robots (ABB dual-arm YUMI and IRB140). During the experiments, the human wears haptic bracelets on both arms and a vibrotactile ring with push-buttons on the non-dominant hand, as described in Section 5.2. The tactile interfaces and the robot controllers are connected to a CPU (Intel Core i7-8550U 1.80GHz, 8 GB RAM), where the scheduling algorithm is implemented as a C++ application. Figure 5.13 shows the complete experimental setup and the tactile displays.

The assembly of the emergency button consists of 5 steps:

1. Screw the *Internal Ferrule* onto the *Body*;
2. Position the *Top* and fix the *External Ferrule*;
3. Fasten the *Bottom* with the *Screws*, then fix the *Head*;
4. Pack the completed product inside a *Container Box*;
5. When the *Box* is full, change it for an empty one.

Some steps can be performed by more than one agent independently to allow for flexibility. Figure 5.14 shows the AAOG that defines all viable assembly plans for the product and the tasks of which they are composed.

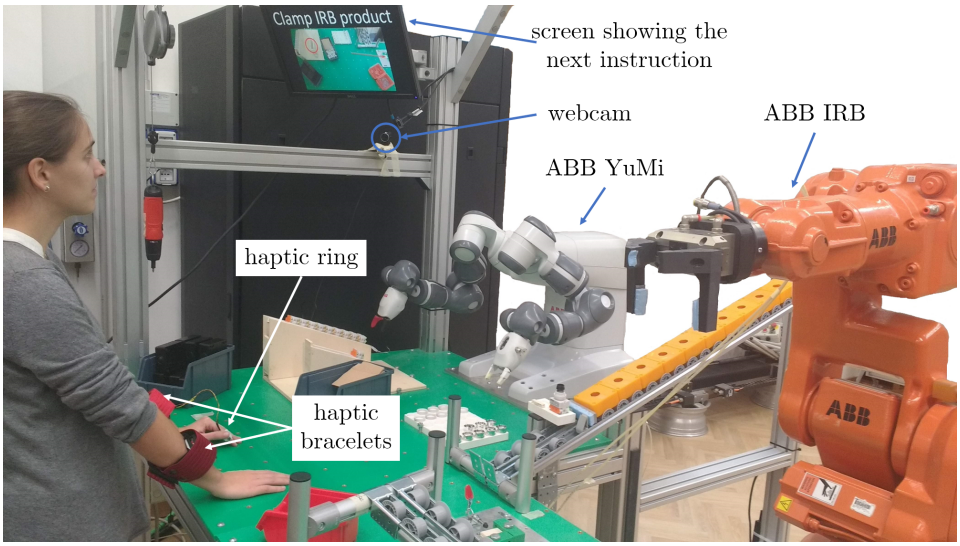


Figure 5.13: *Experimental setup.*

Edge colours specify the agent that performs the task: YuMi (red), IRB (blue) or human (green). For instance, step 2 is performed either by YuMi or IRB alone, or with a combination of a human and an IRB task. When more than one operation per agent is present in the same assembly step (e.g., o_8 and o_{12}), they differ for the origin (destination) buffer, from (to) which the WIP part is taken (placed). A constraint of mutual exclusion holds among step 2 tasks to avoid the simultaneous access to the buffer that stores the *Top* part, which is shared between the two robots. Moreover, a fixture is occupied at the beginning of o_{11} and released after completion of o_{10}, which is also used during o_9. The shared buffer and the fixture are treated as additional resources in the system.

Error cases have been partially included in the model according to Section 4.3. In particular, we considered robot failures in the execution of steps 2 and 4. Accordingly, 5 recovery actions have been included among the possible human tasks. All the other abnormal behaviours have been neglected, as they are not easily supported by the comparison algorithm. For the same reason, the effects of the recovery actions were always kept consistent with the default recovery. During the experiments, faults have been injected to consistently test the behaviour of the schedulers. Specifically, at the sixth step 2 task performed by YuMi, the gripper got stuck and was not able to grasp the product, the robot had to be reset by the operator, but the product was still available for processing. Conversely, IRB executed the

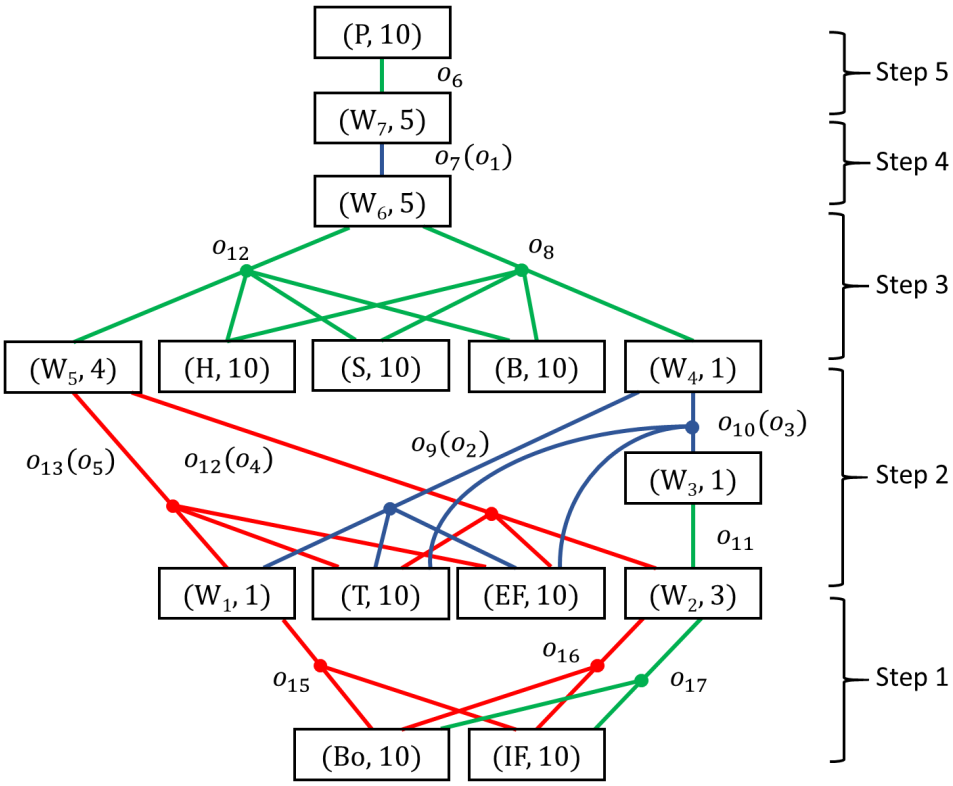


Figure 5.14: *Augmented AND/OR Graph of the job considered in the experiments. Product part names refer to Figure 5.12. Edge labels identify operations and the associated fault recovery actions (in brackets, if present). Colours specify the agent that performs the task: YuMi (red), IRB (blue) or human (green).*

Table 5.3: *List of robot operations.*

ID#	Robot	Operation description	Assembly step
15-16	YuMi	Start new product	1
12-13	YuMi	Assemble Top	2
9	IRB	Clamp product & assemble Top	2
10	IRB	Assemble Top	2
7	IRB	Store finished product in box	5

Table 5.4: *List of human operations and haptic signals.*

ID#	Operation description	Assembly step	Haptic signal
17	Start new product	1	3 left
11	Clamp product for IRB task	2	2 left
12	Complete product from YuMi	3	1 left
8	Complete product from IRB	3	1 right
6	Change full box	5	3 right
4-5	Recover failed YuMi	-	fast left
1-2-3	Recover failed IRB	-	fast right

fourth step 4 operation in a wrong way: the robot was still working, but the product needed human intervention to be completed. In total, 12 assembly operations compose 6 possible assembly sequences. Table 5.3 lists the robot tasks, while Table 5.4 details those available to the human and the associated haptic signals according to Section 5.2.2. The complete TPN that models the assembly process results in 50 places and 34 transitions.

We tested three different conditions: scheduling with [93] and visual feedback (S), proposed scheduling with visual feedback (V), and proposed scheduling with visual and haptic feedback (H). In all three conditions, the next operation for the human was displayed on a screen above the workbench (see Figure 5.13). In the H condition, participants also received tactile feedback according to Table 5.4. The association between haptic signals and operations was learnt by the user before testing the H condition. Participants were instructed to press the button on the wearable ring upon completion of each operation to signal task completion.

Twelve volunteers (3 females and 9 males, age range 23-33) participated in our study. Informed consent was obtained from all of them and the experimental evaluation protocol followed the Declaration of Helsinki. Participants did not perceive any payment and were able to leave the experiment at any moment. None of them had prior experience with the scheduling algorithms, nor with the selected assembly task, nor with the proposed use of haptic bracelets. During a preliminary *training phase*, volunteers were taught how to complete the different human operations, tried each of them twice, and performed a complete trial (10 products) to get comfortable with all tasks. Then, in the *test phase*, we divided the participants into 3 groups of 4. Each group tested a pair of conditions in different orders. For example, four participants tested conditions S and V, two in the order SV and two in the order VS. The same approach was applied to the groups that tested S and H, and V and H. Thus, each participant performed two complete ex-

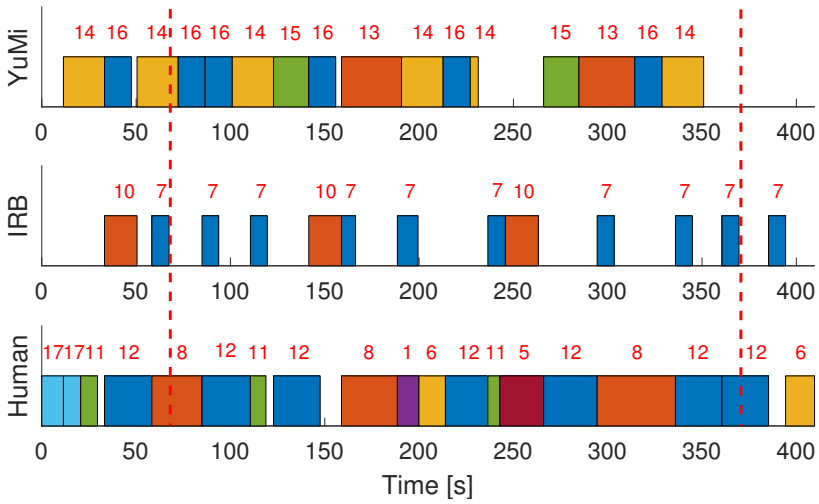


Figure 5.15: *Example of complete experimental trial. Vertical dashed lines indicate the interval considered for cycle time evaluation.*

perimental trials. A single trial consisted of the assembly of 10 emergency buttons and participants were asked to finish as quickly as possible. The average duration of a trial was about 7 min. Figure 5.15 reports an example of the complete execution of one trial.

5.4.2 Results and discussion

Comparison between scheduling approaches

Firstly, we analysed the experimental results based on the cycle time, measured from the end of the first product assembly to the end of the 9$^{\text{th}}$ product assembly (see Figure 5.15). The initial and final parts of the experiments have been neglected to remove transient behaviour. Figure 5.16a shows the measured values for the cycle time for the three conditions, where it is apparent that the proposed scheduling algorithm (V and H conditions) attains better performance than the one employed in S. For instance, the average cycle time in the V condition, which shares the same communication interface with the S condition, decreases by 15.6% ($p < 0.0002$, Wilcoxon test), from 40.6 s to 34.3 s.

The improvement is related to two main properties of the proposed scheduler: it adapts to the actual duration of human tasks and it optimises the management of the fault recovery actions, which can be delayed to avoid a complete stop of the production, giving priority, e.g., to the prepa-

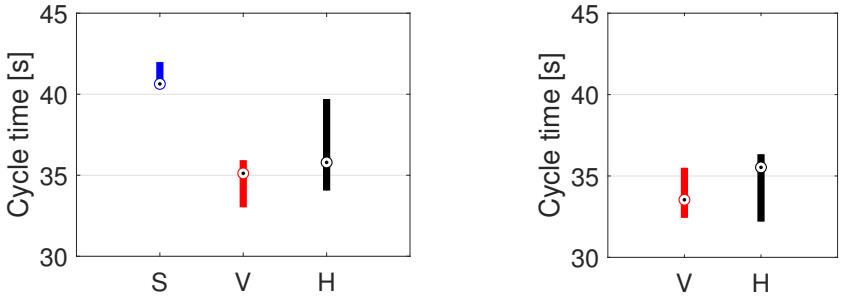
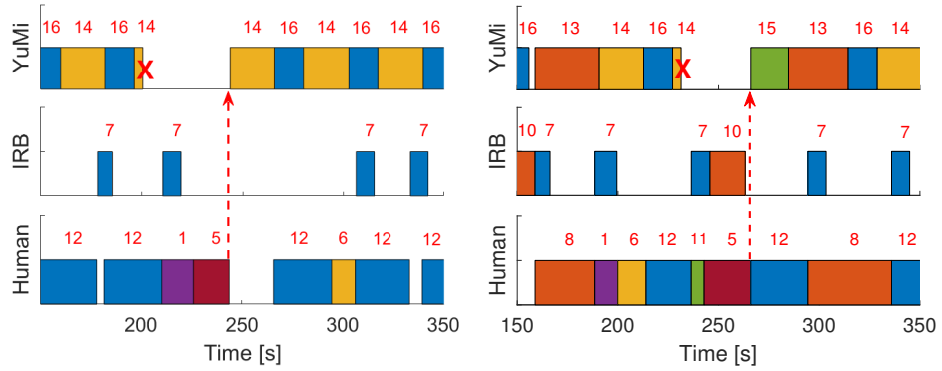


Figure 5.16: *Cycle time obtained for the different conditions (S, V and H) with 12 (a) and 16 (b) participants. Boxplots show median and quartiles.*

ration of a WIP part for another agent before restoring the failed one. An example of the latter situation is shown in Figures 5.17a and 5.17b, which compare two typical cases for the S and V conditions, respectively. With the state-of-the-art scheduler, the human recovery action (o_5) is planned as soon as possible after the failure of task o_{14} (at $t \approx 200$ s). However, this leads to a complete stop in the production and long idle times of all three agents. Conversely, the proposed scheduler inserts task 11 in the human plan before the recovery action to allow IRB to continue the assembly of a product, which is then ready for the human task o_{12}. In this way, the sum of the idle times of all resources is minimised.

On the other hand, the comparison between conditions V and H is less trivial. The reason is twofold. First, cycle time data show high variability in the H case. This can be due either to the use of haptic interfaces or to the small amount of available data. Second, as the difference between V and H conditions lies in a change of interface, it is critical to understand how the two approaches influence not only the performance, but also the attitude of the operator. To gather more data for the comparison between the interfaces, the experimental campaign was extended to other 4 novice volunteers (different form those that performed the first 12 tests), for a total of 16. Two participants tried the pair VH and the other two the pair HV. Figure 5.16b shows the cycle times obtained with conditions V and H considering all 16 participants. One can notice that the large dispersion displayed in Figure 5.16a is no longer present in the results obtained for the H condition and the performance in the two conditions is comparable.

Two other aspects were additionally taken into account, which are discussed in the following: the frequency with which the experimenter di-



(a) **(b)**

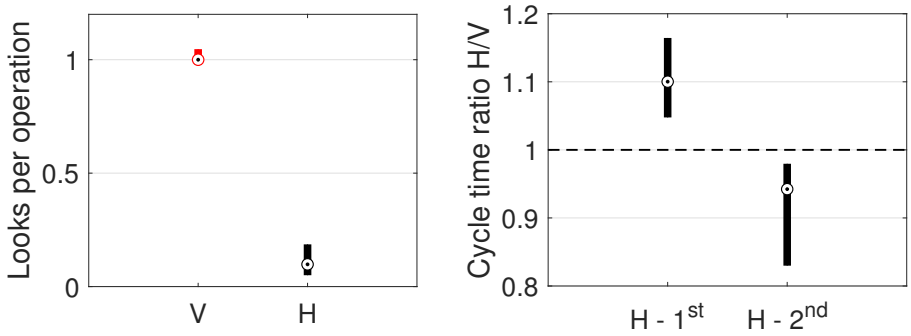
Figure 5.17: *Portions of plans from experiments showing how the two schedulers, [93] (a) and proposed (b), handle a failure of YuMi (red crosses). Arrows mark when the robot is reset. The proposed scheduler minimises the overall idle time by delaying the recovery action.*

rected his/her gaze towards the screen, and the users' subjective impressions on the collaborative task. The former was measured from videos acquired by a webcam mounted below the screen (see Figure 5.13). The latter was assessed through questionnaires filled in by volunteers just after their trials.

The role of tactile communication

Figure 5.18a reports the average number of looks the operators gave to the screen per operation. When only the visual interface is present (V case), the average number of looks remains constant, slightly above 1. This is reasonable since the operator had to look at the screen at least once per operation, to know which task to perform. The extra gazes mostly account for occasional double checks. Conversely, when the tactile instructions are introduced, experimenters consistently gave few looks to the screen, with a median of 0.1 looks per operation that constitute a significant reduction over the V case ($p = 0.0010$). Residual looks concentrate during idle time periods, between the end of a task and the reception of the next haptic command, or occasionally to check the received haptic command.

Figure 5.18b reports the ratio between the cycle time achieved in condition H and the one obtained in condition V by the same subject. Data show a significant reduction in the H-to-V cycle time ratios between the subjects that tested the H condition first and those that tested it as the last trial



(a) (b)

Figure 5.18: *(a) Average number of looks per operation for V and H conditions and (b) cycle time variation from V to H when the H condition was tested in the first (left) or second (right) trial.*

Table 5.5: *Questionnaires on the two feedback modalities answered by the 12 subjects (7 levels linear scale).*

Q1:	It is easy to use.
Q2:	Using it is effortless.
Q3:	I don't notice any inconsistencies as I use it.
Q4:	I can recover from mistakes quickly and easily.
Q5:	I can use it successfully every time.
Q6:	I learned to use it quickly.
Q7:	I easily remember how to use it.
Q8:	I am satisfied with it.
Q9:	The collaboration proceeded smoothly.
Q10:	I complete the operation more quickly when using it.

($p = 0.0286$). Instead, the median cycle time in the V condition does not change significantly between the two groups ($p = 0.4857$). Therefore, the level of expertise of the operator influences more the performance attained using haptic interfaces than those obtained with visual feedback only. Also, more expert users are able to improve results over the V case, with a median reduction of the cycle time of 5.8% from V to H conditions.

To obtain information on the participants' subjective opinions about the two feedback modalities, we relied on a questionnaire with statements taken in part from the USE Questionnaire [94]. The survey was formulated as a linear scale going from 1 (only visual feedback) to 7 (visual plus tactile feedback). Questions are reported in Table 5.5, whereas the distribution of

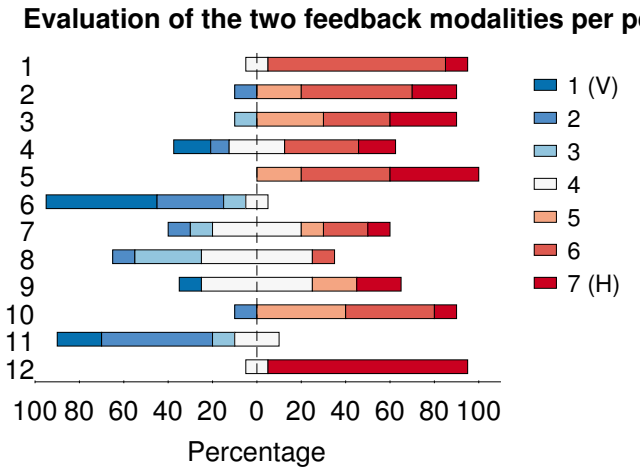


Figure 5.19: *Answers of the 12 subjects that tested the H condition to the questions on feedback modalities (Table 5.5) according to a linear scale going from 1 (only visual) to 7 (visual+tactile).*

the replies of each subject is shown in Figure 5.19. By adding the ratings given to all questions, ranging from a minimum score of 10 to a maximum of 70, we distinguished between two attitudes: preference for the V condition (score 10–40) and preference for the H condition (score 41–70). Although some people preferred the V condition (subjects $\#6, 8, 11$), there was an overall preference for the H condition, as 75% of the subjects appreciated the use of haptic feedback. The relation between the order of execution and the performance has already been discussed, and we noticed that it might have also influenced the acceptance of condition H, as the two most negative evaluations came from subjects that tested the haptic interface in the first trial ($\#6, 11$). Overall, results show that the haptic interface has been preferred and effectively used by most of the participants, despite the complexity of the task and the variety of commands conveyed through the vibrotactile paradigm. In particular, subjects that tested the H case as the second trial were able to exploit the haptic interfaces to increase performance, showing the crucial role of operators' confidence with the task.

5.4.3 Error management

Further experiments have been carried out focusing on the management of the error cases described in Section 4.3. To do so, we referred to a simplified version of the use-case described in Section 5.4.1, whose AAOG is reported in Figure 5.20. Specifically, only the human operator and the

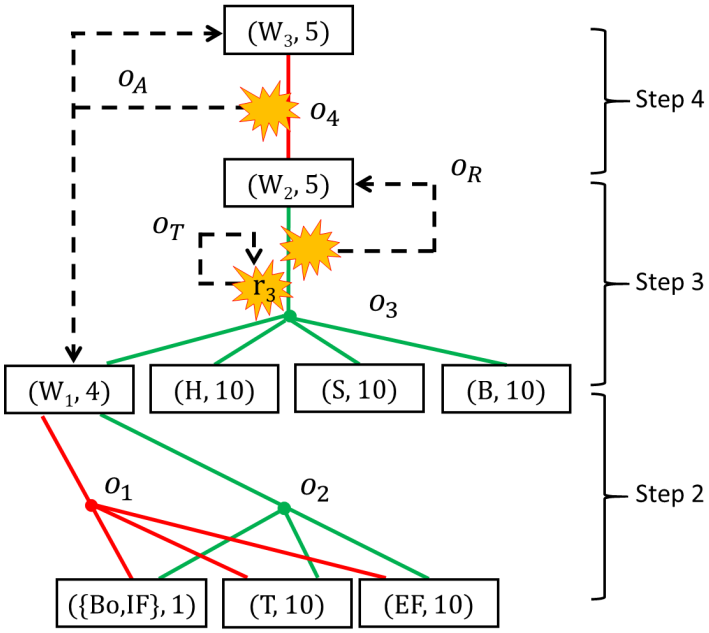


Figure 5.20: *Augmented AND/OR Graph of the job considered to test error cases. Product part names refer to Figure 5.12. Arc labels identify operations that are performed by either YuMi (red) or human (green). Dashed arrows mark recoveries from error cases.*

YuMi robot work in the cell to perform steps 2-3-4 of the emergency button assembly. Both agents can execute step 2, while the human is in charge of completing step 3 with the help of a tool (an electric screwdriver, modelled as a non-necessary resource r_3). The robot performs step 4. The monitoring unit that tracks the human activity is in charge of determining when new products are placed or removed from the intermediate buffers (W_1, W_2, W_3), and when the tool r_3 is picked up and released. This is done using an RGB-D camera to track the positions of the operator's hands in real-time in relation to virtual spheres that mark the zones of interest in the workspace.

In the experiments, we tested 4 error cases, for which we compared the proposed approach with a relevant baseline. To gather experimental data, we performed 24 trials, each one consisting of the assembly of 10 products. Two failures per trial have been injected in the system so that they do not influence one another, i.e. the transient after the first failure expires well before the occurrence of the second fault. Overall, each management strategy for each abnormal behaviour of the system was tested 6 times. The chosen error cases and the associated recovery actions are depicted in the AAOG in Figure 5.20 and better detailed in the following.

As far as human errors are concerned, the scheduler plays a little role in their management. Instead, the monitoring unit has the crucial role of detecting and identifying which operation the operator is performing and whether he/she makes some error during the activity. Then, to resume normal system behaviour a simple correction in the PN marking or the notification of the error to the human is needed. In Part I, monitoring algorithms that can provide the required information to the control system have been already proposed and tested. Therefore, human errors were not included in the present experimental campaign.

Robot faults

Referring again to Section 4.3, three types of robot faults have been included in the list of abnormal system behaviours. Nevertheless, they are managed fundamentally in the same way by scheduling an appropriate recovery action. Moreover, the case when the robot fails and becomes unavailable has been already presented in Figure 5.17, which showed how the proposed scheduler was capable to optimally schedule the recovery action. For this reason, we decided to select only one case to be tested in the present experiments, namely the one where the robot can continue working, but the product has to be recovered.

Specifically, operation o_4 consists of the quality control and packaging of the completed product. If the automatic quality control fails, the robot signals the error to the system, which calls for the final decision of the operator, i.e. the recovery action (indicated as o_A in the AAOG). Two cases have been tested:

1. The fault was a false positive and the product is compliant with quality standards. In this case, the human places the part in the final box (W_3) and concludes the recovery action. This alternative is taken as the default recovery action for the scheduling algorithm.

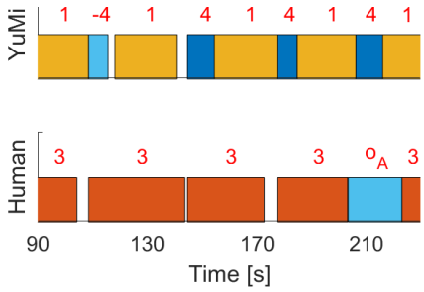
2. The product is truly non-compliant with quality standards. In this case, the human partially disassembles the product into a compliant WIP W_1, places it in the buffer, and concludes the recovery action.

The baseline strategy plans the recovery action as soon as possible and the human can only either accept or discard the product. This situation models the standard industrial practice that does not rely on real-time monitoring of the workspace and the operator's movements. As for the first case, the difference between the proposed and the baseline strategy lies in the freedom of the former in determining the optimal moment to schedule the recovery

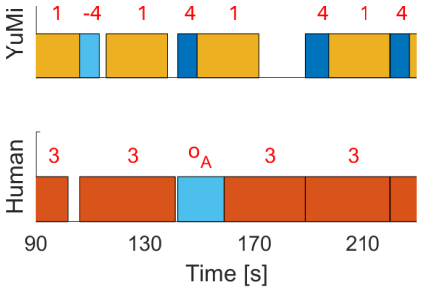
action. Figure 5.21a shows how the recovery o_A is delayed and not performed right after the failed operation, differently from what happens in Figure 5.21b. Similar to what is reported in Figure 5.17 for other types of robot faults, the smart management of failures reduces the total agents' idle time. In the experiments, we observed an average idle time of 15.8 s with the baseline approach, which drops to 10.1 s with the proposed one, reducing by 36.1%. As for the second case, workspace monitoring widens the alternatives available to the operator, who can not only accept or discard the product, but also re-inject a partially disassembled product in the production flow. As a result, the production of the non compliant item does not start from the beginning, but resumes from step 3 (Figure 5.21c). The actual outcome of the recovery action differs from the default one used while predicting the future evolution of the system. Nevertheless, the monitoring unit can correctly track the process state and the scheduler adapts to the new condition. On the other hand, after the baseline recovery the product is discarded and a new one starts (Figure 5.21d). Thus, an additional step 2 operation (o_1 or o_2) must find a place in the schedule with respect to our approach, which decreases productivity. In the reported example, the robot idle time present in Figure 5.21c is replaced by an additional o_1. However, the duration of the robot operation is higher than the idle interval. Besides, the excess delay eventually leads to robot idle time around 175 s. In general, the performance gap between the two strategies is more pronounced the further down the defect occurs in the assembly sequence, as it would be necessary to repeat more operations.

Other failure cases

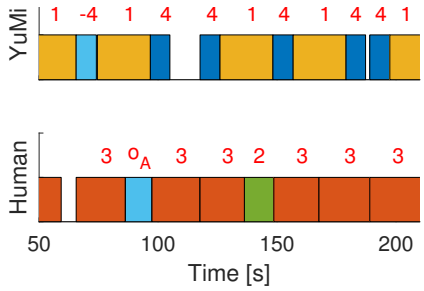
The possibility to find a defective product that must be corrected before continuing has been considered as the third error case. When this happens, the human signals the start of the immediate recovery action (indicated as o_R in the AAOG) through one of the buttons on the haptic interface described in Section 5.2. Then, he/she substitutes the defective part, proceeds with o_3, and places the obtained WIP in the buffer W_2. In the baseline case, the operator follows the same steps but cannot notify the system, which is thus unaware of the delay in the completion of o_3. Figures 5.21e and 5.21f show an example of recovery according to the proposed and baseline methods, respectively. Although the duration of the human activity is the same in both cases, it is apparent that informing the scheduler of the expected delay needed for recovery leads to better performance. In particular, the robot can start o_1 in parallel to the operator, instead of waiting for the end of the human task to perform o_4. Overall, the proposed strategy eliminated



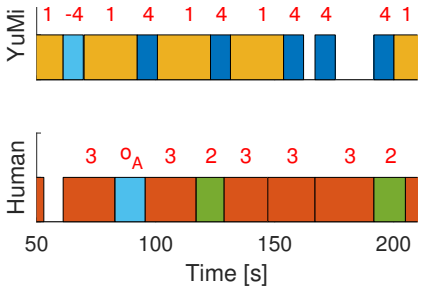
(a) *Robot fault, compliant product.*



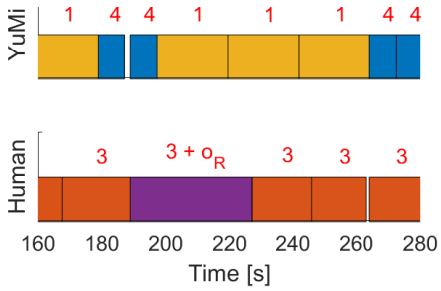
(b) *Robot fault, compliant product.*



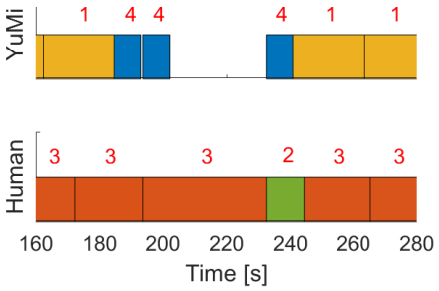
(c) *Robot fault, non compliant product.*



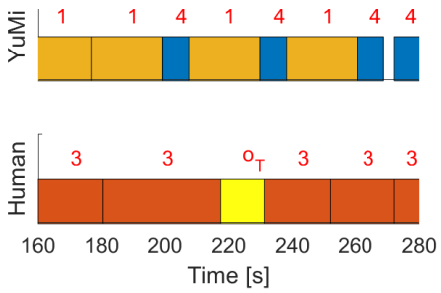
(d) *Robot fault, non compliant product.*



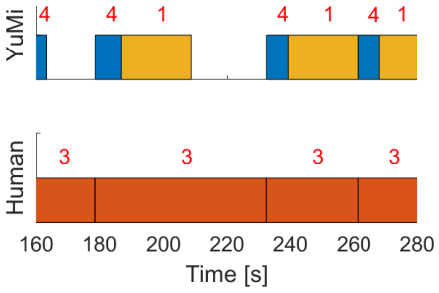
(e) *Defective product.*



(f) *Defective product.*



(g) *Tool failure.*



(h) *Tool failure.*

Figure 5.21: *Examples of management of different error cases with the proposed approach (left column) and the baseline strategy (right column).*

the average idle time of the robot from 26.5 s to 5.0 s over the baseline case.

The last notable case included in the trials is the unavailability of non-necessary resources, such as the electrical tool identified as r_3 in Figure 5.20. Conversely, the recovery from failures of necessary resources is equivalent to the management of robot faults. In particular, we simulate the fact that the electric screwdriver is out of batteries, which must be substituted. Still, the operator can perform o_3 with a manual screwdriver. In the proposed strategy, the operator signals the tool fault and the scheduling algorithm plans the recovery action (indicated as o_T in the AAOG). Instead, following the baseline strategy the operator changes the batteries as he/she becomes aware of the problem without notifying the system. Similar to the case of a defective product, prompt information on the occurrence of the failure allows the correct replanning of robot operations to minimise idle time, as it is shown in Figure 5.21g. During the experiments, we observed a strong reduction from 13.4 s to 1.4 s over the baseline recovery strategy, which is exemplified by Figure 5.21h. Moreover, with the proposed approach the scheduler is free to delay the recovery of the non-necessary tool to the optimal time, which is not possible with the baseline method. Similar to what is shown in Figure 5.17 in response to robot faults, delaying the recovery action can prevent the total stop of production.

Concluding remarks

Overall, experimental results show that intelligent fault detection and recovery action management increase the performance of task scheduling in response to a wide range of abnormal system behaviours. Benefit mainly comes in the form of reduced idle time of the agents and the opportunity to avoid complete stops of the production line to allow for offline manual recovery. The results obtained are only possible through the tight integration of dynamic scheduling and real-time monitoring of human activity. Together, they can correctly track the state of the process and adapt the production plans to react to unforeseen events. In particular, real-time monitoring allows the operator to exploit his/her superior cognitive skills and choose the best recovery action for the specific situation, which means that the system can recover in multiple ways from the same type of failure condition. This allows coping with the high complexity of the manufacturing process, which makes the a priori definition of all failure cases impossible.

CHAPTER 6

Dynamic scheduling of collaborative kitting operations

As already widely discussed, present-day manufacturing is characterised by an increasing level of product variability and small size lots. In this scenario, the management of the logistic processes becomes as important as production control. In particular, Just-In-Time material supply is needed to achieve the required levels of productivity and flexibility, with the additional advantage of limiting floor space utilization and storage costs. One of the most common approaches for part feeding in assembly lines is kitting, which consists in grouping separate items together to be supplied as one unit to the work stations.

Currently, kitting is a time-consuming task, which is mostly done manually by human operators. In past years, most research efforts focused on the design of an efficient and cost-effective part feeding process [40,63,92]. Also, strategies to improve kit quality and ease kit preparation have been studied and commercialised [37, 121]. However, the physical strain of workers must also be considered. Picking objects from the warehouse implies a great repetitiveness in arm motion and postures characterised

by arms that are adducted and elevated [26]. Moreover, the weight of items may increase the physical strain and favour the development of work-related musculoskeletal disorders, which include all medical conditions that are caused or aggravated by work and the circumstances of its performance. These remain common issues that have been limiting improvements in recent years [34]. Therefore, taking into account the workers' health and welfare costs, it is mandatory to apply policies aimed at minimizing the risk of the insurgence of such musculoskeletal disorders.

Recently, there has been a growing interest in transitioning to automated kitting systems using robot manipulators. Design factors to implement an effective robotic system were discussed in [155], whereas [147] analysed the performance of different system configurations, and [16] studied the economic benefit of automated kitting over the manual one. Fully robotic kitting has been proposed, where mobile robots pick parts from the shelves and bring the kit to the assembly line [78, 149, 161]. However, research efforts mostly concentrate on overcoming technical limitations that still hold. For instance, the robust recognition and grasping of a large variety of objects, as well as the precise positioning of parts into kitting boxes, are complex tasks for a robot.

To mitigate these problems, hybrid kitting systems have been introduced, where both a human and a robot contribute to the task. [116] presented a mobile manipulator for kitting tasks, which can share the same space as the human. Similarly to [78], the main focus is on part recognition and placement into the box. Instead, [36] developed a system where the human operator retrieves the correct items from the shelves, while an assistant collaborative robot sorts items inside the kit. In [10], the authors investigated the optimal assignment of parts to either the robot or the operator, each working in their own kitting supermarket, so as to optimise the kit preparation time. More in general, research on task allocation for human-robot kitting has mainly focused on performance and safety, in line with works related the other applications of human-robot collaboration.

On the other hand, the ergonomic benefits of integrating robots into human tasks have been extensively studied for various industrial use cases, where the robot can be used to optimally position or manipulate the product on which the worker must perform his/her task [38, 119, 148, 169]. Ergonomics as a criterion to guide task planning has been considered, for instance, in [158], where multiple criteria are used to define an offline schedule for a hybrid human-robot assembly cell. Also, [35] described an approach to consider ergonomics and human movement capability in the optimal assembly sequence, while in [14] both the task allocation and the

REBA score	Risk level	Action required
1	Negligible	No action required
2-3	Low	Change may be needed
4-7	Medium	Further investigation, change soon
8-10	High	Investigate and implement change
11-12	Very high	Implement change immediately

Table 6.1: *REBA action level list. The highest the REBA score, the higher the ergonomic risk and the more urgent the corrective actions.*

motion planning problems are solved to optimise ergonomics in handover operations. Finally, [117] explores the trade-off between ergonomics and productivity by exploiting a hierarchical task model to quantify physical strain in assembly operations.

In this Chapter, we investigate the convenience of introducing a collaborative robot in the kitting process to share the workload with the human operator. Specifically, the work is distributed in a way that reduces his/her physical strain so to alleviate the risk of musculoskeletal disorders resulting from excessive fatigue. Besides, sharing work between two agents increases the throughput. To do so, an online scheduling algorithm is proposed to guide the picking operations of the human and the robot. The system architecture follows the same principles explained in Chapter 4.1 and relies on the online solution of an optimization problem with a receding horizon approach. Data on the execution of human actions are used to cope with the variability of human task duration and to ensure the coordination of the agents to prevent collisions. To the best of the author's knowledge, this is the first attempt to implement a fully collaborative kitting task, where human and robot dynamically coordinate to complete the same kit in a shared workspace.

The remainder of the Chapter is organised as follows. Section 6.1 describes a way to quantify the operator's physical strain through ergonomic evaluation and to associate an ergonomic measure to each picking action. Section 6.2 details the scheduling algorithm, formalised as a Mixed-Integer Linear Programming optimization, whose performance is experimentally tested and discussed in Section 6.3.

6.1 Ergonomic measurement

The manual kitting process is characterised by the lifting of material and repeated arm movements, which comprise adduction and elevation. Conse-

quently, the risks of work-related musculoskeletal disorders must be taken into account and minimised. This section proposes a method to associate an ergonomic measure to each picking action that composes the kitting process. The ergonomic score is then used by the scheduling algorithm to minimise the operator's strain, as detailed in Section 6.2.

Ergonomic factors are increasingly considered when designing kitting stations [26, 154]. However, the focus is usually to seek a reduction in the physical exertion via a proper design of the workspace. Conversely, no action is taken while the operator is performing the task. To enable an online optimization of ergonomics, the real-time evaluation of the operator's strain is needed. Direct methods for effort measurement require the collection of data from sensors attached to the worker's body, such as electromyographs, which are typically intrusive and expensive. On the other hand, an estimate of physical strain can be obtained with observational methods, which are easy to use and widely applied in industry [8]. In common industrial practice, posture data are collected through the subjective observation of the workers and the estimation of body-joint angles from videos. However, the recent spread of low-cost RGB-D cameras, such as the Microsoft Kinect, allows collecting postural data in real-time and enables online assessment [31, 54, 100, 122]. Also, deep learning techniques allow for automatic postural assessment from simple video streams [115]. Among the observational methods suitable for automatic assessment, the most common are RULA (Rapid Upper Limb Assessment [104]) and REBA (Rapid Entire Body Assessment [61]), which return a score based on the angles of body joints and object weights. Since the kitting process may involve the motion of the whole body, the latter is more relevant to the application.

The REBA method consists in the evaluation of an assessment grid composed of two sections. Section A considers the position of the trunk, legs, and neck, plus a correction that accounts for the force load acting on the limbs; section B considers the posture of the two arms by looking at the positions of the lower arm, upper arm, and wrist. A score is computed for each section and then combined to obtain the final REBA value, which ranges between 1 and 12. An action level list, shown in Table 6.1, indicates the severity of the ergonomic risk and whether intervention is required to reduce the risk of injury associated with the task.

The REBA score function C can be defined as follows:

$$C(\boldsymbol{\theta}, \boldsymbol{\phi}, w) = C(A'(\boldsymbol{\theta}) + A''(w), B(\boldsymbol{\phi})) \tag{6.1}$$

where A and B are the values obtained from the two sections. The score A can be seen as the sum of two functions A' and A'': one depending on the

joint angles $\boldsymbol{\theta}$ of trunk, legs and neck, the other depending on the weight w of the object being picked. Instead, the score B is a function of the joint angles ϕ of the arm that performs the picking operation. Overall, 23 joint angles are required to evaluate the REBA score. For this reason, automatic measurements obtained with vision sensors can easily degrade due to self-occlusions of the operator's body parts or occlusions with objects in the scene. Thus, it is difficult to obtain a reliable ergonomic rating online. A solution to this problem comes by noting that the kitting process consists in taking objects from a rack and that the number of available operations is limited by the number of different objects in the warehouse. Also, the position of the items in the warehouse does not change in time. So, it is possible to associate each picking action with a constant ergonomic score to be used by the scheduler. For the same person, the body joints angles will depend on the object position, whereas two different people in height will take different postures to reach the same object. Therefore, the function C can be rewritten in terms of the object and worker heights, h_o and h_w, respectively. The REBA score can be thus expressed as:

$$C(h_w, h_o, w) = C(A'(h_w, h_o) + A''(w), B(h_w, h_o))$$

In the expression, $A''(w)$ is the same as in equation (6.1), while $A'(h_w, h_o)$ and $B(h_w, h_o)$ can be identified collecting data for different object positions and workers' heights.

To do so, a Microsoft Kinect has been used to track skeletal points of different people (both right-handed and left-handed) performing picking tasks at different heights. Joint angles $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ have been evaluated at the moment of picking. Results show three main areas: a central comfort zone and two high-effort zones to reach very high or very low locations. Boundaries between zones served as a guideline to find a transformation to align data from all people. To identify the ergonomic function $\hat{C}$, the average REBA score for each object height has been computed from the aligned data. Once $\hat{C}$ is available, one can exploit the inverse transformation to adapt the single model to any worker's height and to any configuration of the items in the shelves. Figure 6.1 shows two example comparisons between empirical REBA scores recorded during the experiments and those obtained using $\hat{C}$.

6.2 Dynamic scheduling algorithm

The allocation and scheduling of picking tasks between the human and the robot are considered as a multi-agent coordination problem with temporal and spatial constraints, which is formulated as a Mixed-Integer Linear

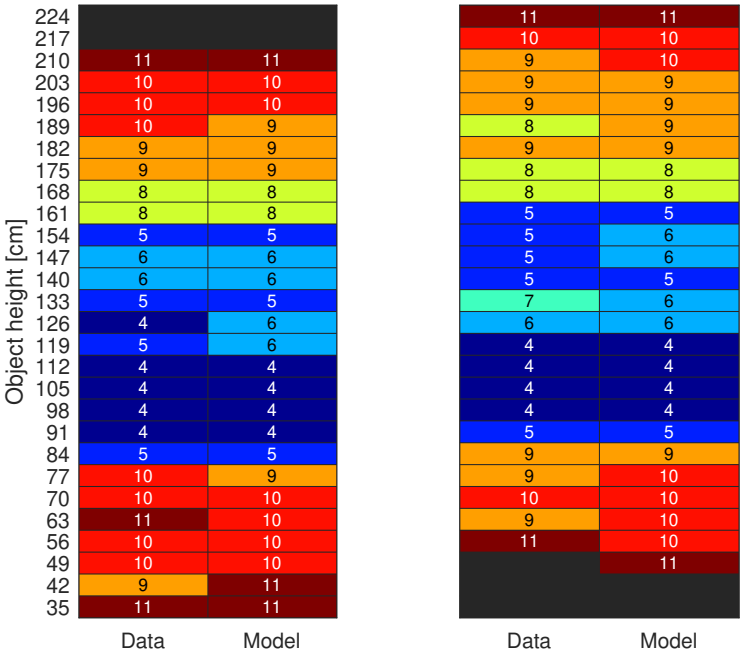


Figure 6.1: *Comparison of experimental and identified REBA scores for different object heights obtained with $h_w = 178$ cm (left) and $h_w = 187$ cm (right).*

Programming (MILP) optimization. Although this approach does not scale well into large problems due to its exponential computational complexity, the minimal team composition and the limited kit size allow for real-time solution. This feature is paramount, as online rescheduling allows the plan to adjust to the variability of human behaviour. Conversely, static scheduling would lead to high idle times of the agents, as the robot would wait for the operator in case human activity was late and vice-versa when the worker finished his task faster than expected.

The goal of the MILP scheduler is to allocate the objects that compose the kit to individual agents such that the kit preparation time and the human physical strain are minimised. The ergonomic evaluation described in Section 6.1 is exploited to quantify the latter. In addition, the scheduler guarantees the satisfaction of job assignment rules and temporal constraints, as well as ensuring the spatial coordination between the agents to avoid collisions. The MILP input consists of information on the kit to be made and the current state of the agents. The output is the set of items assigned to the human and to the robot and the sequence with which they must be picked.

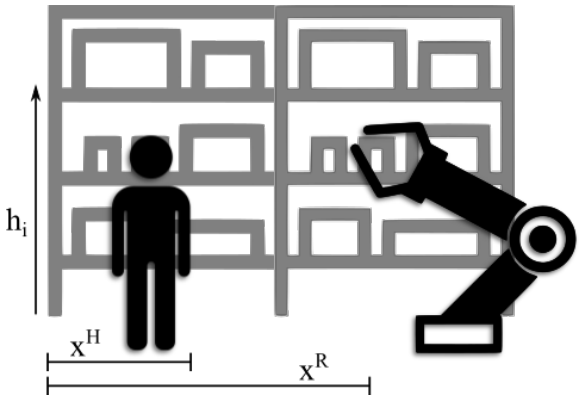


Figure 6.2: *Prototypical workspace layout of the kitting process.*

6.2.1 MILP definition

In this section, we formalise the MILP optimization problem referring to the prototypical layout shown in Figure 6.2. The kitting process is performed by a human operator and an industrial robot that pick items from a rack. Without loss of generality, the human is supposed to stand on the left side of the shelves, while the robot is positioned on the right. The origin of the coordinate frame used to express the positions of the objects and the agents is placed on the bottom-left corner of the rack.

The optimization problem receives as input the set of objects that compose the next kit $K = \{i\}$ to be prepared and information on the picking tasks and the current process state. More formally, each object is characterised by the following parameters:

- Its position in the rack, defined by the horizontal coordinate x_i and the vertical coordinate h_i;
- The expected duration d_i^H and d_i^R of the action of picking the object by the human and the robot, respectively;
- The value of the human strain associated with the picking task $s_i = \hat{C}(h_w, h_i, w_i)$, with w_i the weight of the object;
- A binary parameter $b_i \in \{0, 1\}$, which is equal to one if the robot can pick the object, zero otherwise.

The last parameter accounts for the reachability or manipulability limitations of the robot, while it is supposed that the human is able to pick all items. If this is not the case for a specific application, it is possible to model the operator's capabilities analogously.

The current state of the process is described by:

- A binary parameter $F \in \{0, 1\}$, equal to one if a new kit is starting, zero in case of the rescheduling of a kit in progress;
- The current horizontal positions with respect to the rack of the human x^H and the robot end-effector x^R;
- The estimated remaining time to completion of the ongoing operation of the human d_r^H and the robot d_r^R at the time when scheduling takes place, which is equal to zero if the agents are idle.

Also, the following quantities are considered:

- The time d_B needed to change the completed kit box with an empty one before starting a new kit;
- The upper bound T_{max} on the completion time for the current kit, computed as the sum of the longest duration for each picking task and the longest remaining time among the two agents:

$$T_{max} = \sum_{i \in K} \max(d_i^H, d_i^R) + \max(d_r^H, d_r^R)$$

- The upper bound S_{max} on the worker's physical strain for the current kit, obtained in case all objects are assigned to the human:

$$S_{max} = \sum_{i \in K} s_i$$

The scheduler attempts to solve for the following optimization variables:

- $H_i \in \{0, 1\},\ \forall i \in K$, equal to one if the object i is assigned to the human, zero if it is assigned to the robot;
- $t_i > 0,\ \forall i \in K$, the time when the picking action of item i starts;
- $T_K > 0$, the completion time of the kit K (makespan);
- $S_K > 0$, the total physical strain for the human to perform the scheduled tasks.

The cost function f to be minimised is a trade-off between performance and ergonomics:

$$\min f = \alpha \cdot \frac{T_K}{T_{max}} + (1 - \alpha) \cdot \frac{S_K}{S_{max}} \tag{6.2}$$

where $\alpha \in [0, 1]$ is a design parameter that represents the importance of minimizing makespan over strain.

Finally, the problem constraints are described in logic form in the following, from which a linear formulation can easily be obtained. The constraints ensure the coherence and feasibility of the schedule, the coordination of the agents to avoid collisions, and a limitation to the human idle time.

- The kit is completed when all objects have been taken:

$$T_K \geq t_i + H_i d_i^H + (1 - H_i) d_i^R \qquad \forall i \in K$$

- The total physical strain is equal to the sum of the effort score associated with the objects assigned to the human:

$$S_K = \sum_{i \in K} H_i s_i$$

- Objects that the robot cannot pick are assigned to the human:

$$b_i = 0 \implies H_i = 1 \qquad \forall i \in K$$

- If two objects are assigned to the same agent, one must end before the other starts:

$$H_i \wedge H_j \Rightarrow (t_j \geq t_i + d_i^H) \vee (t_i \geq t_j + d_j^H) \qquad \forall i, j \in K,\ i \neq j$$
$$\neg H_i \wedge \neg H_j \Rightarrow (t_j \geq t_i + d_i^R) \vee (t_i \geq t_j + d_j^R) \qquad \forall i, j \in K,\ i \neq j$$

- Both agents must start all picking actions after completing the ongoing one;

$$H_i \implies t_i \geq d_r^H \qquad \forall i \in K$$
$$\neg H_i \implies t_i \geq d_r^R \qquad \forall i \in K$$

- When a new kit starts, the first human task is to change the completed box with an empty one, while the robot can start a new task provided that it will end after d_B:

$$F \wedge H_i \implies t_i \geq d_B \qquad \forall i \in K$$
$$F \wedge \neg H_i \implies t_i + d_i^R \geq d_B \qquad \forall i \in K$$

  The second constraint allows the robot to start the new kit early, reducing the idle time between two kits. At the same time, it also ensures that the new box is ready to place the first item.

To avoid the two agents crossing each other and to avoid collisions, the scheduler enforces a moving separating line that dynamically changes position during the kitting process. The human is constrained to remain to the left of the separating line, the robot end-effector to the right. Specifically:

- If two objects i and j are assigned to the robot and the human, respectively, and i is to the left of j, one of the tasks must end before the other starts:

$$\neg H_i \wedge H_j \wedge (x_i \leq x_j) \implies (t_j \geq t_i + d_i^R) \vee (t_i \geq t_j + d_j^H)$$
$$\forall i, j \in K,\ i \neq j$$

- All the tasks allocated to one agent and related to an object in the area initially occupied by the other one must start after that the other agent has completed its ongoing operation:

$$H_i \wedge (x_i \geq x^R) \implies t_i \geq d_r^R \qquad \forall i \in K$$
$$\neg H_i \wedge (x_i \leq x^H) \implies t_i \geq d_r^H \qquad \forall i \in K$$

Note that for a workspace where the agents' position with respect to the rack is reversed, the correct constraints are obtained by simply inverting the inequality signs between the horizontal coordinates.

Finally, a constraint to limit human idle time has been introduced. It imposes that all human operations must end before a given time upper bound, which is equal to the minimum time required to execute all the tasks allocated to the human multiplied by a coefficient $\beta \geq 1$ that determines the percentage of idle time allowed. That is:

$$H_i \implies t_i + d_i^H \leq \beta\Big[I + \sum_{j \in K} H_j d_j^H\Big] \qquad \forall i \in K$$
$$I = F d_B + \neg F(N d_r^H + \neg N d_r^R) \qquad N = \bigvee_{i \in K} H_i \wedge (t_i < d_r^R)$$

where I allows accounting for the correct completion time of the previous schedule.

6.2.2 Receding horizon scheduling

In Section 6.2.1 the single optimization problem has been described. In the following, the behaviour of the complete receding horizon scheduling routine, shown in Algorithm 4, is outlined for a set of kits, possibly unlimited and not known a priori.

Algorithm 4 Receding horizon scheduling approach.

```
P ← ADDNEWORDERS( )                                   ▷ Add incoming kits to pending list
while P ≠ ∅ do
    K ← SELECTNEXTKIT(P)                              ▷ Select the first pending kit
    P ← P ∖ K                                         ▷ Remove selected kit from pending
    plan ← SCHEDULE(K)                                ▷ Schedule picking operations
    while K ≠ ∅ do
        sent ← DISPATCH(plan)                         ▷ Send next commands to free agents
        plan ← plan ∖ sent                            ▷ Remove started operations from plan
        K ← K ∖ sent                                  ▷ Remove started operations from kit
        wait until next operation is completed
        if human is free then
            plan ← SCHEDULE(K)                        ▷ Update schedule
        else if robot is free and next operation unfeasible then   ▷ Prevent crossing
            plan ← SCHEDULE(K)
        end if
    end while
    P ← ADDNEWORDERS( )
end while
```

Initially, the available kitting orders are added to the pending queue P. Then, the algorithm selects the next kit to be prepared among the pending one according to a suitable logic, which usually depends on the specific application. Common choices could be to follow a FIFO, Earliest Deadline First, or other priority-based approaches. The optimization problem is solved to find a schedule for the first kit, in which all objects are assigned either to the human or to the robot. Consequently, the first commands are dispatched to the agents, which start their tasks. During the kit preparation, several rescheduling may occur to adapt the plan to the actual duration of concluded tasks. The schedule is updated each time one of the following events occurs:

1. The human operator finishes his/her task;

2. The robot should start its next task, but this would cause a crossing between the human and the robot.

In principle, the latter case is avoided by the MILP, which enforce a moving separating line between the agents at all time, but it may happen during the actual kit preparation if the human is late. At each rescheduling, the MILP considers a kit K composed of the unpicked objects and $F = 0$. The optimization problem is solved according to the current state of the process and the old schedule is updated with the new one. When a kit is finished,

Figure 6.3: *View of the experimental set-up.*

if new incoming orders are available, they are added to the queue. Then, the next pending kit is started. The cycle repeats until all orders have been fulfilled.

6.3 Experiments

The scheduling algorithm presented in this Chapter has been tested on the simulated industrial kitting station shown in Figure 6.3. A gravity rack stores different sized objects. Items whose weight influences the REBA score are labelled, whereas red crosses identify those the robot cannot pick. The former were placed randomly in the rack, the latter are the objects in the two leftmost columns that are outside the robot's reach. The robot is a Comau Smart Six industrial manipulator endowed with a vacuum gripper. The human interface is composed of a screen that shows the next instruction to the worker and a button used to signal the end of each action. The robot controller and the screen are connected via TCP/IP to an external computer that runs the scheduler.

In the following, Section 6.3.1 analyses the trade-off between makespan and ergonomics in the cost function and Section 6.3.2 presents the results of the experimental campaign.

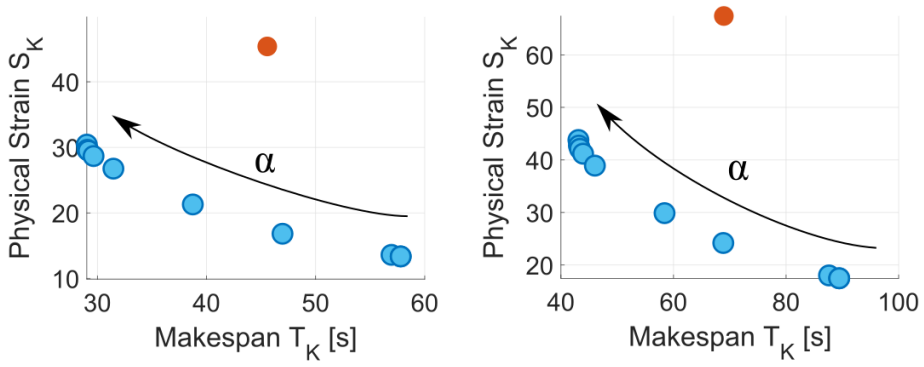


Figure 6.4: *Makespan-strain trade-off for kits composed of 6 (left) and 9 (right) random items. Red dots refer to the case of pure manual kitting, blue dots to human-robot collaboration for increasing values of α from 0 to 1.*

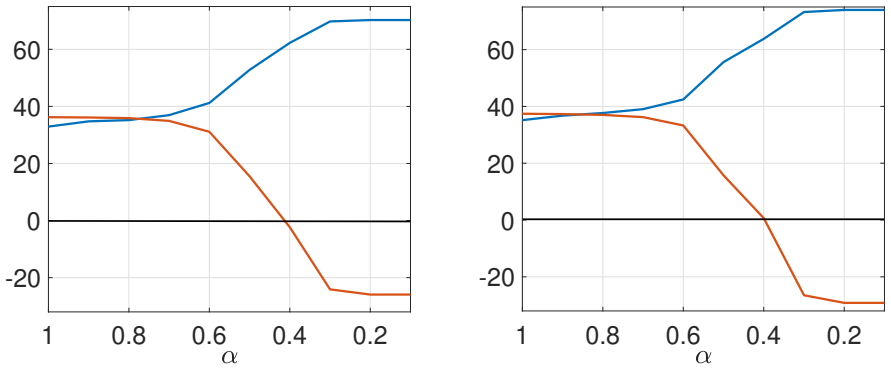


Figure 6.5: *Percentage reduction in makespan (red) and strain (blue) for kits composed of 6 (left) and 9 (right) random objects with respect to pure manual kitting.*

6.3.1 Makespan versus strain trade-off

Before evaluating the scheduler performance in the real set-up, a set of simulations has been performed to investigate how parameter α in the cost function (6.2) affects the output schedule in terms of makespan T_K and strain S_K. Specifically, the algorithm has been fed with random kits of different sizes, and we computed the optimal schedule varying α from 0 to 1 with step size of 0.1.

First of all, we noticed that the obtained results are independent of the kit size. Figures 6.4 and 6.5 report those related to kits composed of 6 and 9 objects as examples. Namely, Figure 6.4 depicts the average values of makespan and strain for varying values of α and compare them with

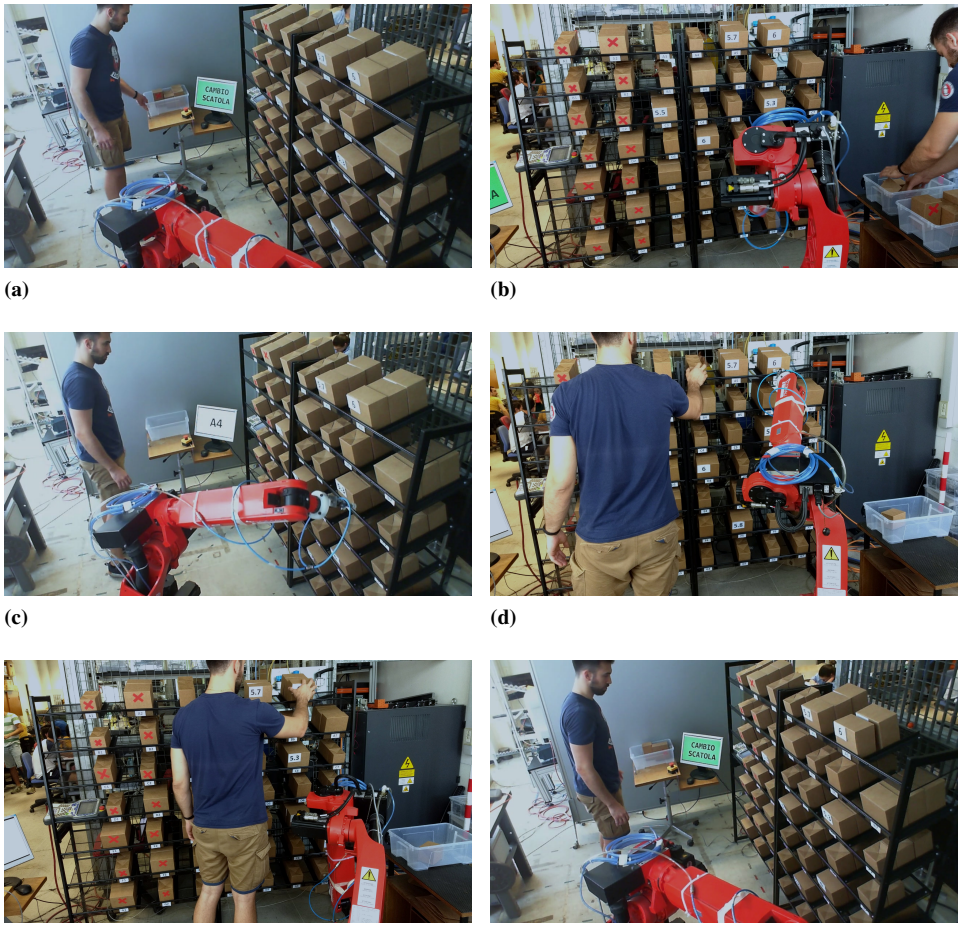
the values obtained in the case of pure manual kitting, where all tasks are performed by the human. To better compare the results associated with different sized kits, Figure 6.5 shows the average percentage reduction in makespan and ergonomics over pure manual kitting. The two quantities are roughly inversely proportional: as makespan increases, strain decreases. Pure manual kitting is the worst case for ergonomics, therefore, strain is reduced for any value of α. Instead, the time to complete a kit may worsen for low α. This happens because, in our experiments, the robot is in general slower than the human in performing the picking operations.

Since the behaviour of the trade-off is the same for all kit sizes, it is possible to choose a constant value of α that is valid throughout the entire kitting process, irrespective of the characteristic of the kit being prepared. From Figure 6.5 we can see that for values of α greater than 0.7 makespan and ergonomics attain similar and constant reduction. As α goes from 0.7 to 0.4 improvement in terms of completion time decreases, while ergonomics enhances. Then, for $\alpha < 0.4$ the makespan starts to get worse than the baseline case and there is a flattening of the two curves when no more objects can be assigned to the robot to further improve ergonomics. In the experimental campaign in Section 6.3.2, the value of α was kept fixed to 0.8, so that the scheduler tends to attain similar improvements for both makespan and ergonomics over the manual kitting case.

6.3.2 Experimental results

To asses the performance of the proposed scheduling algorithm, a first series of experiments were conducted with the dynamic scheduler presented in Section 6.2. Then, other tests have been performed using an offline scheduler, meaning that the plan was only computed once when the kit started and never updated during the kit preparation. This allows comparing results and assessing whether rescheduling performs better by absorbing disturbances due to the variability of human tasks. For both scheduling approaches, three cycles have been performed, each consisting of several consecutive kit preparations, so that for each cycle at least 100 objects were picked. Specifically, kits were randomly generated with a size between 6 and 9 objects.

To give better insight into the execution of the experiments, Figure 6.6 shows frames taken from a video during which the human and the robot assemble one kit. In Figure 6.6a the operator is instructed to change the completed kit before starting the new one. Meanwhile, the robot starts its first picking action for the new kit (Figure 6.6b). When the human has posi-



(a) (b) (c) (d) (e) (f)

Figure 6.6: *Screenshots taken from a video of the experiments.*

tioned the empty box and pressed the button to communicate the end of the task, he/she receives information on the next object to pick (Figure 6.6c). Then, the two agents start to work in parallel coordinating their movements to avoid collisions, until the kit is completed (Figure 6.6d-6.6e). Finally, the operator changes the completed box with an empty one so that a new kit can start (Figure 6.6f). Figure 6.7 shows an example of how the schedule evolves during the assembly of a kit using the online algorithm. The initial plan in Figure 6.7a is recomputed multiple times during the kit preparation. Figures 6.7b to 6.7d show the updated plans when changes with respect to the previous one occur. At each rescheduling, only the operations not yet started can be re-planned.

The results obtained during the experimental campaign have been analysed according to the performance indices described below. Let $P = \{K_i\}$ be the finite set of the kits assembled during the experiments, $\overline{T}_{K_i}$ the time required to complete the i-th kit and $\overline{S}_{K_i}$ the total strain upon the human accumulated during the preparation of such kit. Then we can define:

- The cost of the i-th kit K_i:

$$f_i = \alpha \cdot \frac{\overline{T}_{K_i}}{T^I_{max}} + (1 - \alpha) \cdot \frac{\overline{S}_{K_i}}{S^I_{max}}$$

  where T^I_{max} and S^I_{max} are the upper bound on time and physical strain computed with reference to the entire kit.

- The productivity during the preparation of the i-th kit K_i:

$$\lambda_i = \frac{|K_i|}{\overline{T}_{K_i}}$$

  with $|K_i|$ the number of the objects contained in K_i. That is, the average number of picked items per second.

- The average strain per item of the i-th kit K_i:

$$\sigma_i = \frac{\overline{S}_{K_i}}{\sum_{j \in K_i} H_j}$$

  where the denominator is equal to the number of objects collected by the human.

Figure 6.8 reports the results of the experiments in terms of average cost, productivity and strain for the set P of all completed kits. To normalise with

respect to the kit size, each kit contributes to the aggregated performance according to its duration $\overline{T}_{K_i}$. Data are reported as box plots showing the weighted median and quartiles. Values for the two scheduling approaches (offline and online) as well as for the manual kitting case are shown for comparison. The latter has been estimated through simulations. In particular, Figure 6.8a shows the box plot of the average kit cost. The proposed dynamic scheduler achieves significantly better performance over the offline one, with a decrease in both median and dispersion of about 17%. This result must reflect in an increase in productivity and/or in ergonomics. Specifically, as one can see from Figures 6.8b and 6.8c, the lower values of the cost is mainly due to an increase of the productivity, which grows of about 19% compared to the case without rescheduling. Instead, the strain does not improve significantly over the offline scheduler, although it decreases with respect to the baseline case. The fact that an improvement in productivity seems preferable over one in strain can be explained by analysing results in more detail. When rescheduling takes place, data show that the case in which a task originally assigned to the robot is reallocated to the human is more frequent than the inverse case. Thus, with the online scheduler, the human tends to perform more tasks, with a subsequent rise in the strain. This effect is also strengthened by the fact that the algorithm prefers to assign the objects with worse ergonomic score to the robot.

Overall, the proposed online scheduling algorithm enables effective collaboration between human and robot during kitting operations, with good performance if compared to both the traditional human kitting process and an offline scheduler.

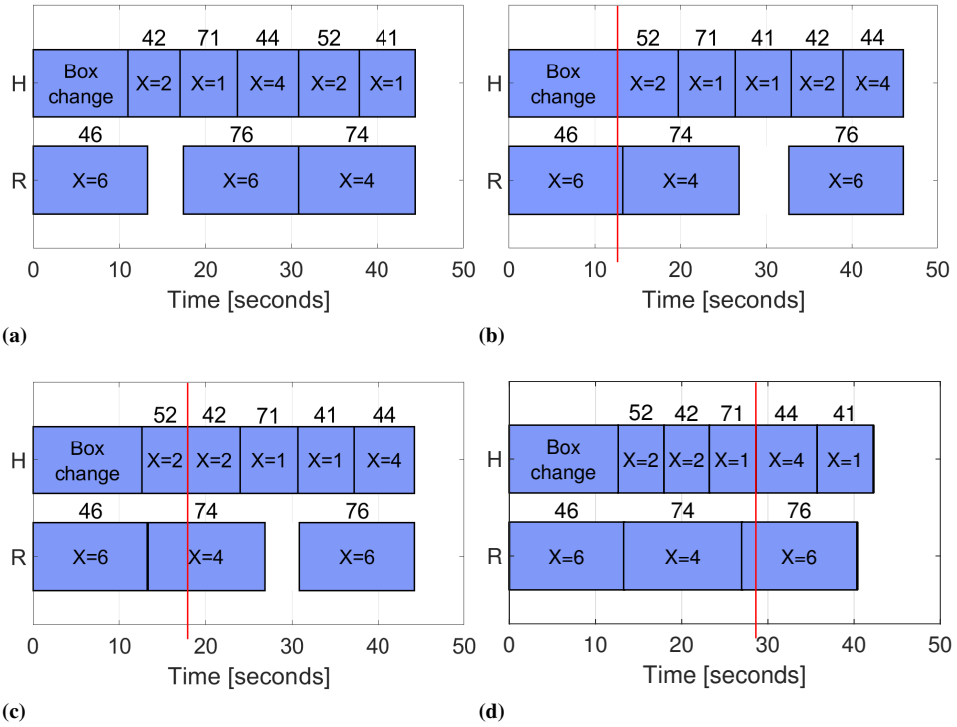


Figure 6.7: *Example of online rescheduling: the first schedule (a) is adapted multiple times to cope with human variability (b)-(d). Only plans that differ from the previous one are shown. Vertical red lines indicate the time at which rescheduling takes place. The object number is reported over each task, while its horizontal position along the rack is shown inside the bar.*

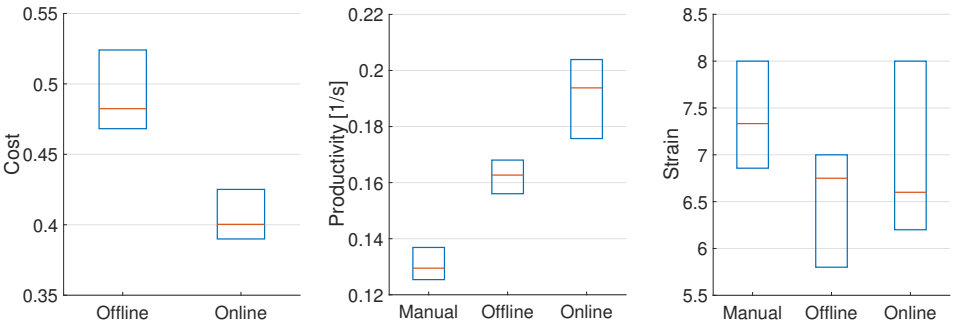


Figure 6.8: *Weighted boxplots of (a) average cost, (b) productivity, and (c) strain. Graphs show the weighted median and quartiles, and compare manual kitting, offline, and online scheduler.*

CHAPTER 7

Conclusions

The purpose of this thesis is to provide a contribution to the field of industrial human-robot collaboration, with a particular focus on controlling the production of multi-product assembly cells. Present-day manufacturing demands for flexible and efficient strategies in order to maintain good productivity regardless of the frequent changes in the production plans and the occurrence of unexpected events. Also, the presence of the human operators injects high variability in the system, which must be taken into account to achieve seamless coordination among workers and robots. This work addresses many aspects of the problem, providing contributions to the real-time monitoring of human activity, the dynamic scheduling of operations in the assembly cell, and the control of the collaborative kitting process.

In Part I the problem of modelling and monitoring the current human activity has been discussed considering that the operator can perform their operations in many ways. Chapter 2 presented a way to learn a model of the human task from demonstrations and train a classifier to parse the human activity in real-time. The early identification of low-level actions enhances the performance of collaboration in two ways. First, the human worker is notified in case of errors, so that he/she can immediately implement corrective actions. Second, early classification allows for a prompter prediction

of the future evolution of the task and, consequently, better planning of the robot actions. A different strategy has been presented in Chapter 3 to enhance flexibility and learning capabilities, which does not require any training phase. The algorithm, based on a modified version of the Dynamic Time Warping, learns online from previous repetitions of the same operation and automatically adds newly discovered variants to the activity model. The main aim of the proposed approach is to estimate the expected duration of the current human task, which is beneficial for agents' coordination. Experimental results showed that the algorithm provides accurate prediction also for long and complex operations and that the presence of many variants of the task does not degrade performance.

Part II dealt with the scheduling of the operations of humans and robots working in a multi-product assembly cell and a collaborative kitting process. First, Chapter 4 introduced the control system architecture composed of a Digital Twin of the assembly cell, a dynamic scheduler, and a human monitoring unit, which can exploit the algorithms presented in Part I. The proposed strategy endows the system with strong adaptation and learning capabilities, that are crucial to quickly react to changes in the workspace layout and the production. A Digital Twin based on Petri Nets describes both the physical structure of the workspace, the assembly tasks, the possible errors and faults, and the related recovery actions. The model tracks the status of the process in real-time based on information coming from the human monitoring unit and the robot controllers. The model is very general and able to describe a wide variety of cell configurations. Also, it has been shown that it can be generated automatically from a user-friendly definition of the assembly tasks and can be easily updated to keep up with changes in the production and the layout. Then, Chapter 5 presented the dynamic scheduling algorithm. A temporal description is added to the Petri Net model to simulate feasible system evolutions starting from the current state of the Digital Twin. Results showed that the scheduler is able to optimally plan the task of the agents also in case of a time-varying production mix and the occurrence of faults. The algorithm maximises productivity with respect to other scheduling algorithms that are available in the literature. Also, a novel visuo-haptic interface to give instructions to human operators has been introduced, which can be used in complex human-robot collaboration scenarios and allows expert users to improve performance. Finally, Chapter 6 applied the human-robot collaboration concept to the kitting process in order to enhance ergonomics and productivity. The proposed receding horizon dynamic scheduler, based on Mixed-Integer Linear Programming optimization, was able to obtain promising results that make

further research on the subject particularly interesting.

7.0.1 Future developments

In this work, we proposed strategies to address different aspects of the problem of controlling human-robot collaboration in a flexible manufacturing cell. Besides improving the proposed algorithms to increase performance and applicability, future works should concentrate on closer integration of all the components of the system architecture. The ultimate goal should be the synthesis of a unified framework capable of managing all aspects of collaborative flexible manufacturing: from production control to logistics, from robot control to human monitoring. To do so, the focus should also shift more and more from the single robotic cell to the whole plant. Indeed, in the factory of the future all functions will be tightly integrated to optimise the response time, and thus productivity and costs, in the face of changes.

For human-robot collaboration to be effective in this scenario, the employed framework must be able to deal with a continuously evolving and non-structured environment. Although the work presented in this thesis already goes in the direction of fast reconfiguration and online learning, further steps can be done to provide methods for interpreting and managing the evolution of the manufacturing workspace. For this purpose, recent advances in machine learning and artificial intelligence could be leveraged.

Finally, in this work, the human has been considered as a (weakly) controllable agent that receives instruction from the control system to optimise productivity. Although a certain freedom in the execution of the human tasks and the possibility of making errors has been considered, little consideration was given to the worker's acceptance of the scheduler commands and his/her comfort and stress during operation. As for the latter, both physiological and psychological measures could be included to guide optimal task allocation and planning, similar to what has been made for the collaborative kitting process in Chapter 6. To do so, additional sensors can help to quantify the strain and stress of the operators and allow the production to adapt not only to the current production demand, but also to each individual worker's preferences and capabilities. Regarding acceptance, human operators might be frustrated or stressed by being constantly monitored while working and having to always follow the received instructions with limited decision autonomy. Moreover, in the case of unmodelled situations or misinterpretations by the monitoring algorithms, expert users should be able to override the received instructions and perform the correct action. These

considerations call for strategies to manage the trade-off between productivity and the workers' initiative and well-being.

APPENDIX $\mathcal{A}$

Virtual simulation of a flexible assembly cell

In this Appendix, we discuss how to integrate the control strategy proposed in the present thesis, and in Chapter 4 in particular, into a commercial Product Lifecycle Management (PLM) software. PLM combines tools to manage the flow of information through the various stages of the product lifecycle in order to cope with the increasing complexity of manufacturing processes [103, 140]. The short product lifecycles and the large number of product variants that characterise present-day flexible manufacturing require frequent re-designs of workstations. In this view, PLM software is a powerful technology that supports the design and commissioning of new manufacturing processes with the aim of faster launch of new products and higher production quality. To this purpose, consolidated approaches for information management in job-shop or fully automated plants are already employed by enterprises, whereas suitable methods to address human-robot collaboration scenarios are still under study [60, 133].

The most advanced software, such as Siemens Process Simulate [151], provides 3D virtual simulation environments and automatic verification tools. The possibility to analyse new production concepts prior to imple-

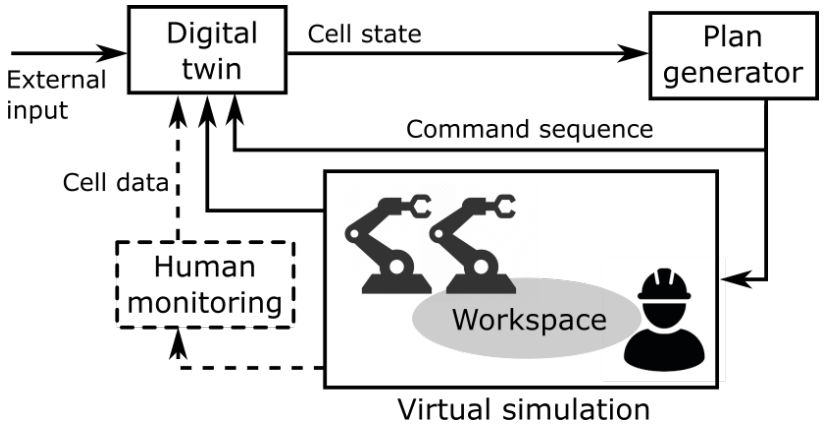


Figure A.1: *Control system architecture with virtual simulation of the assembly cell.*

ment them on the real plant brings various benefits, such as the reduction of costs by limiting the need of physical prototypes and tests, the mitigation of risk by performing what-if scenarios, the upfront optimization of cycle times, and the early validation of PLC logic and robotic operations. When the human is present, virtual simulation can be also used to evaluate ergonomics and safety of the workstation. Obviously, the results of the analysis performed on the virtual environment transfer to the real plant only to the extent that the former is an exact copy of the latter. To better mimic the reality, software like Process Simulate allows connecting PLCs and robot controllers to simulate the process on the actual hardware. Moreover, a lot of effort has been put recently on developing realistic digital models of the human. In this regard, one of the most advanced technologies is the Jack suite [131] that is integrated with Process Simulate.

One of the main limitations of virtual environments is the fact that simulation is always deterministic. Instead, real processes are characterized by high variability, especially when close human-robot collaboration is considered. Moreover, to govern such uncertainty, advanced control architectures like the one proposed in Section 4.1 are employed. For this reason, the objectives of this Section are:

- Building a 3D simulation of the assembly process that includes the uncertainty of human behaviour and the possibility of robot faults;
- Establishing a connection between the simulation and the proposed dynamic scheduler that keeps the same interface of the real system.

Commercial PLM applications are not specifically designed for these purposes, therefore they do not provide straightforward methods to fulfil the

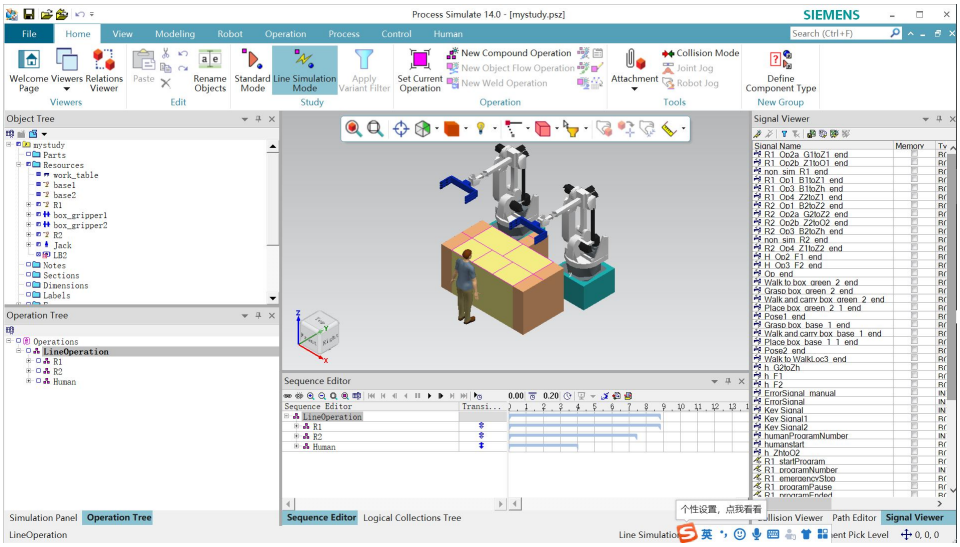


Figure A.2: *Graphical user interface of Siemens Process Simulate.*

aforementioned goals. However, if both objectives are achieved, it becomes possible to integrate a realistic 3D simulation of the manufacturing process with complex dynamic scheduling algorithms. The benefit is twofold. On the one hand, the behaviour of the system can be simulated in response to the dynamic scheduler that is implemented on the real cell and considering not only standard operating conditions, but also exploring fault situations. On the other hand, the simulation environment provides a realistic test-bed to analyse the performance of different scheduling algorithms. In the following, we outline the procedure to obtain the desired system architecture depicted in Figure A.1 using Siemens Process Simulate (to be compared to the one shown in Figure 4.1). The choice of the software is motivated by its widespread use in companies and the fact that it provides a state-ot-the-art human model.

A.1 Simulation of the assembly process

The first step is to build the simulation of the complete manufacturing cell. Figure A.2 shows the graphical user interface of Process Simulate that displays the main functionalities available on the software. Process Simulate provides an event-based simulation mode that allows simulating continuous production according to user-defined logic. Transition conditions are encoded based on the value of virtual signals that are associated with the occurrence of specific events. For example, signals can represent the end of

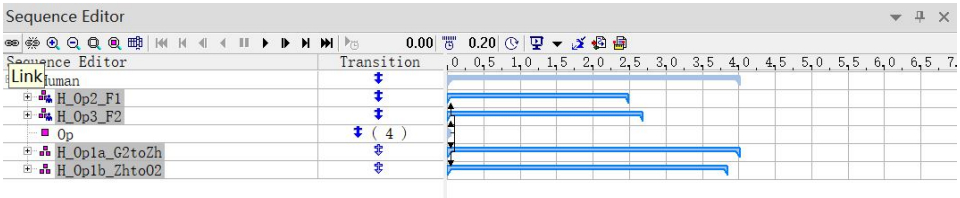


Figure A.3: *Example of human operations, with the non-sim operation (named Op) that allows for dynamic choice of the next task to perform.*

operations, commands sent to the agents, data collected by sensors, or the status of resources and devices (e.g. whether an automatic fixture is open or closed). By defining the required signals and the appropriate logic conditions, it is possible to simulate complex behaviours of the system. If the functions available on Process Simulate are insufficient, one can implement external user-defined functions exploiting the C# Tecnomatix API.

First of all, the 3D model of the environment is prepared by importing CAD models of all the elements that compose the manufacturing cell, e.g. robots, devices, worktables, and fences. Each object is defined with the appropriate type in Process Simulate. Human workers are included through the Jack model and can be personalised specifying gender and anthropometric data. The more complete the virtual model, the more relevant the results obtained in the simulation can be for the real system.

Second, each operation available to robots and humans must be implemented. Process Simulate gives the possibility to fine-tune task execution to the tiniest detail for both robots and humans. In our application agents should execute their tasks based on of the commands received by the external scheduler. Therefore, we must define operations in a way that allows the desired level of flexibility. A *robot program* can be used to gather all the tasks performed by a specific robot. Then, the next operation to start is selected by means of the *programNumber* signal, which is automatically created by Process Simulate. Robot programs cannot be used with the Jack model. Still, we need a similar mechanism to govern human operations, which is obtained by linking them to a root *non-sim operation* (see Figure A.3). The transition from the non-sim operation, which means that the human is idle, to the actual operations is controlled by user-defined signals, named *key signals*, associated to each task. Instead, signals that notify the end of the operations are generated by default for both humans and robots.

Finally, to complete the definition of the assembly process, operations must be associated with parts and products to model the material flow inside the robotic cell, as depicted in Figure A.4.

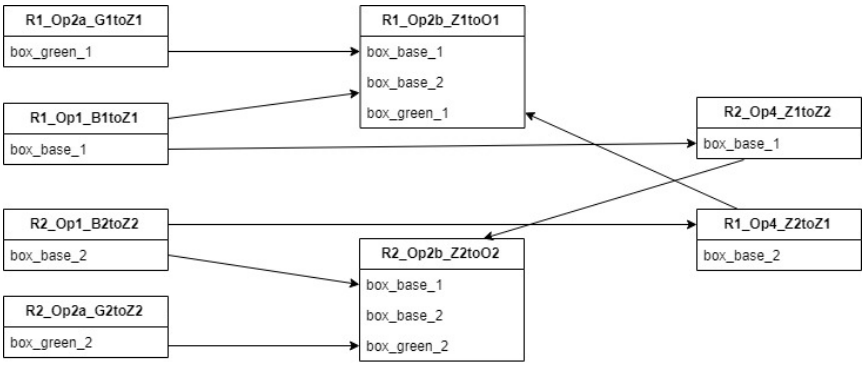


Figure A.4: *Example of material flow definition. Each box refers to an operation: the name is specified in the top part, the bottom one lists the parts involved in the operation. Arrows define the material flow.*

A.2 Simulation of the system variability

We are interested in endowing the simulation with two sources of uncertainty, namely the variability in the operation duration and the occurrence of errors and faults. Usually, variability in the duration of robot tasks arises from safety measures in the presence of the human. For instance, when an operator gets close to the robot, the robot might slow down and, if needed, stop completely. Process Simulate directly provides functionalities to model such cases. Instead, expedients must be conceived to encode human variability and faults. As for the former, the only solution we were able to implement inside the constraints of Process Simulate was to insert a random wait at the beginning of every new human operation. This model the fact that the worker can execute a task at different speeds.

To simulate faults and errors one can define additional operations that model interesting failure cases. Multiple failure conditions can be designed for each operation. Then, a logic block inside Process Simulate is used to manage the random occurrence of faults with a probability that can be changed by the user. For instance, when the command to start a task with one failure case is received, the standard operation is executed with probability p and the fault execution is executed with probability $1 - p$. Also, the possibility to directly command the faulty operation is included to allow the user to test specific what-if scenarios. To model a subset of robot faults, another possibility is given by manipulating the *programPause* signal during the execution of the robot program: when a fault occurs it is set to *true* and the operation stops, when the error is recovered the value is set back to *false* and the operation resumes.

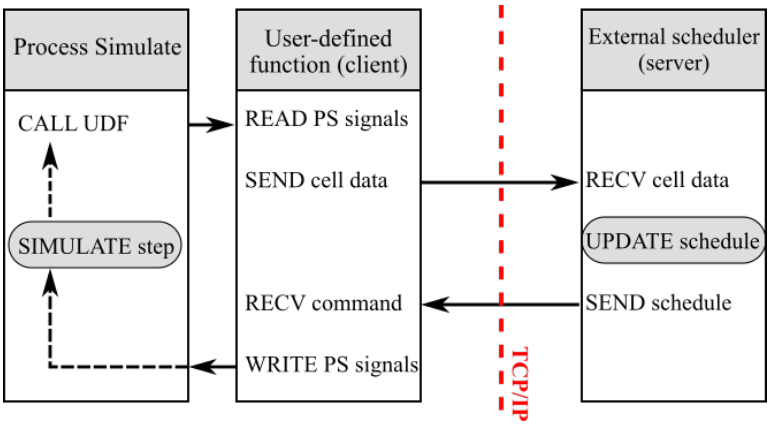


Figure A.5: *Client-server communication between Process Simulate and the external scheduler, mediated by the user-defined function.*

A.3 Communication with the external scheduler

Communication between Process Simulate and the external control architecture takes place with a client-server architecture. The virtual simulation acts as the client, the external scheduler as the server. The scheme of the communication routine is sketched in Figure A.5. Specifically, communication is mediated by a user-defined function, which is implemented in a logic block inside Process Simulate. At each simulation step, Process Simulate calls the user-defined function, which reads the current value of notable virtual signals and elaborates them to obtain data that are compliant with the interface of the external scheduler. To track the status of the agents in the cell, the user-defined function implements finite state machines whose transitions are triggered by signals related to the start and completion of operations. Then, the response coming from the scheduler is decoded into new values for the Process Simulate signals that update the status of the simulation before the next step is elaborated. In this way, the commands on the next operations to perform are dispatched to the virtual agents. The bridging role of the user-defined function allows a direct connection with the simulation environment without requiring any modification to the control architecture, which is ready to be applied to the real system.

Particular attention should be put on the human monitoring unit. In principle, it is possible to add virtual vision sensors in Process Simulate and provide the monitoring algorithm with the images required to analyse the behaviour of the simulated human model. However, images taken from the simulation cannot be realistic. Therefore, monitoring algorithms designed

for usage on a real system will probably have difficulties in analysing pictures coming from the digital simulation. For this reason, in most cases it is more meaningful to remove the monitoring unit from the control architecture, implement it directly in the Process Simulate environment, and provide the external scheduler with all the required data.

A.3.1 Validation tests

We tested the proper functioning of the overall architecture with a cell composed of two robots and a human operator. The assembly process comprises a single product type composed of two parts that can be completed following six different sequences of operations. Namely, each robot can perform the assembly on its own, with the help of the other robot, or the human. Each robot can perform four operations, plus their faulty alternatives (one for each task). The human can perform one operation, plus the recovery actions to reactivate the failed robots. The setup has been used to test the communication with the external scheduler and simulate different scenarios with various levels of complexity and variability. The system always worked as intended and proved effective to be used as test-bed for the scheduling algorithm.

List of Figures

List of Tables

Bibliography

[1] CoLLaboratE: Co-production CeLL performing Human-Robot Collaborative AssEmbly. https://collaborate-project.eu/.

[2] ColRobot: Collaborative Robotics for Assembly and Kitting in Smart Manufacturing. https://colrobot.eu/.

[3] SHAREWORK: Safe and effective HumAn-Robot coopEration toWards a better cOmpetiveness on cuRrent automation lacK manufacturing processes. https://sharework-project.eu/.

[4] THOMAS: Mobile dual arm robotic workers with embedded cognition for hybrid and dynamically reconfigurable manufacturing systems. http://www.thomas-project.eu/.

[5] A. Ajoudani, A. M. Zanchettin, S. Ivaldi, A. Albu-Schäffer, K. Kosuge, and O. Khatib. Progress and prospects of the human–robot collaboration. *Autonomous Robots*, 42(5):957–975, 2018.

[6] M. Awais and D. Henrich. Human-robot collaboration by intention recognition using probabilistic state machines. In *19th International Workshop on Robotics in Alpe-Adria-Danube Region (RAAD 2010)*, pages 75–80, 2010.

[7] J. Baraglia, M. Cakmak, Y. Nagai, R. P. N. Rao, and M. Asada. Efficient human-robot collaboration: When should a robot take initiative? *The International Journal of Robotics Research*, 36(5-7):563–579, 2017.

[8] C. Berlin and C. Adams. *Production Ergonomics*. Ubiquity Press, London, Jun 2017.

[9] A. Bilberg and A. A. Malik. Digital twin driven human–robot collaborative assembly. *CIRP Annals*, 68(1):499 – 502, 2019.

[10] M. E. A. Boudella, E. Sahin, and Y. Dallery. Kitting optimisation in just-in-time mixed-model assembly lines: assigning parts to pickers in a hybrid robot–operator kitting system. *International Journal of Production Research*, 56(16):5475–5494, 2018.

[11] G. Bruno and D. Antonelli. Dynamic task classification and assignment for the management of human-robot collaborative teams in workcells. *The International Journal of Advanced Manufacturing Technology*, 98(9):2415–2427, October 2018.

[12] G. Buizza Avanzini, N. M. Ceriani, A. M. Zanchettin, P. Rocco, and L. Bascetta. Safety control of industrial robots based on a distributed distance sensor. *IEEE Transactions on Control Systems Technology*, 22(6):2127–2140, 2014.

[13] J. L. Burke, M. S. Prewett, A. A. Gray, L. Yang, F. R. B. Stilson, M. D. Coovert, L. R. Elliot, and E. Redden. Comparing the effects of visual-auditory and visual-tactile feedback on user performance: A meta-analysis. In *Proceedings of the 8th International Conference on Multimodal Interfaces*, page 108–117, 2006.

[14] B. Busch, M. Toussaint, and M. Lopes. Planning ergonomic sequences of actions in human-robot interaction. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1916–1923, May 2018.

[15] Z. Cao, T. Simon, S. Wei, and Y. Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1302–1310, 2017.

[16] A. C. Caputo, P. M. Pelagagge, and P. Salini. A model for planning and economic comparison of manual and automated kitting systems. *International Journal of Production Research*, 0(0):1–24, 2020.

[17] A. Casalino, D. Bazzi, A. M. Zanchettin, and P. Rocco. Optimal proactive path planning for collaborative robots in industrial contexts. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6540–6546, 2019.

[18] A. Casalino, C. Messeri, M. Pozzi, A. M. Zanchettin, P. Rocco, and D. Prattichizzo. Operator awareness in human–robot collaboration through wearable vibrotactile feedback. *IEEE Robotics and Automation Letters*, 3(4):4289–4296, 2018.

[19] A. Casalino, A. M. Zanchettin, L. Piroddi, and P. Rocco. Optimal scheduling of human-robot collaborative assembly operations with time petri nets. *IEEE Transaction on Automation Science and Engineering*, pages 1–15, 2019.

[20] C. Chao and A. Thomaz. Timed petri nets for fluent turn-taking over multimodal interaction resources in human-robot collaboration. *The International Journal of Robotics Research*, 35(11):1330–1353, 2016.

[21] F. Chen, K. Sekiyama, F. Cannella, and T. Fukuda. Optimal subtask allocation for human and robot collaboration within hybrid assembly system. *IEEE Transactions on Automation Science and Engineering*, 11(4):1065–1075, 2014.

[22] S. Chen, N. B. Shroff, and P. Sinha. Heterogeneous delay tolerant task scheduling and energy management in the smart grid with renewable energy. *IEEE Journal on Selected Areas in Communications*, 31(7):1258–1267, 2013.

[23] Y. Chen, X. Mao, F. Hou, Q. Wang, and S. Yang. Combining re-allocating and re-scheduling for dynamic multi-robot task allocation. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 395–400, 2016.

[24] A. Cherubini, R. Passama, A. Crosnier, A. Lasnier, and P. Fraisse. Collaborative manufacturing with physical human–robot interaction. *Robotics and Computer-Integrated Manufacturing*, 40:1 – 13, 2016.

[25] C. D. Chitraranjan, A. S. Perera, and A. M. Denton. Tracking vehicle trajectories by local dynamic time warping of mobile phone signal strengths and its potential in travel-time estimation. *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pages 445–450, March 2015.

[26] M. Christmansson, L. Medbo, G.-Å. Hansson, K. Ohlsson, J. Unge Byström, T. Möller, and M. Forsman. A case study of a principally new way of materials kitting—an evaluation of time consumption and physical workload. *International Journal of Industrial Ergonomics*, 30(1):49 – 65, 2002.

[27] G. Civitarese and C. Bettini. Monitoring objects manipulations to detect abnormal behaviors. In *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 388–393, March 2017.

[28] E. Coupeté, F. Moutarde, S. Manitsaris, and O. Hugues. Recognition of Technical Gestures for Human-Robot Collaboration in Factories. In *The Ninth International Conference on Advances in Computer-Human Interactions*, Venise, Italy, April 2016.

[29] J. Dai, A. Benini, H. Lin, P. J. Antsaklis, M. J. Rutherford, and K. P. Valavanis. Learning-based formal synthesis of cooperative multi-agent systems with an application to robotic coordination. In *2016 24th Mediterranean Conference on Control and Automation (MED)*, pages 1008–1013, 2016.

[30] L. S. H. de Mello and A. C. Sanderson. And/or graph representation of assembly plans. *IEEE Transactions on Robotics and Automation*, 6(2):188–199, 1990.

[31] J. A. Diego-Mas and J. Alcaide-Marzal. Using kinect™ sensor in observational methods for assessing postures at work. *Applied Ergonomics*, 45(4):976 – 985, 2014.

[32] J. L. Drury, J. Scholtz, and H. A. Yanco. Awareness in human-robot interactions. In *2003. IEEE International Conference on Systems, Man and Cybernetics*, volume 1, pages 912–918. IEEE, 2003.

[33] M. Elshafie and G. M. Bone. Markerless human tracking for industrial environments. In *2008 Canadian Conference on Electrical and Computer Engineering*, pages 001139–001144, 2008.

[34] European Foundation for the Improvement of Living and Working Conditions. Sixth european working conditions survey: 2015. Available online at: `https://www.eurofound.europa.eu/surveys/european-working-conditions-surveys/sixth-european-working-conditions-survey-2015`.

[35] M. Faber, S. Kuz, A. Mertens, and C. M. Schlick. Model-based evaluation of cooperative assembly processes in human-robot collaboration. In Christopher Schlick and Stefan Trzcieliński, editors, *Advances in Ergonomics of Manufacturing: Managing the Enterprise of the Future*, pages 101–112, Cham, 2016. Springer International Publishing.

[36] P. Fager, M. Calzavara, and F. Sgarbossa. Kit preparation with cobot-supported sorting in mixed model assembly. *IFAC-PapersOnLine*, 52(13):1878 – 1883, 2019. 9th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2019.

[37] P. Fager, R. Hanson, L. Medbo, and M. I. Johansson. Links between kit quality and kit preparation design. *International Journal of Production Research*, 0(0):1–15, 2020.

[38] F. Ferraguti, R. Villa, C. Talignani Landi, A. M. Zanchettin, P. Rocco, and C. Secchi. A unified architecture for physical and ergonomic human–robot collaboration. *Robotica*, 38(4):669–683, 2020.

[39] C. Ferreira, G. Figueira, and P. Amorim. Optimizing dispatching rules for stochastic job shop scheduling. In *International Conference on Hybrid Intelligent Systems*, pages 321–330, 2018.

[40] C. Finnsgård and C. Wänström. Factors impacting manual picking on assembly lines: an experiment in the automotive industry. *International Journal of Production Research*, 51(6):1789–1798, 2013.

[41] F. Flacco, T. Kröger, A. De Luca, and O. Khatib. A depth space approach to human-robot collision avoidance. In *2012 IEEE International Conference on Robotics and Automation*, pages 338–345, 2012.

[42] International Organization for Standardization. *ISO 10218:2011 Robots and robotic devices — Safety requirements for industrial robots*. July 2011.

[43] International Organization for Standardization. *ISO/TS 15066:2016 Robots and robotic devices — Collaborative robots*. February 2016.

[44] L. Franco, G. Salvietti, and D. Prattichizzo. Command acknowledge through tactile feedback improves the usability of an emg-based interface for the frontalis muscle. In *IEEE 2019 World Haptics Conference*, 2019.

[45] M. Geravand, F. Flacco, and A. De Luca. Human-robot physical interaction and collaboration using an industrial robot with a closed control architecture. In *2013 IEEE International Conference on Robotics and Automation*, pages 4000–4007, 2013.

[46] A. Gilchrist. Introducing industry 4.0. In *Industry 4.0*, pages 195–215. Springer, 2016.

[47] A. Gilchrist. Smart factories. In *Industry 4.0*, pages 217–230. Springer, 2016.

[48] A. Giridhar and P. R. Kumar. Scheduling automated traffic on a network of roads. *IEEE Transactions on Vehicular Technology*, 55(5):1467–1474, 2006.

[49] M. C. Gombolay, R. J. Wilcox, and J. A. Shah. Fast scheduling of robot teams performing tasks with temporospatial constraints. *IEEE Transactions on Robotics*, 34(1):220–239, 2018.

[50] L. Gualtieri, E. Rauch, R. Vidoni, and D. T. Matt. An evaluation methodology for the conversion of manual assembly systems into human-robot collaborative workcells. *Procedia Manufacturing*, 38:358 – 366, 2019. 29th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM 2019), June 24-28, 2019, Limerick, Ireland, Beyond Industry 4.0: Industrial Advances, Engineering Education and Intelligent Manufacturing.

[51] S. Haddadin, A. Albu-Schaffer, A. De Luca, and G. Hirzinger. Collision detection and reaction: A contribution to safe physical human-robot interaction. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3356–3363, 2008.

[52] S. Haddadin, S. Haddadin, A. Khoury, T. Rokahr, S. Parusel, R. Burgkart, A. Bicchi, and A. Albu-Schäffer. On making robots understand safety: Embedding injury knowledge into control. *The International Journal of Robotics Research*, 31(13):1578–1602, 2012.

[53] S. Haddadin, M. Suppa, S. Fuchs, T. Bodenmüller, A. Albu-Schäffer, and G. Hirzinger. Towards the robotic co-worker. *Robotics Research*, 70:261–282, 2011.

[54] H. Haggag, M. Hossny, S. Nahavandi, and D. Creighton. Real time ergonomic assessment for assembly operations using kinect. In *2013 UKSim 15th International Conference on Computer Modelling and Simulation*, pages 495–500, April 2013.

[55] A. Hamacher, N. Bianchi-Berthouze, A. G. Pipe, and K. Eder. Believing in bert: Using expressive communication to enhance trust and counteract operational error in physical human-robot interaction. In *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 493–500, 2016.

[56] S. Hameed, T. Ferris, S. Jayaraman, and N. Sarter. Supporting interruption management through informative tactile and peripheral visual cues. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 50(3):376–380, 2006.

[57] B. Hartmann. *Human worker activity recognition in industrial environments*. KIT Scientific Publishing, 2011.

[58] K. P. Hawkins, Nam Vo, S. Bansal, and A. F. Bobick. Probabilistic human action prediction and wait-sensitive planning for responsive human-robot collaboration. In *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 499–506, 2013.

[59] Z. He, Z. Li, and A. Giua. Cycle time optimization of deterministic timed weighted marked graphs by transformation. *IEEE Transactions on Control Systems Technology*, 25(4):1318–1330, 2017.

[60] W. Herfs, S. Storms, and O. Petrovic. An approach on simplifying the commissioning of collaborative assembly workstations based on product-lifecycle-management and intuitive robot programming. In *International Conference on Intelligent Human Systems Integration (IHSI 2019)*, pages 43 – 49, 2019.

[61] S. Hignett and L. McAtamney. Rapid entire body assessment (reba). *Applied Ergonomics*, 31(2):201 – 205, 2000.

[62] E. Hourdakis and P. Trahanias. A robust method to predict temporal aspects of actions by observation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1931–1938, May 2018.

[63] S. Y. Hua and D. J. Johnson. Research issues on factors influencing the choice of kitting versus line stocking. *International Journal of Production Research*, 48(3):779–800, 2010.

[64] C. Huang and B. Mutlu. Anticipatory robot control for efficient human-robot collaboration. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 83–90, 2016.

[65] American National Standards Institute. *ANSI/RIA R15.06-2012 - Industrial Robots and Robot Systems - Safety Requirements*. 2012.

[66] International Federation of Robotics. *World Robotics Report 2019 - Industrial Robots*. 2019. `https://ifr.org/worldrobotics/`.

[67] S. Jlassi, S. Tliba, and Y. Chitour. An online trajectory generator-based impedance control for co-manipulation tasks. In *2014 IEEE Haptics Symposium (HAPTICS)*, pages 391–396, 2014.

[68] L. Johannsmeier and S. Haddadin. A hierarchical human-robot inter-action-planning framework for task allocation in collaborative industrial assembly processes. *IEEE Robtics and Automation Letters*, 2(1):41–48, 2017.

[69] L. A. Jones and N. Sarter. Tactile displays: Guidance for their design and application. *Human Factors*, 50(1):90–111, 2008.

[70] S. Kaczmarek, S. Hogreve, and K. Tracht. Progress monitoring and gesture control in manual assembly systems using 3d-image sensors. *Procedia CIRP*, 37:1 – 6, 2015. CIRPe 2015 - Understanding the life cycle implications of manufacturing.

[71] R. Kelley, A. Tavakkoli, C. King, M. Nicolescu, M. Nicolescu, and G. Bebis. Understanding human intentions via hidden markov models in autonomous mobile robots. In *Proceedings of the 3rd ACM/IEEE Int. Conference on Human Robot Interaction*, pages 367–374, 2008.

[72] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 500–505, 1985.

[73] H. Kim, J. Lee, and T. Lee. A petri net-based modeling and scheduling with a branch and bound algorithm. In *2012 IEEE International Conference on Systems, Man, and Cybernetics*, pages 1779–1784, 2012.

[74] H. S. Koppula and A. Saxena. Anticipating human activities using object affordances for reactive robotic response. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(1):14–29, 2016.

[75] Y. A. Korablev and M. Y. Shestopalov. Faults diagnostics on the basis of dtw-classification. In *2016 XIX IEEE International Conference on Soft Computing and Measurements (SCM)*, pages 94–97, May 2016.

[76] J. Krüger, T.K. Lien, and A. Verl. Cooperation of human and machines in assembly lines. *CIRP Annals*, 58(2):628 – 646, 2009.

[77] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn. Digital twin in manufacturing: A categorical literature review and classification. *IFAC-PapersOnLine*, 51(11):1016 – 1022, 2018. 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018.

[78] V. Krueger, F. Rovida, B. Grossmann, R. Petrick, M. Crosby, A. Charzoule, G. M. Garcia, S. Behnke, C. Toscano, and G. Veiga. Testing the vertical and cyber-physical integration of cognitive robots in manufacturing. *Robotics and Computer-Integrated Manufacturing*, 57:213 – 229, 2019.

[79] A. Kshirsagar, H. Kress-Gazit, and G. Hoffman. Specifying and synthesizing human-robot handovers. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5930–5936, 2019.

[80] D. Kulić and E. Croft. Pre-collision safety strategies for human-robot interaction. *Autonomous Robots*, 22:149–164, October 2007.

[81] W. Y. Kwon and I. H. Suh. A temporal bayesian network with application to design of a proactive robotic assistant. In *2012 IEEE International Conference on Robotics and Automation*, pages 3685–3690, 2012.

[82] B. Lacevic, P. Rocco, and A. M. Zanchettin. Safety assessment and control of robotic manipulators using danger field. *IEEE Transactions on Robotics*, 29(5):1257–1270, 2013.

[83] P. A. Lasota and J. A. Shah. Analyzing the effects of human-aware motion planning on close-proximity human–robot collaboration. *Human Factors*, 57(1):21–33, 2015. PMID: 25790568.

[84] P. A. Lasota and J. A. Shah. Bayesian estimator for partial trajectory alignment. In *2019, Robotics: Science and System (RSS)*, June 2019.

[85] J. Lee, B. Bagheri, and H. Kao. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing Letters*, 3:18 – 23, 2015.

[86] D. Lefebvre and F. Basile. Design of control sequences for timed petri nets based on tree encoding. *IFAC-PapersOnLine*, 51(7):218 – 223, 2018. 14th IFAC Workshop on Discrete Event Systems.

[87] C. Lenz, A. Sotzek, T. Röder, H. Radrich, A. Knoll, M. Huber, and S. Glasauer. Human workflow analysis using 3d occupancy grid hand tracking in a human-robot collaboration scenario. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3375–3380, 2011.

[88] H. Li, L. Kulik, and K. Ramamohanarao. Spatio-temporal trajectory simplification for inferring travel paths. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 63–72, 2014.

[89] K. Li and Y. Fu. Prediction of human activity by discovering temporal sequence patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1644–1657, Aug 2014.

[90] T. Lisini Baldi, G. Paolocci, and D. Prattichizzo. Human guidance: Suggesting walking pace under manual and cognitive load. In *Eurohaptics*, Pisa, Italy, June 2018.

[91] X. Liu and J. Zhang. Active learning for human action recognition with gaussian processes. In *2011 18th IEEE International Conference on Image Processing*, pages 3253–3256, 2011.

[92] F. Lolli, R. Gamberini, C. Giberti, B. Rimini, and F. Bondi. A simulative approach for evaluating alternative feeding scenarios in a kanban system. *International Journal of Production Research*, 54(14):4228–4239, 2016.

[93] P. Lou, Q. Liu, Z. Zhou, H. Wang, and S. X. Sun. Multi-agent-based proactive–reactive scheduling for a job shop. *Int. Journal of Advanced Manufacturing Technology*, 59(1):311–324, 2012.

[94] A. Lund. Measuring usability with the use questionnaire. *Usability interface*, 8, 2001.

[95] R. Luo, R. Hayne, and D. Berenson. Unsupervised early prediction of human reaching for human–robot collaboration in shared workspaces. *Autonomous Robots*, 42:631–648, 2018.

[96] G. Maeda, M. Ewerton, G. Neumann, R. Lioutikov, and J. Peters. Phase estimation for fast action recognition and trajectory generation in human–robot collaboration. *The International Journal of Robotics Research*, 36(13-14):1579–1594, 2017.

[97] S. Makris, P. Karagiannis, S. Koukas, and A. S. Matthaiakis. Augmented reality system for operator support in human–robot collaborative assembly. *CIRP Annals-Manufacturing Technology*, 65(1):61–64, 2016.

[98] A. Malaisé, P. Maurice, F. Colas, F. Charpillet, and S. Ivaldi. Activity Recognition With Multiple Wearable Sensors for Industrial Applications. In *ACHI 2018 - Eleventh International Conference on Advances in Computer-Human Interactions*, Rome, Italy, March 2018.

[99] A. A. Malik and A. Bilberg. Digital twins of human robot collaboration in a production setting. *Procedia Manufacturing*, 17:278 – 285, 2018. 28th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM2018), June 11-14, 2018, Columbus, OH, USAGlobal Integration of Intelligent Manufacturing and Smart Industry for Good of Humanity.

[100] V. M. Manghisi, A. E. Uva, M. Fiorentino, V. Bevilacqua, G. F. Trotta, and G. Monno. Real time rula assessment using kinect v2 sensor. *Applied Ergonomics*, 65:481 – 491, 2017.

[101] N. Mansfeld, M. Hamad, M. Becker, A. G. Marin, and S. Haddadin. Safety map: A unified representation for biomechanics impact data and robot instantaneous dynamic properties. *IEEE Robotics and Automation Letters*, 3(3):1880–1887, July 2018.

[102] G. Marin, F. Dominio, and P. Zanuttigh. Hand gesture recognition with jointly calibrated leap motion and depth sensor. *Multimedia Tools and Applications*, 75(22):14991–15015, Nov 2016.

[103] F. Mas, R. Arista, M. Oliva, B. Hiebert, I. Gilkerson, and J. Rios. A review of plm impact on us and eu aerospace industry. *Procedia Engineering*, 132:1053 – 1060, 2015. MESIC Manufacturing Engineering Society International Conference 2015.

[104] L. McAtamney and E. N. Corlett. Rula: a survey method for the investigation of work-related upper limb disorders. *Applied Ergonomics*, 24(2):91 – 99, 1993.

[105] L. Meli, C. Pacchierotti, and D. Prattichizzo. Sensory subtraction in robot-assisted surgery: fingertip skin deformation feedback to ensure safety and improve transparency in bimanual haptic interaction. *IEEE Transactions on Biomedical Engineering*, 61(4):1318–1327, 2014.

[106] C. Messeri, A. M. Zanchettin, P. Rocco, E. Gianotti, A. Chirico, S. Magoni, and A. Gaggioli. On the effects of leader-follower roles in dyadic human-robot synchronisation. *IEEE Transactions on Cognitive and Developmental Systems*, pages 1–1, 2020.

[107] A. Mohammed, B. Schmidt, and L. Wang. Active collision avoidance for human–robot collaboration driven by vision sensors. *International Journal of Computer Integrated Manufacturing*, 30(9):970–980, 2017.

[108] R. Mosberger, H. Andreasson, and A. J. Lilienthal. Multi-human tracking using high-visibility clothing for industrial safety. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 638–644, 2013.

[109] M. Munaro, C. Lewis, D. Chambers, P. Hvass, and E. Menegatti. RGB-D human detection and tracking for industrial environments. *Intelligent Autonomous Systems*, 302(13):1655–1668, 2016.

[110] N. Najmaei, M. R. Kermani, and M. A. Al-Lawati. A new sensory system for modeling and tracking humans within industrial work cells. *IEEE Transactions on Instrumentation and Measurement*, 60(4):1227–1236, 2011.

[111] E. Negri, L. Fumagalli, and M. Macchi. A review of the roles of digital twin in cps-based production systems. *Procedia Manufacturing*, 11:939 – 948, 2017. 27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy.

[112] N. Nikolakis, N. Kousi, G. Michalos, and S. Makris. Dynamic scheduling of shared human-robot manufacturing operations. *Procedia CIRP*, 72:9 – 14, 2018. 51st CIRP Conference on Manufacturing Systems.

[113] E. Nunes, M. McIntire, and M. Gini. Decentralized allocation of tasks with temporal and precedence constraints to a team of robots. In *2016 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, pages 197–202, 2016.

[114] I. Oikonomidis, N. Kyriazis, and A. A. Argyros. Tracking the articulated motion of two strongly interacting hands. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1862–1869, June 2012.

[115] B. Parsa, A. Narayanan, and B. Dariush. Spatio-temporal pyramid graph convolutions for human action recognition and postural assessment. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1069–1079, March 2020.

[116] D. Pavlichenko, G. García Martín, S. Koo, and S. Behnke. Kittingbot: A mobile manipulation robot for collaborative kitting in automotive logistics. In *Intelligent Autonomous Systems 15*, pages 849–864. Springer International Publishing, 2019.

[117] M. Pearce, B. Mutlu, J. Shah, and R. Radwin. Optimizing makespan and ergonomics in integrating collaborative robots into manufacturing processes. *IEEE Transactions on Automation Science and Engineering*, 15(4):1772–1784, Oct 2018.

[118] S. Pellegrinelli, F. L. Moro, N. Pedrocchi, L. Molinari Tosatti, and T. Tolio. A probabilistic approach to workspace sharing for human–robot cooperation in assembly tasks. *CIRP Annals*, 65(1):57 – 60, 2016.

[119] L. Peternel, W. Kim, J. Babič, and A. Ajoudani. Towards ergonomic control of human-robot co-manipulation and handover. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 55–60, 2017.

[120] E. K. Phillips and F. G. Jentsch. Supporting situation awareness through robot-to-human information exchanges under conditions of visuospatial perspective taking. *Journal of Human-Robot Interaction*, 6(3):92–117, 2017.

[121] Pick To Light Systems, S. L. Kit to light. Website: `https://www.picktolightsystems.com/en/picking-products/kitting`.

[122] P. Plantard, H. P. H. Shum, A. Le Pierres, and F. Multon. Validation of an ergonomic assessment method using kinect data in real workplace conditions. *Applied Ergonomics*, 65:562 – 569, 2017.

[123] C. Pérez-D'Arpino and J. A. Shah. Fast target prediction of human reaching motion for cooperative human-robot manipulation tasks using time series classification. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6175–6182, May 2015.

[124] C. Qian, X. Sun, Y. Wei, X. Tang, and J. Sun. Realtime and robust hand tracking from depth. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1106–1113, June 2014.

[125] M. Ragaglia, L. Bascetta, and P. Rocco. Multiple camera human detection and tracking inside a robotic cell - an approach based on image warping, computer vision, k-d trees and particle filtering. In *Proceedings of the 11th International Conference on Informatics in Control, Automation and Robotics - Volume 1: ICINCO,*, pages 374–381. INSTICC, SciTePress, 2014.

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

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

[128] S. M. M. Rahman, B. Sadrfaridpour, and Y. Wang. Trust-based optimal subtask allocation and model predictive control for human-robot collaborative assembly in manufacturing. In *Dynamic Systems and Control Conference*, volume 57250, page V002T32A004. American Society of Mechanical Engineers, 2015.

[129] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 262–270, New York, NY, USA, 2012. ACM.

[130] F. Ranz, V. Hummel, and W. Sihn. Capability-based task allocation in human-robot collaboration. *Procedia Manufacturing*, 9:182 – 189, 2017. 7th Conference on Learning Factories, CLF 2017.

[131] U. Raschke and C. Cort. Siemens jack. In Sofia Scataglini and Gunther Paul, editors, *DHM and Posturography*, pages 35 – 48. Academic Press, 2019.

[132] H. C. Ravichandar and A. P. Dani. Human intention inference using expectation-maximization algorithm with online model learning. *IEEE Transactions on Automation Science and Engineering*, 14(2):855–868, 2017.

[133] P. Rückert, K. Tracht, W. Herfs, S. Roggendorf, V. Schubert, and M. Schneider. Consolidation of product lifecycle information within human-robot collaboration for assembly of multi-variant products. *Procedia Manufacturing*, 49:217 – 221, 2020. Proceedings of the 8th International Conference on Through-Life Engineering Services – TESConf 2019.

[134] D. Riedelbauch and D. Henrich. Exploiting a human-aware world model for dynamic task allocation in flexible human-robot teams. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6511–6517, 2019.

[135] S. Robla-Gómez, V. M. Becerra, J. R. Llata, E. González-Sarabia, C. Torre-Ferrero, and J. Pérez-Oria. Working together: A review on safe human-robot collaboration in industrial environments. *IEEE Access*, 5:26754–26773, 2017.

[136] G. Rogez, C. Orrite, J.J. Guerrero, and P. H. S. Torr. Exploiting projective geometry for view-invariant monocular human motion analysis in man-made environments. *Computer Vision and Image Understanding*, 120:126 – 140, 2014.

[137] A. Roncone, O. Mangin, and B. Scassellati. Transparent role assignment and task allocation in human robot collaboration. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1014–1021, May 2017.

[138] R. Rosen, G. von Wichert, G. Lo, and K. D. Bettenhausen. About the importance of autonomy and digital twins for the future of manufacturing. *IFAC-PapersOnLine*, 48(3):567 – 572, 2015. 15th IFAC Symposium onInformation Control Problems inManufacturing.

[139] L. Rozo, S. Calinon, D. G. Caldwell, P. Jiménez, and C. Torras. Learning physical collaborative robot behaviors from human demonstrations. *IEEE Transactions on Robotics*, 32(3):513–527, 2016.

[140] D. Dutta M. Garetti S. Terzi, A. Bouras and D. Kiritsis. Product lifecycle management – from its history to its new role. *International Journal Product Lifecycle Management*, 4(4):360–389, 2010.

[141] H. Sadeghian, L. Villani, M. Keshmiri, and B. Siciliano. Task-space control of robot manipulators with null-space compliance. *IEEE Transactions on Robotics*, 30(2):493–506, 2014.

[142] B. Sadrfaridpour and Y. Wang. Collaborative assembly in hybrid manufacturing cells: An integrated framework for human-robot interaction. *IEEE Transactions on Automation Science and Engineering*, 2017.

[143] W. Sakata, F. Kobayashi, and H. Nakamoto. Robot-human handover based on motion prediction of human. In *2017 6th International Conference on Informatics, Electronics and Vision 2017 7th International Symposium in Computational Medical and Health Technology (ICIEV-ISCMHT)*, pages 1–4, 2017.

[144] K. Sakita, K. Ogawara, S. Murakami, K. Kawamura, and K. Ikeuchi. Flexible cooperation between human and robot by interpreting human intention from gaze information. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 846–851, 2004.

[145] A. Salmi, P. David, J.D. Summers, and E. Blanco. A modelling language for assembly sequences representation, scheduling and analyses. *International Journal of Production Research*, 52(13):3986–4006, 2014.

[146] S. Scheggi, M. Aggravi, and D. Prattichizzo. Cooperative navigation for mixed human-robot teams using haptic feedback. *IEEE Transactions on Human-Machine Systems*, 47(4):462–473, 2017.

[147] C.J. Sellers and S.Y. Nof. Performance analysis of robotic kitting systems. *Robotics and Computer-Integrated Manufacturing*, 6(1):15 – 24, 1989.

[148] A. Shafti, A. Ataka, B. U. Lazpita, A. Shiva, H. A. Wurdemann, and K. Althoefer. Real-time robot-assisted ergonomics*. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1975–1981, 2019.

[149] J. Shi and G. S. Koonjul. Real-time grasping planning for robotic bin-picking and kitting applications. *IEEE Transactions on Automation Science and Engineering*, 14(2):809–819, April 2017.

[150] Z. Shi, L. Wang, P. Liu, and L. Shi. Minimizing completion time for order scheduling: Formulation and heuristic algorithm. *IEEE Transactions on Automation Science and Engineering*, 14(4):1558–1569, 2017.

[151] Siemens Digital Industries Software. Process simulate documentation. Available online at: `https://docs.plm.automation.siemens.com/tdoc/tecnomatix/15.0.2/PS_TC`.

[152] A. Sklar and N. Sarter. Good vibrations: Tactile feedback in support of attention allocation and human-automation coordination in event-driven domains. *Human Factors*, 41(4):543–552, 1999.

[153] L. Sui, L. Miao, and Z. Li. Human body action recognition based on bone information by kinect. In *2017 International Conference on Computer Technology, Electronics and Communication (ICCTEC)*, pages 1076–1081, 2017.

[154] A. Sundin and L. Medbo. Computer visualization and participatory ergonomics as methods in workplace design. *Human Factors and Ergonomics in Manufacturing & Service Industries*, 13(1):1–17, 2003.

[155] K. Tamaki and S. Y. Nof. Design method of robot kitting system for flexible assemble. *Robotics and Autonomous Systems*, 8:255–273, 12 1991.

[156] J. Teiwes, T. Bänziger, A. Kunz, and K. Wegener. Identifying the potential of human-robot collaboration in automotive assembly lines using a standardised work description. In *2016 22nd International Conference on Automation and Computing (ICAC)*, pages 78–83, Sep. 2016.

[157] P. Tormene, T. Giorgino, S. Quaglini, and M. Stefanelli. Matching incomplete time series with dynamic time warping: an algorithm and an application to post-stroke rehabilitation. *Artificial Intelligence in Medicine*, 45(1):11 – 34, 2009.

[158] P. Tsarouchi, S. Makris, and G. Chryssolouris. On a human and dual-arm robot task planning method. *Procedia CIRP*, 57:551 – 555, 2016. Factories of the Future in the digital environment - Proceedings of the 49th CIRP Conference on Manufacturing Systems.

[159] V. Villani, F. Pini, F. Leali, and C. Secchi. Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications. *Mechatronics*, 2018.

[160] N. N. Vo and A. F. Bobick. From stochastic grammar to bayes network: Probabilistic parsing of complex activity. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2641–2648, 2014.

[161] F. von Drigalski, C. Nakashima, Y. Shibata, Y. Konishi, J. C. Triyonoputro, K. Nie, D. Petit, T. Ueshiba, R. Takase, Y. Domae, T. Yoshioka, Y. Ijiri, I. G. Ramirez-Alpizar, W. Wan, and K. Harada. Team o2as at the world robot summit 2018: an approach to robotic kitting and assembly tasks using general purpose grippers and tools. *Advanced Robotics*, 34(7-8):514–530, 2020.

[162] L. Wang, Y. Huang, X. Chen, and C. Zhang. Task scheduling of parallel processing in cpu-gpu collaborative environment. In *2008 International Conference on Computer Science and Information Technology*, pages 228–232, 2008.

[163] X. Wang, R. Wang, and F. Zhou. Fingertips detection and hand tracking based on curve fitting. In *2014 7th International Congress on Image and Signal Processing*, pages 99–103, Oct 2014.

[164] Z. Wang, J. Kinugawa, H. Wang, and K. Kazahiro. A human motion estimation method based on gp-ukf. In *2014 IEEE International Conference on Information and Automation (ICIA)*, pages 1228–1232, 2014.

[165] R. Wilcox, S. Nikolaidis, and J. Shah. Optimization of temporal dynamics for adaptive human-robot interaction in assembly manufacturing. *Robotics*, 8:441, 2013.

[166] W. K. H. Wu, A. C. S. Chung, and H. H. N. Lam. Multi-resolution lc-ms images alignment using dynamic time warping and kullback-leibler distance. In *2012 19th IEEE International Conference on Image Processing*, pages 1681–1684, Sept 2012.

[167] A. M. Zanchettin, A. Casalino, L. Piroddi, and P. Rocco. Prediction of human activity patterns for human-robot collaborative assembly tasks. *IEEE Transactions on Industrial Informatics*, pages 1–1, 2018.

[168] A. M. Zanchettin, N. M. Ceriani, P. Rocco, H. Ding, and B. Matthias. Safety in human-robot collaborative manufacturing environments: Metrics and control. *IEEE Transactions on Automation Science and Engineering*, 13(2):882–893, April 2016.

[169] A. M. Zanchettin, E. Lotano, and P. Rocco. Collaborative robot assistant for the ergonomic manipulation of cumbersome objects. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6729–6734, 2019.

[170] A. M. Zanchettin and P. Rocco. Probabilistic inference of human arm reaching target for effective human-robot collaboration. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017.

[171] A. Zanella, A. Cisi, M. Costantino, M. Di Pardo, G. Pasquettaz, and G. Vivo. Criteria definition for the identification of hrc use cases in automotive manufacturing. *Procedia Manufacturing*, 11:372 – 379, 2017. 27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy.

[172] M. Zinn, O. Khatib, B. Roth, and J. K. Salisbury. Playing it safe [human-friendly robots]. *IEEE Robotics Automation Magazine*, 11(2):12–21, June 2004.