



POLITECNICO
MILANO 1863

POLITECNICO DI MILANO
DEPARTMENT OF ELECTRONICS INFORMATICS AND BIOENGINEERING
DOCTORAL PROGRAMME IN INFORMATION TECHNOLOGY

ALLOWING A REAL COLLABORATION BETWEEN
HUMANS AND ROBOTS

Doctoral Dissertation of:
Andrea Casalino

Supervisor:
Prof. Paolo Rocco

Tutor:
Prof. Luca Bascetta

The Chair of the Doctoral Program:
Prof. Barbara Pernici

Cycle XXXII

*A Chiara
alla mia
e alla sua famiglia*

«You don't completely control anything,
except from your thoughts.»

— Rene Descartes

«I tell you: one must still have chaos in one,
to give birth to a dancing star.»

— Friedrich Nietzsche

«Do not judge wrong what you don't know,
take the opportunity to understand!»

— Pablo Picasso

Acknowledgements

First of all, I would like to thank Prof. Paolo Rocco for giving me the possibility to develop a PhD in Robotics. Working at the Merlin lab was a great experience, full of opportunities that will have a big impact on my future path. I want to thank Prof. Andrea Zanchettin for the constructive feedbacks he gave throughout my entire academical experience and I would like to thank also Prof. Luigi Piroddi, for his suggestions and interesting meetings.

My PhD experience would have not been the same without the interactions I had with my office mates, Riccardo Maderna, Davide Nicolis, Renzo Villa, Matteo Parigi Polverini, Roberto Rossi and Nicolò Tomiati. A special mention must be done for Costanza Messeri and Davide Bazzi, who were firstly two students I supervised and secondly two valuable mates, who had good times with me in the office till late.

This work was also made possible by the efforts spent by the master students that collaborated with me, Luka Juricic, Silvia Quartulli, Nicola Massarenti, Eleonora Mazzocca, Maria Grazia Di Giorgio, Filippo Cividini, Alberto Brameri, Sebastain Guzman, Marco Maiocchi and Roberto Cherubin. Working with each of them was constructive and positive.

Finally, I would like to thank Prof. Alessandro De Luca and Prof. Olaf Stursberg for reviewing this thesis and providing valuable suggestions for improving this work.

Abstract

Collaborative Robotics is emerging as one of the most active lines of research in automation. This term indicates a group of methodologies and techniques that allow robots to work side by side with humans. The human should execute highly cognitive tasks, like e.g. assembly operations that could be too difficult to fully automatize, while robots have to both undertake autonomous operations and assist the humans in many ways. The combination of the human flexibility and the robots efficiency can significantly improve the production process. This level of interaction requires at least the sharing of a common space. This topic has attracted the interest of many researchers in the recent years and many controlling algorithms have been developed to allow a safe coexistence of humans and robots. In this context, tracking the human motion is of paramount importance. Then, a safe motion controller can optimize the trajectory of the robots with the aim of dodging humans. We can state that the safe interaction of humans and robots, while performing disjoint tasks, is something achieved.

For this reason, the aim of this thesis was to study more in depth the collaboration between human and robot. In particular, this was done by focusing on industrial contexts, where typical applications are collaborative assemblies (or co-assemblies). In such scenarios, humans and robots have to execute alternating tasks, with the aim of realizing a set of possible finite products. The robots have to adapt and synchronize with the humans, since the collaboration was conceived as human-centric: it's the human that regulates the interaction. To this purpose, robots have to interpret the human intentions as well as to predict them in order to take the best actions for providing a reliable assistance. Such an interpretation is possible only through increased cognitive capabilities. For this, sensors can be exploited to produce a large amount of data describing the workspace surrounding a robot, which are at a second stage interpreted by machine learning techniques.

Within the above scenario, this thesis proposes the three following main contributions:

- introduce algorithms and methodologies for inferring the current action that a human operator is undertaking, from the simplest ones, as for instance those for

reaching tools or objects, to the more complex ones, as performing a screwing. Two inferring algorithms will be proposed. The first one analyzes the motion of the hands, as well as the orientation of the gaze, for inferring the next reaching target of an operator adopting a Gaussian Mixture model. The second algorithm takes into account the motion of the entire body and is based on Markov Random Fields.

- predict the actions performed by human operators in the near-far future. The proposed solution is made of two parts. One models the sequence of operations, while the other one the time durations. The first kind of modelling can be done by making use of two alternative approaches, one based upon Higher Order Markov model and the other one based on the construction of a Suffix Tree.
- optimally schedule the operations assigned to robots, with the aim of assisting the human and minimizing the inactivity times. This must be done by properly taking into account the time variability of human actions. All the developed scheduling approaches consider a particular class of Timed Petri Nets, specifically derived for describing collaborative tasks. The optimal commands to be sent to robots are extracted from a reachability tree representing many alternative evolutions of the system.

Although collaborative robots are intrinsically safe, an additional minor objective of the thesis was to investigate how to optimally control their motion in collaborative cells. This problem was solved as similarly done for the aforementioned scheduling, i.e. by taking into account a prediction of the human motion.

All the proposed methodologies were tested in realistic robotic co-assemblies.

Sommario

LA Robotica Collaborativa si sta affermando come una delle linee di ricerca maggiormente studiate nell'ambito dell'automazione. Il termine indica quel gruppo di tecniche e metodologie che permettono ai robot di lavorare fianco a fianco degli uomini, ai quali dovrebbero essere destinati i compiti altamente cognitivi, per cui una completa automatizzazione risulterebbe difficile o quasi del tutto impossibile. Invece, i robot dovrebbero allo stesso tempo compiere delle azioni autonome e assistere gli uomini in vari modi. La combinazione data dalla flessibilità umana e dall'efficienza dei robot consente di migliorare notevolmente i processi produttivi. Un tale livello di interazione richiede quantomeno la coesistenza sicura in uno spazio condiviso. Questa tematica ha attratto gli sforzi di molti ricercatori nel recente passato e molte strategie di controllo per i robot sono state sviluppate con il fine di garantire una sicura coesistenza fra uomini e robot. In un tale scenario, riuscire a monitorare nel tempo i movimenti degli operatori umani diventa di primaria importanza. Infatti, analizzando il moto degli umani presenti nella cella robotica, i controllori possono ottimizzare la traiettoria dei manipolatori robotici allo scopo di evitare pericolose collisioni. Si può affermare con buona sicurezza che l'ottenimento di un'interazione sicura fra operatori e robot, quando questi svolgono operazioni autonome, è un risultato pienamente raggiunto.

Per questa ragione, lo scopo di questa tesi è stato quello di studiare con maggiore attenzione la tematica legata alla collaborazione fra uomini e robot. Nello specifico, ci si è concentrati su scenari industriali, dove applicazioni tipiche sono gli assemblaggi collaborativi (detti anche co-assemblaggi). In queste situazioni, operatori e robot devono compiere delle azioni che si alternano fra loro, con l'obiettivo di realizzare un certo numero di possibili prodotti finiti. I manipolatori devono adattarsi e sincronizzarsi con gli operatori umani, dato che la collaborazione è stata concepita in questa tesi come umano-centrica: deve essere l'operatore a regolare l'interazione con il robot. Per ottenere una tale sinergia, i robot devono poter essere in grado di interpretare le reali intenzioni correnti degli operatori e allo stesso tempo prevedere quelle future, allo scopo di poter decidere l'azione ottimale da svolgere nel presente per assistere nella maniera migliore gli umani. L'interpretazione del comportamento umano è possibile solo attraverso migliorate capacità cognitive. A tal proposito i sensori di cui può essere

dotato un manipolatore sono oggi in grado di produrre una grande quantità di dati, che deve essere in seconda battuta interpretata attraverso tecniche di machine learning.

Nel contesto descritto, questa tesi presenta i seguenti contributi:

- proporre algoritmi e tecniche per comprendere l'azione svolta da un operatore umano, basandosi sulla sola analisi del suo movimento nel tempo, da quelle più semplici, come ad esempio afferrare degli oggetti in dei contenitori, a quelle più complesse, come eseguire l'avvitamento di alcuni componenti. Due algoritmi distinti verranno proposti per affrontare questo problema. Il primo analizza il moto delle mani e l'orientamento del volto, per stimare la destinazione dei movimenti di un operatore. Tale stima viene svolta attraverso l'uso di un modello basato su misture di Gaussiane. Il secondo algoritmo proposto considera il movimento di tutto il corpo ed è basato sull'analisi di un Markov random field.
- predire le azioni che verranno svolte dall'operatore nel breve-lungo periodo. La soluzione proposta è costituita da due parti principali. La prima modella la sequenza logica delle operazioni, mentre la seconda descrive le loro possibili durate temporali. La modellazione logica delle sequenze può essere svolta tramite due approcci alternativi, uno basato su un modello Markoviano di ordine superiore e l'altro basato sulla costruzione e sul mantenimento di un albero di suffissi.
- schedulare nella maniera più ottimale le operazioni assegnate ai robot collaboranti, con lo scopo di assistere l'umano e minimizzare i tempi di inattività. Questa pianificazione di attività dovrà essere svolta tenendo in conto la variabilità associata alla durata delle operazioni svolte dagli operatori umani. Le tecniche di schedulazione sviluppate sono tutte basate sull'utilizzo di una particolare classe di reti di Petri temporizzate, appositamente introdotta per descrivere scenari collaborativi.

Sebbene i robot collaborativi siano intrinsecamente sicuri, un obiettivo secondario della tesi ha riguardato lo studio di tecniche innovative di controllo del moto in ambiti di collaborazione. Le tecniche proposte risultano essere migliorative in quanto, similmente a quelle di scheduling, prendono in considerazione una predizione del comportamento futuro degli umani (in questo caso in termini di movimenti futuri).

Tutte le tecniche sviluppate sono state sperimentate in situazioni realistiche di co-assemblaggio.

Outline

The future production plants will see more and more the presence of robots, performing a large variety of actions, from simple moving tasks to complex manipulations. In the past, the most active research lines were mainly devoted to develop algorithms for motion or force control, while the current trend is trying to provide robots with some more sophisticated cognitive capabilities. Indeed, robots are becoming smarter, thanks to an increased computational power. Therefore, artificial intelligence and machine learning are entering more and more into robotic researches.

This is also due to the fact that robots are no longer conceived as something that will completely replace the human workers. On the opposite, robots will enter into the production lines (and not only there) in order to become valuable assistants of the humans, helping them in many ways. At the same time, robots will have a certain level of autonomy, in order to prevent humans from doing something wrong or anticipate the human needs. This kind of tasks can be accomplished by robots only with an increased cognitive capability. The field studying all this aspects is the Collaborative Robotics and is progressively attracting the interest of the robotic community. The way Collaborative Robotics conceive robots is far from the old paradigm adopted, where robots were physically segregated from humans for safety reasons.

The first step toward a collaboration was done by allowing the robots to share the space with the humans. Safety was ensured by developing many dedicated motion control strategies. Clearly, this was only an initial step towards a real human-robot collaboration, which only in very recent times is starting to be studied. Collaborate with a human mate is a non trivial task for a robotic device. Indeed, robots should be able to behave as humans naturally do, interpreting the other workers actions and forecasting the future ones. This is actually what make Collaborative Robotics a tough topic.

The interpretation of the human behaviour can be tackled by exploiting advanced sensors, providing the robots a huge amount of data to process. Finding a way to handle such data in a fast and reliable way is becoming crucial.

The main goal of this thesis was to develop algorithms and strategies able to allow an efficient collaboration between humans and robots, with a particular attention to industrial contexts, where these agents have to alternate and synchronize for performing structured tasks, as for instance assemble some components. The approaches devel-

oped allow robots to adapt, in many ways, to the human mates while at the same time they provide the proper assistance. All the proposed methods are validated in various realistic use cases of collaboration.

Thesis Contributions and Organization

In this thesis, the following main contributions are given:

- the proposal of algorithms and methodologies for inferring the current action that a human operator inside a robotic cell is undertaking, from the simplest ones as for instance those for reaching tools or objects, to the more complex ones as for instance performing a screwing.
- the study of an approach for predicting the human actions in the near-far future.
- the introduction of innovative scheduling algorithms for planning the operations assigned to robots, with the aim of assisting human mates and minimizing the inactivity times. All the developed scheduling approaches consider a particular class of Timed Petri Nets, specifically derived for describing collaborative tasks.
- the proposal of innovative approaches for the motion control of collaborative robots (cobots).

This manuscript is organised as follows:

Part I will focus on the problem of inferring the human behaviour, mainly in terms of: what is the current action that the human is undertaking, Chapter 3 and 4; what actions will be executed in the future, Chapter 5. Chapter 3 exploits a Gaussian Mixture for building a model of the human intention, with the aim of inferring the current goal of the human. This is done by mainly considering the motion of the operator's wrists. A second approach for the intention inference is proposed in Chapter 4, where Markovian Random Fields are exploited for segmenting the actions performed by the human in the recent past. The motion of the entire body of an operator is considered in this case.

Regarding the prediction topic, the strategy proposed in Chapter 5 is made of two main parts. The first one models the sequence of actions, without considering temporal implications. To this purpose, two possible approaches for performing such modelling will be proposed: the one in Section 5.1 exploits higher order Markov models, while the one in Section 5.2 makes use of suffix trees. Both these models can be used for predicting the time to see again a certain human activity, Section 5.3.

The predictions made by the algorithms in Chapter 5 are exploited for optimally schedule the actions assigned to robots, as extensively detailed in Part II. Some theoretical aspects regarding the scheduling of collaborative cells will be discussed in Chapter 6. The concepts discussed will be used for describing the principles behind the developed scheduling approaches, which are detailed in Chapter 7. In particular, three approaches will be proposed in Sections 7.2, 7.3 and 7.4. They are all based on Timed Petri Nets. The ones in Section 7.2 and 7.4 propagate the uncertainties characterizing the system, due mainly to the durations of the human actions, in a closed form; while the approach

in Section 7.3 makes use of a numerical method. Several use cases of realistic co-assemblies will be adopted for validating these techniques.

Finally, Part III will focus on the safe control of collaborative robots. Although this topic is well studied, new algorithms were developed applying some of the concepts adopted for the scheduling approaches discussed in the previous Part. Chapter 8 will propose a reactive approach in Section 8.1 and a proactive one in Section 8.2. One key aspect of the above methodologies is the possibility to keep track of the human motion during time. In this context, managing occlusions became crucial and is the aim of the approach proposed in Chapter 9.

Concluding considerations will be provided in Chapter 10.

Publications

This thesis is based on the following publications.

International Journals:

1. Andrea Casalino, Andrea Maria Zanchettin, Luigi Piroddi, and Paolo Rocco "Optimal scheduling of human-robot collaborative assembly operations with time petri nets". **IEEE Transactions on Automation Science and Engineering**, available on line, DOI: 10.1109/TASE.2019.2932150
2. Andrea Maria Zanchettin, Andrea Casalino, Luigi Piroddi, and Paolo Rocco. "Prediction of human activity patterns for human-robot collaborative assembly tasks". **IEEE Transactions on Industrial Informatics**, Vol. 15, No. 7, pp. 3934–3942, July 2019
3. Andrea Casalino, Costanza Messeri, Maria Pozzi, Andrea Maria Zanchettin, Paolo Rocco, and Domenico Prattichizzo. "Operator awareness in human–robot collaboration through wearable vibrotactile feedback". **IEEE Robotics and Automation Letters**, Vol. 3, No. 4, pp. 4289–4296, October 2018

International Conferences:

1. Andrea Casalino, Eleonora Mazzocca, Maria Grazia Di Giorgio, Andrea Maria Zanchettin, and Paolo Rocco. "Task scheduling for human–robot collaboration with uncertain duration of tasks: a fuzzy approach". **IEEE International Conference on Control, Mechatronics and Automation (ICCMA 2019)**, Delft (The Netherlands), November 2019.
2. Andrea Casalino, Alberto Brameri, Andrea Maria Zanchettin, and Paolo Rocco. "Adaptive swept volumes generation for human–robot coexistence using gaussian processes". **IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2019)**, Macau, November 2019.
3. Andrea Casalino, Davide Bazzi, Andrea Maria Zanchettin, and Paolo Rocco. "Optimal proactive path planning for collaborative robots in industrial contexts". **IEEE International Conference on Robotics and Automation (ICRA 2019)**, Montréal (Canada), May 2019.

-
4. Andrea Casalino, Filippo Cividini, Andrea Maria Zanchettin, Luigi Piroddi, and Paolo Rocco. "Human–robot collaborative assembly: a use–case application". **16th IFAC Symposium on Information Control Problems in Manufacturing (INCOM 2018)**, Bergamo (Italy), June 2018
 5. Andrea Casalino, Sebastian Guzman, Andrea Maria Zanchettin, and Paolo Rocco. "Human pose estimation in presence of occlusion using depth camera sensors, in human–robot coexistence scenarios". **IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2018)**, Madrid (Spain), October 2018, pp. 6117–6123.
 6. Andrea Casalino, Andrea Maria Zanchettin and Paolo Rocco. "Enhance the collaboration between human and robots through activity prediction and reactive scheduling". **12th International Workshop on Human–Friendly Robotics (HFR 2019)**, Reggio Emilia (Italy), October 2019

and on the following submitted material:

1. Andrea Casalino, Nicola Massarenti, Andrea Maria Zanchettin, and Paolo Rocco. "Predicting the human behaviour in human–robot co–assemblies: an approach based on suffix trees". **IEEE International Conference on Robotics and Automation (ICRA 2019)**, Paris (France), May 2020.
2020 International Conference on Robotics and Automation (ICRA), IEEE, November 2020.

Finally, the following publications contain relevant results, that are not covered in the doctoral dissertation:

1. Andrea Casalino, Andrea Maria Zanchettin, and Paolo Rocco. "MT–RRT: a general purpose multithreading library for path planning". **IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2019)**, Macau, November 2019.
2. Andrea Casalino, Paolo Rocco, and Maria Prandini. "Hybrid control of manipulators in human–robot coexistence scenarios". **American Control Conference (ACC 2018)**, Milwaukee (USA), June 2018, pp. 1172–1177
3. Riccardo Maderna, Andrea Casalino, Andrea Maria Zanchettin, and Paolo Rocco. "Robotic handling of liquids with spilling avoidance: a constraint–based control approach". **IEEE International Conference on Robotics and Automation (ICRA 2018)**, Brisbane (Australia), May 2018, pp. 7414–7420
4. Andrea Casalino, Andrea Maria Zanchettin, and Paolo Rocco. "Online planning of optimal trajectories on assigned paths with dynamic constraints for robot manipulators". **IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2016)**, Daejeon (Korea), September 2016.

Contents

1	Introduction and State of the Art	1
I	Understanding the human behaviour	7
2	Preamble	9
3	Understanding the human intentions	11
3.1	Estimating the current intended goal	11
3.1.1	Background	12
3.1.2	Proposed approach	13
3.1.3	Operator awareness through intention recognition	18
4	Segmenting the human actions by analysing the upper body motion	27
4.1	Evolving factor graphs for segmenting the human actions	27
4.1.1	Modelling the correlation existing between the observations and the human actions	28
4.1.2	Modelling the sequence of actions	30
4.1.3	Segmenting the actions on a fixed window of observations	31
4.1.4	Segmenting the actions on evolving windows	32
4.1.5	Tuning of the parameters	34
4.2	Experiments	34
4.2.1	Results	36
5	Predicting the future activities	41
5.1	Higher Order Markov Model	42
5.2	Suffix Tree	46
5.2.1	The predictive model	46
5.2.2	Comparison of the proposed predictive models	52
5.3	Evaluating waiting times	54
5.4	Performance comparison	56
5.4.1	Use case a	56

II	Assistive scheduling	61
6	Scheduling of the robotic actions	63
6.1	Petri Nets as scheduling tools	65
6.2	Temporal Petri Nets: main concepts	66
6.2.1	Reachability Tree	67
6.3	Task specification	75
6.3.1	Modeling robot actions	77
6.3.2	Modeling human actions	77
6.3.3	Modelling collaborative actions	78
6.3.4	Modelling mobile robots	78
7	Scheduling approaches	81
7.1	Receding horizon scheduling	81
7.2	Best scenario approach	83
7.2.1	Description of the use case	83
7.2.2	Selection of the best plan	85
7.2.3	Remarks	88
7.3	Monte Carlo scheduling	89
7.3.1	Selection of the best plan	89
7.3.2	Off line simulations	95
7.4	Fuzzy scheduling	96
7.4.1	Selection of the best plan	99
7.5	Validating experiments	100
7.5.1	Use case a	100
7.5.2	Use case b	101
7.5.3	Use case c	110
III	Motion control of cobots	115
8	Safe control of cobots	117
8.1	The reactive approach	120
8.1.1	Background about Swept Volumes generation	120
8.1.2	Gaussian Processes for Swept Volumes generation	121
8.1.3	Experiments	126
8.2	The proactive approach	130
8.2.1	Modelling the human-robot collaboration	130
8.2.2	Probabilistic description of the human motion	131
8.2.3	Proactive path planning	134
8.2.4	Experiments	136
9	Occlusions handling	143
9.0.1	Representation of the human pose	144
9.1	Dealing with occlusions in the human silhouette	145
9.1.1	Single point tracking	145
9.1.2	Human pose tracking	148
9.1.3	Experiments	151

10 Conclusions	155
Appendices	159
A Generalities about learning	161
B Expectation Maximization	163
B.0.1 Learning of Gaussian Mixture Models	165
C Factor graphs	169
C.0.1 The message passing algorithm	172
C.0.2 Learning the weight parameters	179
C.0.3 Learning structures with shared weights	182
D Fuzzy theory	185
E Gaussian Processes	189
E.0.1 Scalar case	189
E.0.2 Vectorial case	192
Bibliography	209

CHAPTER 1

Introduction and State of the Art

Up to the recent past, the paradigm universally adopted for industrial robotics provided for the strict segregation of robots in protected environments, adopting fences or optical barriers. The robots were mainly conceived to work alone in highly structured environments, replacing humans in carrying out activities that were repetitive, dangerous or requiring high precision. Only recently, the potential benefits of a real collaboration between humans and robots have gained the attention of roboticists [45], [62], [1], [25] and [32]. Such interest is mainly motivated by the Industry 4.0 paradigm [113], which considers as a fundamental pillar the massive presence of robots in production plants, cooperating with humans. Indeed, there are still many actions that also the more evolved robots are not able to perform without exploiting high-tech sensors or dedicated end-effectors, [63]. Therefore, it seems natural to let humans and robots share common tasks: highly cognitive actions are undertaken by humans, while those requiring high precision and repeatability are performed by robots. This framework is particularly appealing in that it allows for a higher variable production mix with respect to pure automatized and dedicated assembly lines [10]. This aspect is particularly relevant for small and medium sized enterprises (SMEs). On the other hand non-industrial applications are rapidly emerging such as healthcare and domestic robotics. Collaborative applications must be accurately designed in order to get the maximum possible benefits from the human-robot interaction. Regarding the latter aspect, [137] revised some automotive use cases, with the aim of highlighting the principles for a profitable collaborative robotic implementation.

Safe coexistence of humans and robots

The starting point for a real cooperation is clearly the coexistence in a shared space. This kind of issues have been heavily studied in the past years, since were the main fo-

cus of almost every research in the field of Human Robot Collaboration (HRC), [113]. Indeed, standard industrial robots are able to move at a high speed and they could severely injure humans [44]. For this reason, a new generation of robots, called cobots, has been specifically designed to allow for a safe interaction with humans¹. They are lighter, without edges, sometimes covered with paddings to damp the effects of impacts, and often equipped with kinematic redundancy [45], i.e. they have a number of joints greater than the strictly necessary ones in order to perform more dexterous motions able to let a task advancing while at the same time preserving the safety of humans [21].

This new designing paradigm has also been endowed with new motion control techniques, [105], [36], based on the use of sensors perceiving the scene, that track the motion of the human operators. To this purpose, high-visibility industrial clothing detection strategies based on RGB and IR cameras have been proposed in [88], while [92] introduced the concept of "smart floor" by adopting pressure-sensitive sensors to keep track of human motions. RGB-D cameras have been exploited in [89].

Consider a manipulator whose task is reaching some target configuration. If no human operator is within its workspace, then, the manipulator can move following a nominal point-to-point trajectory, which is computed ignoring the presence of humans so as to be optimal for the assigned task. When a human gets too close to the manipulator, an action has to be taken. An option is to interrupt the task execution and slow down/arrest the manipulator [134]. Alternatively, a corrective trajectory can be planned online with the aim of dodging the human and, at the same time, keep driving the manipulator to the desired target position [106]. Also a combination of the previous strategies is possible as done in [85]. Such motion controllers were developed both for cobots, which are intrinsically safe, as well as standard industrial robots, see [36], [64] and [116]. [79] focused also in the safety aspect, but considering service robotics contexts. Since the trajectories of the robots may be altered from the nominal one, the time required to complete a robotic task becomes an aleatory quantity. For this reason Pellegrinelli *et al.* [97] proposed a way to estimate such completion time, considering a probabilistic time independent description of the space occupied by humans.

In all works cited so far, robots were conceived as something that should interfere as less as possible with the activities simultaneously undertaken by the human populating the same cell. While the safe coexistence of humans and robots, performing independent tasks, seems to have a well-established literature, the level of collaboration needs still to be explored.

Understanding the behaviour of the human

More recently, researchers have started working on methods to allow a fluent collaboration between humans and robots. The possibility for a properly instrumented robotic device to understand and somehow predict humans' intentions is now considered as important as safety. This kind of activities are made possible by providing the robots the proper cognitive capability. Indeed, artificial intelligence and machine learning are entering more and more into modern robotics. In the last years, collaborative robots have become faster, smarter, more accurate and reliable, even though challenges remain in adaptability [130], decision making and robustness to changing, especially when a con-

¹See e.g. <http://blog.robotiq.com/collaborative-robot-ebook>.

tinuous interaction with a human mate is required. Cognitive algorithms allow robots to understand the behaviour of their fellow human team-mates in order to anticipate, and adapt to them, [52].

In this context, a crucial role is played by vision sensors, which give to the robot the sense of sight. The analysis of the human motion is one of the most important feature to consider for understanding an operator intentions. For instance, in [38] the intuition that people tend to follow efficient trajectories rather than random paths is exploited. The proposed strategy learns common destinations within the environment by clustering training examples of trajectories.

Gaussian Processes (GPs) have been also proved to be effective in predicting the human motion. Such a prediction can be used also for compensating occlusions and noise, as done in [125]. The method proposed in [129] exploits a GP in conjunction with an Unscented Kalman Filter for motion tracking, while [73] adopts GPs for activity classification.

Even though, the analysis of the human motion is paramount, the correct way to approach the human intention estimation problem is through a multimodal perspective: any kind of information should be exploited. For example [71] introduced an approach based on game theory to estimate the objective of the human, through the measured interaction force. In this context, many results have been reported showing the increasing capability of robots to semantically interpret their human fellows. In [61] a method based on conditional random files (CRF) is used by the robot to anticipate its assistance. In Luo et al. [77] Gaussian Mixture Models (GMMs) are used to predict human reaching targets. HMMs have been also adopted in [69] to recognise and label sequences of activities based on occupancy grids. The approach capitalises on the multi-modal perception algorithm discussed in [68]. In [111] human intention is inferred by combining expectation-maximisation (EM) algorithms and an online model learning strategy.

Once the intention of the human has been recognized, the robot should plan a complementary assistive action. The development of anticipatory behaviors has been extensively investigated in the literature, and several results have highlighted the corresponding benefits, see *e.g.* [3, 50]. This behavior, which is also referred to as *proactive behavior*, has been also applied to multi-agent scheduling for job shops in [76]. Hawkins et al. [47] developed an inference mechanism based on Hidden Markov Models (HMMs) allowing the robot to predict when particular robot actions would be appropriate, based on the current state of the human worker. Other approaches based on neural networks [95], or Dynamic Bayesian Networks (DBN) [3, 65, 78] have been developed to investigate the mutual adaptation of hybrid human-robot teams by modelling motion patterns.

At the end of the intention inference process, it could be beneficial to inform the human about what the robot device understood of his or her real intention. Recent papers have started to motivate the need for mutual understanding [33, 100]. The technology advancements have introduced a plethora of new methodologies to increase the awareness of the operator during the collaboration with a robot. For example, in [81] an augmented reality (AR) has been introduced to support the human in collaborative assembly operations. Signal lights and their optimal positioning have been addressed in [56] to inform the human operator about the status of the robot. Verbal feedback, *i.e.* the most natural (for the human) interaction modality, has been addressed in [122]

showing its capability to improve the performance in HRC.

Not only to infer the current human action is important, but also to predict the ones of the near-far future is relevant for planning in the medium-long term robotic actions. To this aim Li et al. [70] proposed a framework based on variable order Markov models to predict activity patterns using causal relationships between actions. Variable order stochastic automata were before used for predictions in [114]. Other works focusing on high-order stochastic processes, but not applied to robotics, can be found in [26, 103, 104]. In [59] a model for the prediction of the worker's arrival time at a certain working position has been introduced and applied within an automotive assembly process. In [47] a planning algorithm was developed to select robot-actions that minimize the expected waiting cost based upon the distribution over predicted human-action timings.

Assistive scheduling

Typical applications of human-robot cooperation are collaborative assemblies. In such contexts, the prediction of the sequence of future human activities is crucial. Indeed many assigned tasks are usually required to obtain a single finite product, having many precedence constraints. For this reason, the behaviour of the robots influences the one of the human, possibly leading to situations where the operator is forced to wait. The optimization of the robotic action sequence should be done through a multi agent scheduling approach, accounting for the predicted human actions.

The task allocation problem can be solved prior to scheduling, when assuming a static approach, or simultaneously. Chen et al. [23] describe a genetic algorithm for a collaborative assembly station which minimises the assembly time and costs. In [108], a trust-based dynamic subtask allocation strategy for manufacturing assembly processes has been presented. The method, which relies on a Model Predictive Control (MPC) scheme, accounts for human and robot performance levels, as well as for their bilateral trust dynamics. Furthermore, in [54], the authors proposed a multi-layered planner for task allocation, sequencing and execution using AND/OR graph and A^* graph search. Similarly, in [124] Tsarouchi et al. proposed an intelligent decision-making method that allows human-robot task allocation according to their capabilities. By taking inspiration from real-time processor scheduling policies, Gombolay et al. [41] developed a multi-agent task sequencer, where task specifications and constraints are solved using a MILP (Mixed Integer Linear Programming) algorithm, showing near-optimal task assignments and schedules. A similar approach has been also derived in [118]. Bruno *et al.* proposed to formalize the task allocation as a classification problem: novel tasks are assigned to agents according to a certain training set [10]. Finally, [93] proposes a task assignment method, based on the exploration of possible alternatives, that enables the dynamic scheduling of tasks to available resources between humans and robots.

Regarding the pure scheduling of multi-agent systems, not necessarily humans and robots, a rich literature can be found. In [35] a theory on correlation scheduling is developed, while [34] formalises the problem as a graph search in the space of all possible sequences of actions. Moreover, [76] proposed an approach that can be robust with respect to uncertain durations of tasks. All these works consider all agents to be controllable. Indeed, a centralised approach is followed to compute a global plan, which is then dispatched to every agent. Decentralized approaches were also developed by split-

ting some global task into smaller ones and synthesising some local supervisors for agents [29] with a top-down approach. With the aim of reducing the worker's waiting time, Kinugawa et al. [59] developed an online learning algorithm to feed an adaptive task scheduling system for the collaborative robot. More recently, Bogner *et al.* proposed a pure integer linear programming (ILP) formulation for optimal scheduling of robots that are shared among multiple collaborative stations, using some heuristics to obtain a solution within an acceptable time [8]. [48,49] focused on cycle time optimization based on deterministic marked graphs, proposing an iterative heuristic algorithm to find a proper schedule for a generic manufacturing system.

Regarding HRC assemblies, the collaborative cell can be designed according to a specific assembly sequence, as done in [101]. Then a centralised global plan can be computed by exploiting one of the above techniques, which is then executed in real time by humans and robots. Many alternative plans are computed in [124], where a dynamic task allocation of activities is also possible, by taking into account the different capabilities of agents (robots and humans). In [24], the scheduling problem is solved using a Generalised Stochastic Petri Net. The selection of the optimal plan takes into account the amount of time for which the agents remain inactive, waiting for the activation of some tasks. However, all the aforementioned works model the human operators no more than highly cognitive manipulators whose actions can be scheduled in an optimal way.

The factories of the future will adapt their behaviours, reacting to rapidly changing production plants. In this scenario, robots can no longer be adopted to simply accomplish repetitive tasks. Instead, humans and robots will both adapt and synchronize in many ways, collaborating to accomplish common tasks. Human-robot turn-taking models based on time Petri nets have been presented in [22], emphasizing their ability to handle multiple resources and actions, such as speech, gaze, gesture, and manipulation.

Finally attention must be paid to the modelling of co-assemblies. De Mello *et al.* proposed a representation of automated assembly sequences using AND/OR graphs [83]. In [115], a representation of assembly sequences using an *ad hoc* developed SFC-like language has been developed, while precedence graphs were used in [112]. The adoption of queuing theory in manufacturing processes is also discussed in [42]. However, a single tool or formalism is unlikely to simultaneously fit the need to capture both the high-level flow of the process (which is needed from the point of view of the process expert) and the low-level details which are needed for computation and optimization purposes. For example, in [75] a visual programming method has been proposed to automatically translate an assembly process described in terms of action blocks into an equivalent Petri net.

This work will propose an approach that allows and **promotes an active collaboration between industrial (collaborative) robots and human operators**. Remarkably, such an approach will be applied in realistic industrial scenarios, which is something that most of the previous approaches did not. **The robot will be conceived as a smart agent**, able to continuously **interpret the human actions** and needs and consequently **adapt to provide the optimal assistance**, without waiting for explicit human requests. With respect to the cited literature, few information will be assumed as a prior knowl-

edge, since the data collected during the interaction will be used to progressively refine a model describing the human to assist.

An efficient collaboration will be made possible by proposing a **unique framework that integrates** methodologies for solving the following three problems: **understand and predict the human behaviour**, **schedule** in an optimal way the **tasks assigned to robots** for providing the proper assistance and **safely control the motion of the robots**, by taking into account the predicted human behaviour. Such an integrated solution is original in the current state of the art, where prior works typically focus on only one or two of the above aspects.

Part I

Understanding the human behaviour

CHAPTER 2

Preamble

One fundamental goal to achieve for allowing an efficient cooperation between any set of agents, is to endow them with some capabilities to understand the behaviour of the other agents. This is true also when considering human-robot teams. In particular, the human should have the control, allowing him or her to advance a global task, while the robots should assist in the proper way the operator. This could be made possible by interpreting the human behaviour both in terms of:

- Understanding which is the current action that the human is performing.
- Predicting the sequence of future actions.

Both the above problems are non trivial to address, due to the highly stochastic and weakly repeatable nature of human behaviour. We can assume the operator inside a cell as monitored by sensors like depth cameras, able to perceive the motion of the human body during time. According to such data, the algorithms described in this part of

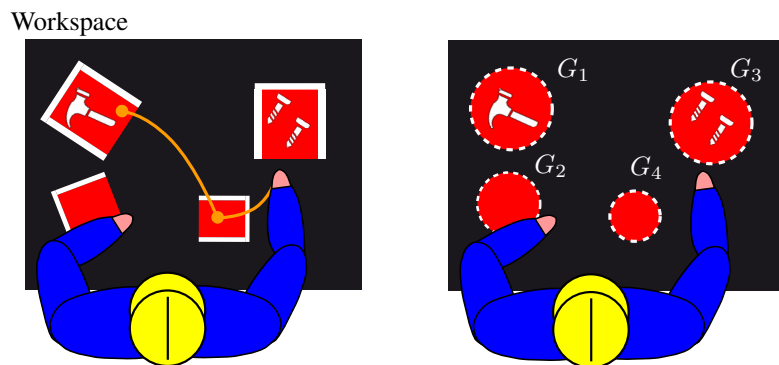


Figure 2.1: *Examples of goals related to actions in A.*

this dissertation will address the above problems.

The prediction of the future human actions will be the starting point for all the scheduling strategies detailed in Part II.

For the rest of this document, the following notation will be adopted. When considering any kind of human-robot collaboration, there are actions that both the human and the robots have to perform for completing a common goal, like for instance performing the assembly of an object. For industrial contexts it is reasonable to assume $\mathcal{A} = \{a_1, \dots, a_m\}$, the set of actions assigned to the human, as finite and known. Actions in \mathcal{A} can be simple operations like taking tools from buffers or moving objects from one place to another one. In such cases, every action $a \in \mathcal{A}$ is associated to a region G_a of the workspace: when an operator's hand enters this area the beginning of the corresponding action can be detected. G_a is assumed as a spherical region with a radius r_a and a center C_a , see Fig. 2.1 as an example.

When having a one to one correspondence between actions and goals, the motion of the hands is critical for inferring the human intention see Section 3.1,5.1 and 5.2. Otherwise, when more complex operations compose set \mathcal{A} , the analysis of the whole human body motion has to be considered, as discussed in Chapter 4.

CHAPTER 3

Understanding the human intentions

To achieve an efficient human-robot collaboration, it is essential for the robot to infer the human's intention and then decide the best action to take. The difficulty of this inference process is proportional to the level of abstraction associated to the possible activities that the operator can undertake. For instance, when monitoring an operator in a robotic cell, we can just infer which tool the human intends to reach, or solve the more difficult problem of inferring what part he/she is going to assemble within a set of possible ones (activity labelling).

The first kind of problem will be the objective of the method proposed in Section 3.1, while the second one could be solved by the approach reported in Chapter 4.

Generally speaking, reducing the level of abstraction, we can make inference in more general contexts, relying on a reduced a priori knowledge. Indeed, the method of Chapter 4 is effective for inferring complex behaviours of the human, even though it requires an extensive initial data set for tuning the inferring model. On the other hand, the approach in Section 3.1 can be applied only to simple reaching actions. The best approach to follow depends on the specific application.

3.1 Estimating the current intended goal

The aim of this Section is to propose a method for inferring the next goal that will be reached by the operator's hand, among a set of possible known ones G_1, \dots, G_m (see the introduction of Part I). In principle, this problem could be solved by monitoring the whole operator's arm trajectory [77]. Indeed, the observed trajectory of the hand, as well as other skeletal points of interest (see also Figure 9.1), are treated as features, which are exploited to classify the trajectory using a learnt Gaussian Mixture Model (GMM) (see Section 3.1.2). After classification, it is possible to forecast the trajectory

on a small future horizon time. The predicted motion is evaluated to make inference about the target location of the human.

The approach detailed in this Section has a similar intent and extends the one proposed in [135], which considered only the operator’s hand trajectory and solved the problem by using a recursive Bayesian classifier. The novel proposed approach considers the estimation of the operator’s gaze direction and additional features not taken into account by [135]. This is actually what allows the method to perform better.

Monitoring the operator’s gaze is something already explored in human-robot interaction (HRI). It is used, for example, to let social robots perform human-like motions. In [139], authors focused on producing coordinated head-arm motions for a humanoid robot with a two degrees-of-freedom head.

Even for human-human interactions, the gaze constitutes an important measure, since it is closely tied to what people are thinking and doing [60]. Motivated by this fact, the estimate of the gaze (face orientation) was introduced to help in the process of inference. In a similar way, Bednarik *et al.*, in [5], developed a classifier that is able to detect when a person observes an intended goal location. The classifier is adopted to avoid the so called Midas touch problem, *i.e.*, considering as intended every goal that at a certain time is in the operator’s field of view. The same problem can be mitigated by considering also other observations rather than only the gaze, as for example one hand trajectory. This approach was followed in [11], where a hidden Markov model (HMM) was adopted. The HMM allows to make inference about a sliding temporal window of observations.

3.1.1 Background

The position p of the operator’s hand was adopted in [135] to update with discrete sampling time the distribution of probability $[\mathbb{P}(G_1) \cdots \mathbb{P}(G_m)]$, explaining in a probabilistic way which is the next goal intended by the human with that hand (two distinct probability distributions can be updated simultaneously for the two hands). The update of $\mathbb{P}(G_{1,\dots,m})$ was done computing the angles $\delta^{1,\dots,m}$. δ_k^i is the angle between an estimation of the operator’s hand velocity v_k and a nominal one \tilde{v}_k^i . \tilde{v}_k^i is the velocity at step k of a minimum jerk trajectory that starts in p_{k-1} with a velocity equal to v_{k-1} and terminates into C_i , *i.e.* the center of G_i , refer to the left picture of Figure 3.1.

Notice that at every step, m different angles δ must be computed, considering the different positions $C_{1,\dots,m}$ of goals in \mathcal{A} . The more δ^i is low, at least null, the more it is evident that the human is intending G_i . Since only a single operator’s hand was considered for the inference process, the model proposed in [135] had a feature set $\Phi = \delta$. The probabilities $\mathbb{P}(G_{1,\dots,m})_k$ were updated considering a recursive Bayesian classifier which assumes $\mathbb{P}(G_{1,\dots,m})_{k-1}$ as priors, leading to:

$$\begin{aligned} \mathbb{P}(G_i)_k &\propto L(\delta^i_k|G_i)\mathbb{P}(G_i)_{k-1} \\ \mathbb{P}(G_i)_k &= \frac{L(\delta^i_k|G_i)\mathbb{P}(G_i)_{k-1}}{\sum_{j=1}^m L(\delta^j_k|G_j)\mathbb{P}(G_j)_{k-1}} \end{aligned} \quad (3.1)$$

L is a likelihood function that assumed the value of a Gaussian distribution, having a null mean and a variance Σ which was a tunable parameter, whose value was set according to heuristic considerations.

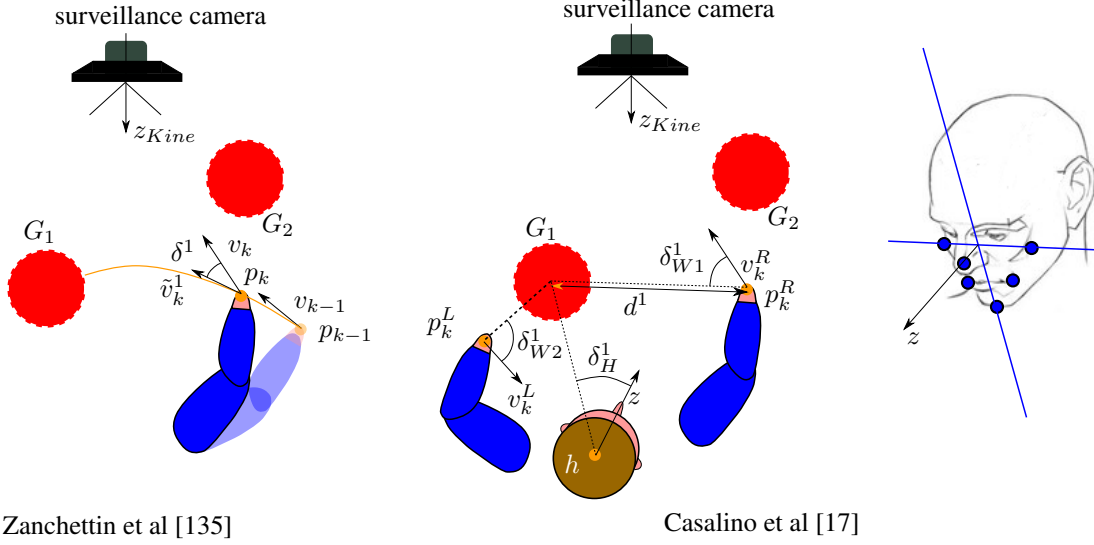


Figure 3.1: Sets of evidences Φ considered by the approach in [135] and [17]. In case of [17], the estimation of z is made according to some detected facial points, which are depicted as blue points.

The pipeline of the approach in [135] is reported in upper part of Figure 3.2, while the left part of Figure 3.1 describes the features involved for making inference.

3.1.2 Proposed approach

In [17] the probability distribution modelling the intended goal is computed in a recursive way, as done in [135], however considering a different set of evidences. For notational purposes, we'll distinguish the position p of the operator's hands, adopting p^R for the position of the right wrist, while p^L will be adopted for the left one. z will indicate the estimated orientation of the head, while h will denote the estimated position. Depth cameras like MICROSOFT KINECT, compute z as the normal of a plane which interpolates some detected facial points (Figure 3.1). Not only the estimate of z is returned, but a flag F about its validity is returned too. When the operator looks towards a direction that is very different from the normal of the camera frame z_{Kin} (Figure 3.1), the retrieved measure is non reliable.

The directions of the velocities of the wrists, v^R and v^L , are estimated as similarly done in the method described in the previous Section, *i.e.* according to the past acquired samples for p^R and p^L . The probabilities $\mathbb{P}(G_{1,\dots,m})$ are updated by making use of a recursive Bayesian classifier, however adopting a different likelihood function L as will be described in the following.

In principle, the probabilities $\mathbb{P}(G_{1,\dots,m})^R$ of the right arm intended goal could be updated considering only the evidences related to the right arm and similarly $\mathbb{P}(G_{1,\dots,m})^L$ with the left one. Actually, this was the approach followed in [135], equation (3.1). On the opposite, [17] proposed to compute L by considering as features both the hand positions for updating $\mathbb{P}(G_{1,\dots,m})^R$ and $\mathbb{P}(G_{1,\dots,m})^L$. The entire set of evidences Φ is composed as follows (see also Figure 3.1):

$$\Phi = (\delta_{W1} \quad \delta_{W2} \quad \delta_H \quad d)^T$$

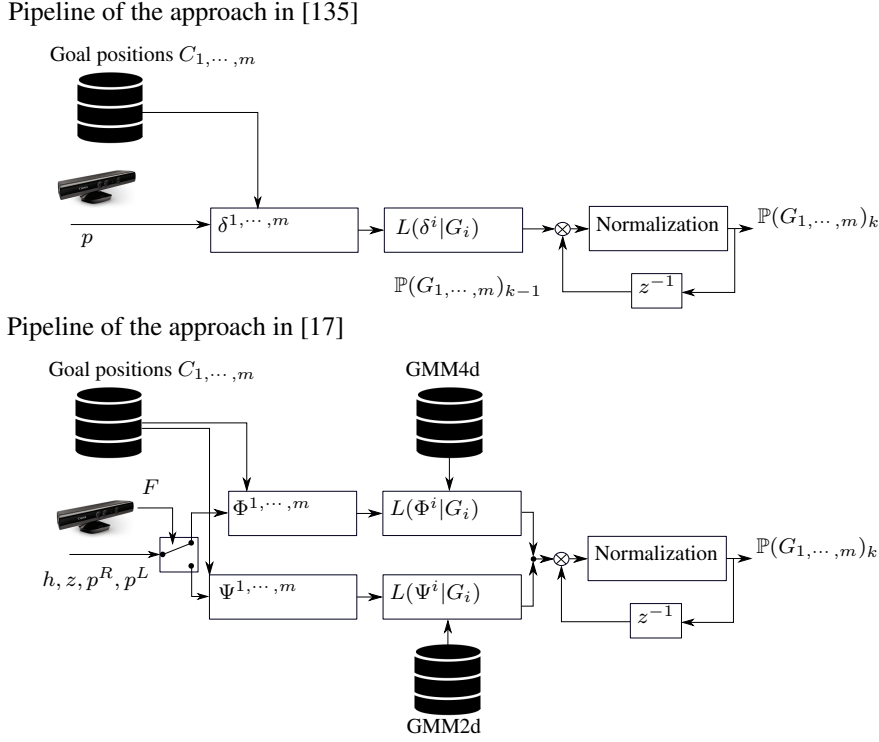


Figure 3.2: Pipelines of the inference approaches proposed in [135] and [17]. In [17], flag F is exploited to select the proper mixture model to use for computing the likelihood L .

When considering inference for the right hand, δ_{W1}^i is taken as the angle between v^R and the vector connecting p^R to the center C_i of G_i , while δ_{W2}^i is a similar angle but considering v^L . On the contrary, when inference is made for left hand, δ_{W1}^i is computed according to v^L and δ_{W2}^i according to v^R . The angle δ_H^i is the one between z and the vector connecting h to C_i , while d^i is the Euclidean distance between p^R and C_i , when considering inference for the right hand, while is the same involving p^L when considering inference for the left one. Angle δ_{W2} is included in Φ because a high evidence that the left hand is going to a certain goal, reduces the probability that this goal is intended for the right hand. Clearly we assume that the operator uses a single hand to reach a certain goal¹. On the contrary, not considering δ_{W2} in Φ would lead to update the probability distributions related to the hands in a completely independent way, which is not realistic. Also in this case, Φ must be computed at every step for every possible goal, leading to a population of $\Phi^{1,\dots,m}$. The updating law is the following one:

$$\begin{aligned} \mathbb{P}(G_i)_k^R &\propto L(\Phi_k^i | G_i) \mathbb{P}(G_i)_{k-1}^R \\ \mathbb{P}(G_i)_k^R &= \frac{L(\Phi_k^i | G_i) \mathbb{P}(G_i)_{k-1}^R}{\sum_{j=1}^m L(\Phi_k^j | G_j) \mathbb{P}(G_j)_{k-1}^R} \end{aligned} \quad (3.2)$$

A data driven approach was adopted for determining the likelihood function L . In fact, L was approximated with a GMM learnt from a training set. Prior to detail the steps involved for learning a model for L , some basic concepts about Gaussian Mixture Models will be provided.

¹But moving simultaneously both hands to reach different locations.

Gaussian Mixture models as approximating functions

GMMs can be adopted for approximating unknown probability density distributions as will be shown.

Mixture models Mixture models are in general a way to define a probability density function as the combination of a certain number of simpler ones. Let f be the probability density function of a continuous random variable x . Assuming X as domain for the possible realizations of x it holds what follows:

$$f : X \rightarrow [0, 1] \tag{3.3}$$

$$\int_X f(x)dx = 1 \tag{3.4}$$

It is possible to define a generic mixture model, by combining N probability densities f_1, \dots, f_N satisfying equation (3.4) and having the same domain X . Indeed, considering N weights $\lambda_1, \dots, \lambda_N$, the density of the mixture f_{mix} is defined as follows:

$$f_{mix}(x) = \sum_{i=1}^N \lambda_i f_i(x) \tag{3.5}$$

To ensure that f_{mix} is in turn a valid probability density function satisfying equation 3.4, it is necessary to impose that the combination expressed in equation (3.5) should be convex, meaning that:

$$\sum_{i=1}^N \lambda_i = 1 \tag{3.6}$$

In fact, when the above specification holds, it is true that:

$$\begin{aligned} \int_X f_{mix}(x)dx &= \int_X [\lambda_1 f_1(x) + \dots + \lambda_N f_N(x)]dx \\ &= \int_X \lambda_1 f_1(x)dx + \dots + \int_X \lambda_N f_N(x)dx \\ &= \lambda_1 \int_X f_1(x)dx + \dots + \lambda_N \int_X f_N(x)dx \\ \int_X f_{mix}(x)dx &= \lambda_1 + \dots + \lambda_N = \sum_{i=1}^N \lambda_i = 1 \end{aligned} \tag{3.7}$$

The simplifications made in the above equation are justified by the fact that every function f_i is a probability distribution function satisfying equation (3.4):

$$\begin{cases} \int_X f_1(x)dx = 1 \\ \vdots \\ \int_X f_N(x)dx = 1 \end{cases} \tag{3.8}$$

It is worth noticing that no particular hypothesis were posed about the combining distributions f_1, \dots, f_N . We need only to require they are valid probability density functions defined over the same domain. The above considerations are true also when considering multivariate distributions.

The values of the weights $\lambda_{1,\dots,N}$, as well the parameters characterizing every single f_i in the mixture, are tuned through a learning process which considers as training set $\langle x^1, \dots, x^M \rangle$, i.e. M realizations of f_{mix} .

The functions characterizing the mixture can be interpreted as clusters. In such cases, the values of weights $\lambda_{1,\dots,N}$ are priors for the probability that a sample was drawn from the corresponding f_i . The classification of a value \bar{x} , according to the Bayes formula, can be done as follows:

$$\begin{aligned} \mathbb{P}(\bar{x} \in Cluster_i) &= \mathbb{P}(Cluster_i|\bar{x}) \propto L(\bar{x}|Cluster_i)\mathbb{P}_{prior}(Cluster_i) \\ &\propto f_i(\bar{x})\lambda_i \\ &= \frac{f_i(\bar{x})\lambda_i}{\sum_{j=1}^N f_j(\bar{x})\lambda_j} \end{aligned} \quad (3.9)$$

Gaussian Mixture models Gaussian mixture models are particular mixture models combining N multivariate Gaussian distributions. From Section 3.1.2, it follows that the distribution of a GMM, f_{GMM} , is defined in this way:

$$\begin{aligned} f_{GMM}(x_{1,\dots,n}) &= \sum_{i=1}^N \lambda_i f_{Gauss\ i}(x_{1,\dots,n}) \\ f_{Gauss\ i}(x_{1,\dots,n}|\Sigma_i, \mu_i) &= \frac{1}{\sqrt{(2\pi)^n |\det(\Sigma_i)|}} \\ &\exp\left(-\frac{1}{2}(x - \mu_i)^T \Sigma_i^{-1} (x - \mu_i)\right) \end{aligned} \quad (3.10)$$

where in the above equation Σ_i and μ_i are the variance and mean, respectively, of the i^{th} Gaussian distribution in the mixture.

Learning of GMM is typically made using the Expectation Maximization (EM), whose steps are extensively described in Appendix B. EM, is essentially an iterative algorithm that starts from an initial guess for the parameters of the clusters $\theta = \{\dots, \lambda_i, \Sigma_i, \mu_i, \dots\}$, usually computed employing a k-means classifier, and then adjusts the model values until the convergence to a maximum for $L(\theta|X)$, see Appendix B.

EM is considered as an unsupervised algorithm, since only the number of clusters must be specified when performing learning, omitting the labels ² of the elements in the training set.

Approximating capabilities of GMM GMM can be also adopted for approximating complex unknown distributions. Indeed, samples in the training set do not have to be actually generated from a mixture of Gaussians. They can be generated by an unknown distribution, that will be approximated later by a set of Gaussians. Then, considering the proper number of clusters, any kind of probability distribution can be approximated by a GMM. To this purpose, it is possible to undertake many training sessions with the EM algorithm, varying every time the number of clusters of the model, obtaining a population of possible models. Then, the one maximising the likelihood of the training

²The Gaussian in the mixture that produced that sample

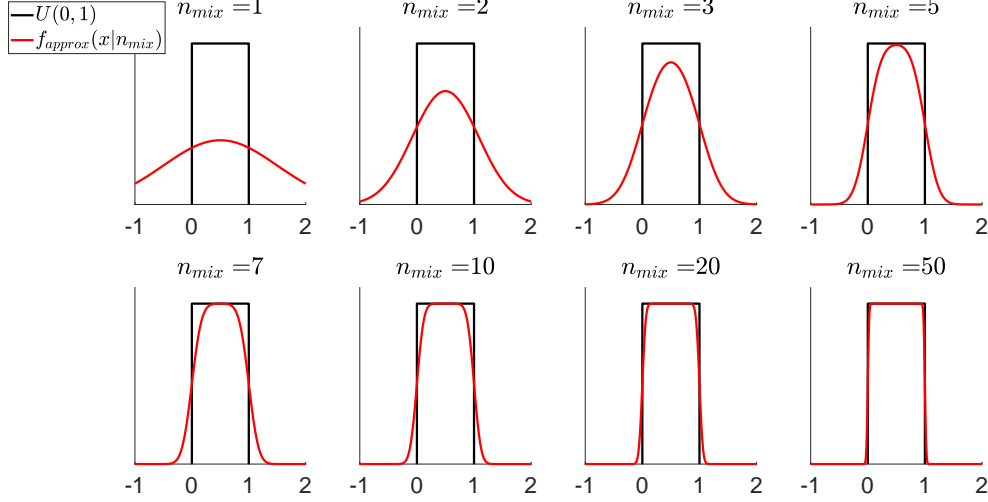


Figure 3.3: The uniform density between 0 and 1 is compared with an approximating mixture, varying the number of components.

set is assumed as the one approximating the unknown distribution.

As an example, consider to approximate the uniform distribution $U(0, 1)$ with a GMM, f_{approx} , made of n_{mix} components defined as follows:

$$f_{approx}(x|n_{mix}) = \sum_{i=1}^{n_{mix}} f_{Gauss\ i}(x|\Sigma_i, \mu_i)$$

$$\Sigma_i = \left(\frac{1}{n_{mix}}\right)^2 \quad \mu_i = \frac{1}{n_{mix}}(i - 0.5) \quad (3.11)$$

In Figure 3.3, the density of $U(0, 1)$ is compared with the one of f_{approx} , varying the number of approximating clusters. As can be visually appreciated, the approximation error decreases when increasing the number of clusters in the model. This phenomenon is not always verified when considering other kind of distributions.

Learning the likelihood function describing the human intention

We are now in the position to detail the procedure adopted for learning the likelihood function $L(\Phi)$ (see equation (3.2)). As already discussed, $L(\Phi)$ is approximated by an $L_{GMM}(\Phi)$.

L_{GMM} can be learnt from data in a supervised manner. Indeed, we can monitor some operators during the execution of predefined sequence of tasks, collecting some trajectories $p^R(t), p^L(t), h(t), z(t)$. Then, knowing the intended goal locations³, it is possible to extract from the collected data a population of samples $\langle \Phi_{1, \dots, M} \rangle$, which can be used to fit L_{GMM} by making use of EM algorithm. Learning is done varying the number of clusters of the mixture in order to take the model maximising the likelihood of the parameters w.r.t. to the training set, see Appendix B.

It is worthy to point out that the learnt model is general, in the sense that it can be

³In this sense the approach is supervised.

applied to any kind of robotic cell (with its own locations C for the goals G). In this way, when changing the layout of a cell, we can still rely on the learnt GMM, without a new training of the model. On the other hand, a specific GMM learnt for a newer layout could achieve higher performance.

Managing non accurate sensor information

Once the GMM describing the likelihood function has been learnt, it is possible to exploit it on-line, to make inference about the intended goal. For the sake of simplicity, suppose that only the distribution related to one hand of the operator is updated. To this purpose, the measures retrieved from the sensor as well as the knowledge about every possible goal location are exploited to compute $\Phi_{k+1}^1, \dots, \Phi_{k+1}^m$, *i.e.* the evidences. Then, equation (3.2) is adopted to compute the a posteriori probabilities of every goal at step $k + 1$.

Since the information about z is not always reliable, a switching model must be taken into account. In fact, when z is not available, we can make inference according to a reduced vector of features $\Psi_k^i = (\delta_{W1k}^i \ d_k^i)^T$. Another GMM distribution, let us call it GMM2d, can be learnt from samples of Ψ , in a similar way as for the one describing Φ , that can be denoted as GMM4d. Then, depending on the value returned at step k for flag F , the update of probabilities for every goal is made according to GMM4d, or according to GMM2d.

The overall pipeline of the presented approach is reported in the lower part of Figure 3.2.

Managing uncertainties for the location of the goals

As stated in the previous Section, the on line computation of the feature vector Φ^i (or Ψ^i) requires to know the goal centres $C_{1, \dots, m}$. However, in realistic contexts, this quantity is not a precise value, but it is rather a random variable with a certain distribution of probability. To manage this uncertainty it is only required to characterize the distribution which describes the goal location, possibly assuming an additional approximating GMM. In this way, some possible locations for goals can be sampled every time the update of probability is required.

As done with particle filter algorithms, when computing the likelihood of the i^{th} uncertain goal we can consider a set of Np samples c^{i1}, \dots, c^{iNp} as hypothesis about the true goal location. Then, for every sample c^{ij} , it is possible to compute the likelihood function $L(\Phi^{ij} | c^{ij})$, which implies to compute different Φ^{ij} . The global likelihood function adopted for updating the probabilities can be computed as a mean of the likelihood of every sample⁴:

$$L(\Phi_{k+1} | G_i) = \frac{1}{Np} \sum_{p=1}^{Np} L(\Phi_{k+1}^{ij} | c^{ij}) = \frac{1}{Np} \sum_{p=1}^{Np} L_{GMM}(\Phi_{k+1}^{ij}) \quad (3.12)$$

3.1.3 Operator awareness through intention recognition

The method described in the previous Section was tested in a collaborative task, where a human operator has to assemble a box with the help of a robotic manipulator. The

⁴Which can translated in a weighted sum, in case samples are not equally probable.

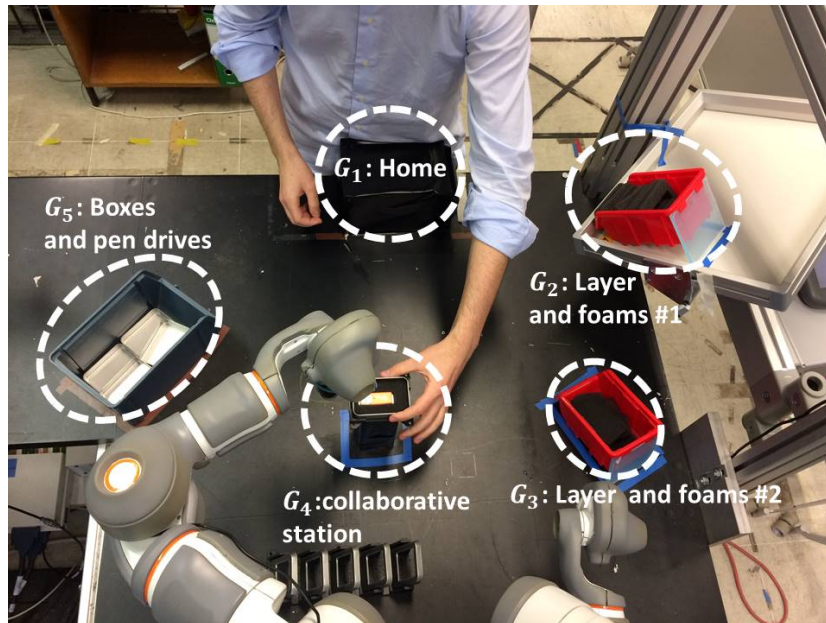


Figure 3.4: Layout of the experimental robotic cell. Locations of possible human goals are indicated with white dotted circles.



Figure 3.5: Vibrotactile ring with its controller box. During the experiments the ring is worn on the operator's left hand and the box is attached to a Velcro bracelet worn on the forearm.

experimental setup is reported in Figure 3.4. The operator's left hand is equipped with a vibrotactile ring (Figure 3.5), which contains a 4 mm vibration motor (PRECISION MICRODRIVES™)⁵, that is controlled through an Arduino Pro Mini⁶. The communication with the ring is wireless, thanks to two XBee® RF modules (Digi International Inc.)⁷. The ring itself weighs around 2g, whereas the complete device (ring plus controller box placed on the wrist) weights around 40g.

The depth camera, the vibrotactile ring and the robot are connected to a CPU, where the inference algorithm is implemented. The CPU reads the measurements retrieved from the MICROSOFT KINECT and sends commands to both the robot and the ring.

⁵ <http://www.precisionmicrodrives.com/product/304-101-4mm-vibration-motor-11mm-type>

⁶ <http://store.arduino.cc/arduino-pro-mini>

⁷ <http://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Datasheet.pdf>

Description of the collaborative task

The objective of the task is to assemble a box containing a USB pen drive. The task has 5 steps:

1. The human fills the box with two layers of foam, and the USB pen drive.
2. The human brings the filled box towards the robot.
3. The robot adds the cover on the box.
4. The human fixes the cover with some tape.
5. The robot stores the finished product.

The operator is monitored during the experiment by the depth camera, whose observations are given to the inference algorithm presented before. The set of possible human goals $G_{1,2,\dots}$ is composed as follows (see Figure 3.4):

1. Home position
2. Feeder of the first kind of layer foams
3. Feeder of the second kind of layer foams
4. Collaborative station
5. Feeder of the boxes and pen drives

The haptic feedback is used twice. The first time to inform the human that the robot understood his/her intention of putting the box in the shared workspace, *i.e.*, the collaborative station (beginning of step 2). The second time to inform the human that the robot understood that step 4 has been completed. The decision about the proper time to send one new feedback is made according to the evolution of probabilities about the human's goals, as well as according to the state machine depicted in Fig. 3.6. When at least one new human is detected in the scene, the state machine goes out of its initial state and reaches state 1. This state persists until the probability related to goal 4 grows above a predefined high threshold (0.8 for instance), meaning that the human has completed the first step and is about to deliver the partially assembled box to the collaborative station. When this happens, the machine enters state 2 and returns to state 1 only when the probability of goal 1 grows above another threshold, implying that the operator has finished step 4 and is about to begin a new cycle. Every time the machine goes from state 1 to state 2 or vice versa, a vibrotactile feedback (a vibration burst lasting 120 ms, with a frequency of 200 Hz and an amplitude of 0.8 g) is sent to the operator. The same kind of feedback was adopted in both cases because the aim of the delivered message is the same, *i.e.*, informing in a reactive way the human that the robot has understood his/her intention and is about to move.

Experiments

16 participants were recruited for the experiments. Half of them performed the collaborative task with the haptic ring, and half without. In both groups, 5 out of 8 subjects were considered **non-skilled**, as they declared to be not familiar with the use of robots.

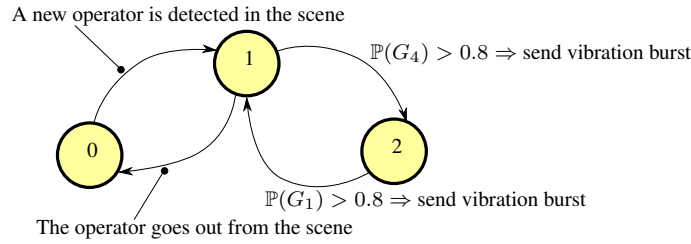


Figure 3.6: State machine adopted to send feedback to the operator. Note that vibration bursts are sent before the operator actually reaches the goals, according to the probabilities evolution.

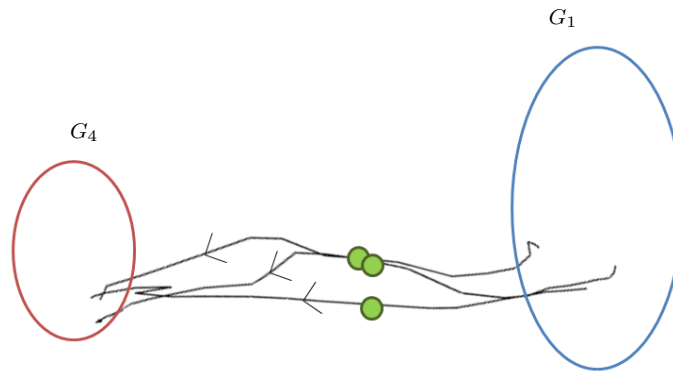


Figure 3.7: Some trajectories taken from the experiments, going from goal 1 (blue) to goal 4 (red). Green markers are located at the points of the path for which the subject receives the haptic feedback.

The **skilled** participants, instead, had previous experience with human-robot collaborative tasks (*e.g.*, they already took part in HRC experiments), but were not specifically trained for the proposed assembly task. All subjects were asked to perform the collaborative task for 5 consecutive times, and those wearing the ring were instructed on the meaning of the vibration burst. During the experiments the ring was worn on the operator’s left hand, that was the one tracked with the MICROSOFT KINECT. The execution time of each trial was recorded and subjects who used the vibrotactile interface were asked to evaluate their experience with a questionnaire. Fig. 3.7 shows some trajectories taken from one subject, with the point at which the haptic feedback is sent, which is almost in the middle of the path.

The method described in Section 3.1.2 was adopted to train the GMM4d and GMM2d models. Results regarding training of GMM4d are reported in Figure 3.8, where the likelihood of the model is plotted against the number of clusters considered. A number of 7 clusters was considered for GMM4d since, as can be seen from Figure 3.8, considering a greater number does not improve significantly the likelihood of the model.

For the experiments, three different aspects were analyzed: the benefits of using gaze estimation, the benefits of using the vibrotactile feedback, and the overall subjects’ evaluation of the haptic ring.

Benefits of using gaze estimation

To highlight the benefits of using the estimate of the gaze in the inference process, the probabilities evolution obtained with the following different methods were compared:

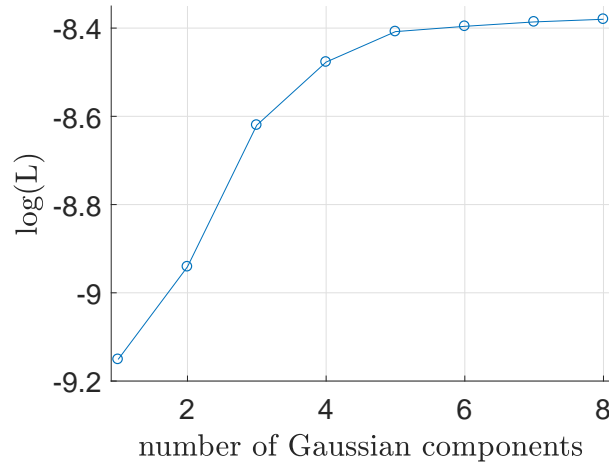


Figure 3.8: The figure reports the likelihood of model GMM4d, varying the number of clusters considered when training the model. The selected number of clusters was 7.

M1: the method proposed in [135];

M2: an approach that updates the probabilities based on the trajectory of a single hand and using the likelihood function denoted as GMM2d, simulating that the information about gaze is always invalid;

M3: the complete approach proposed in Section 3.1.2, where inference is made by considering both the detected positions of the operator’s hands and his/her gaze.

M3 was applied on-line, while the other two were applied off-line on the same measurements retrieved from the depth camera. Figure 3.9 reports the distribution of the distance at which goal 4 was correctly recognized by using the aforementioned methods. As can be seen, the performance of the methods is quite similar. Anyway, the robustness of the three methods must be taken into account. The number of false positives is defined as the number of times for which the probability of a certain goal has risen beyond the threshold of 0.8, but the operator was going to a different target. The number of true negatives, instead, corresponds to the number of times for which the operator was going to a certain goal, but the probability of the same did not rise above the threshold. From the analysis of Table 3.1, that reports data related to goals 1 and 4, we can state that the information about the gaze allows to acquire an improved robustness in the inference process. The position of goal 1 is much more scattered than goal 4, and this reflects on the performance of our algorithm, even though it remains better than the method proposed in [135].

3.1. Estimating the current intended goal

Table 3.1: Percentages of false positives and true negatives are with respect to the total number of times the operators went to goal 4 for the upper table. The lower one refers to goal 1.

Goal 4	M1	M2	M3*
% False Positives	4.54	11.53	0
% True Negatives	16.67	0	0
Goal 1	M1	M2	M3
% False Positives	9.09	11.15	6.06
% True Negatives	21.21	12.3	8.3

*The third column of the first table contains zeros because during the experiments neither false positives, nor true negatives were noticed (N.B. method 3 was applied on-line).

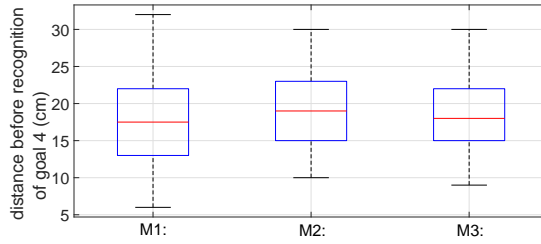


Figure 3.9: Comparison of the three different inference methods regarding the goal recognition performance. Distance before recognition refers to the length of the path described by the wrist of the operator, from the instant at which the recognition of goal happens until the time at which the operator reaches goal 4.

Benefits of using haptic feedback

To evaluate the advantages of using haptic feedback, the task execution time was considered. The intuition behind these kind of experiments is that if the human is aware that the robot has understood his/her intention, he/she does not have to wait to see it moving, and can proceed with the task execution in a more fluent and fast way. In other words, the haptic feedback makes the user confident that his/her artificial mate is working as expected.

According to the data gathered in the experiments described, the variability in cycle time is reduced with the help of vibrotactile feedback, however no statistical evidence of this fact can be proven (Figure 3.10(a)). Within the population of non-skilled participants (Figure 3.10), instead, it is possible to find a statistical evidence of the fact that the average cycle time is reduced with the help of vibrotactile feedback (single-tailed Wilcoxon rank sum test with confidence $\alpha = 0.05$ returns $r = 0.9907$). A tangible decrease of the cycle time variability for the same population can also be appreciated, however still without statistical evidence.

The time elapsed from the instant the human finishes to fix the tape on the cover and the time he/she reaches goal 1 to begin a new assembling cycle was considered too, for both skilled and non-skilled participants (Fig. 3.11). The average time when receiving the vibrotactile feedback is strongly proved to be statistically lower (single-tailed Wilcoxon rank sum test with confidence $\alpha = 0.05$ returns $r = 0.9998$). This confirms the hypothesis that the usage of vibrotactile feedback for some crucial parts of the interaction makes the user more confident about the robot behaviour, improving his/her productivity.

The task here proposed is rather easy, and that is probably why performance of peo-

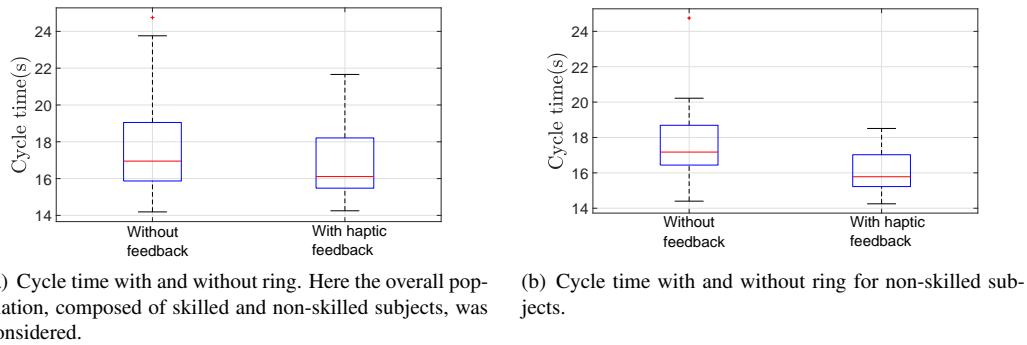


Figure 3.10

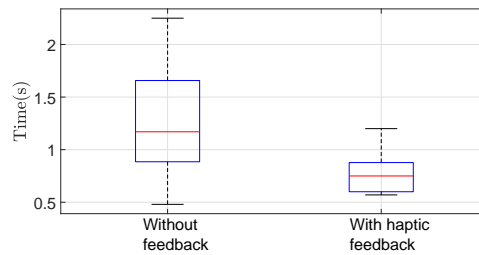
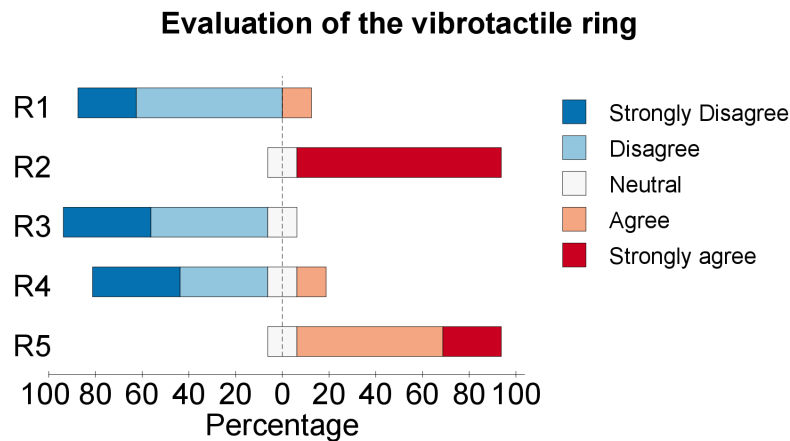


Figure 3.11: Time elapsed from the instant when the tape is fixed until the one when the operator returns to goal 1. The overall population, composed of skilled and non-skilled subjects, was considered.

ple with previous experience in robotics does not seem to benefit from haptic feedback. However, having seen the results obtained with non-skilled subjects, we can expect that in contexts where the production lines change frequently (*e.g.*, agile manufacturing), haptic feedback could prove to be significantly beneficial.

To further assess its usability in industrial contexts, haptic feedback could be compared with auditory or visual cues, in case hearing and sight are not impaired during the task execution. However, the comparison between different types of feedback modalities was out of the scope of this work. Future developments could address this issue and also study how multimodal interaction [46] can be combined with intention prediction algorithms to perform HRC tasks.



R1: I found the vibrotactile ring very cumbersome to use.

R2: The vibrations produced by the ring are easy to distinguish.

R3: Wearing the ring impedes some movements.

R4: The vibration is annoying.

R5: Having an acknowledgement from the robot helps in going on with the task.

Figure 3.12: Subjects' answers in percentage to the question *Express how much you agree with the following statements concerning the vibrotactile ring.*

Overall evaluation of the haptic ring

After the experiments done with the haptic ring, a questionnaire was submitted to every subject to get their subjective evaluation of the device. None of the 8 subjects had previously used wearable haptic interfaces before. The questions were formulated as five-level Likert items, and from the answers shown in Figure 3.12 we can derive an overall appreciation of the ring, that was felt more as a help (cf. R5) than as an encumbrance (cf. R1, R3). Answers to question R5 are in line with the results shown in Figure 3.11, underlining that the vibrotactile feedback helps in proceeding smoothly with the task.

Segmenting the human actions by analysing the upper body motion

It is not always possible to follow the approach described in Chapter 3 for detecting the inception or the termination of a human activity. Indeed, it is not ensured that a one-to-one correspondence exists between actions and goals. For example, multiple different tools can be stored in the same buffer or distinct goals may be too close for recognizing that the human is entering into one of them. Moreover, there could be actions not related to goals as for instance those performed by the human after taking all the necessary tools.

In these circumstances, the motion of the entire body of an operator must be analysed and interpreted with more sophisticated approaches, as the one proposed in this Section. The set of possible human actions \mathcal{A} (see the initial Section of Part I) is still assumed to be finite and known.

4.1 Evolving factor graphs for segmenting the human actions

The aim of the algorithm proposed in this Section is to detect the starting and the ending time of human actions, by analysing the motion performed by the operator in the recent past. To this purpose, RGB-D sensors can be exploited to keep track of some points of interest in the human silhouette (see also Chapter 9). The signals retrieved are subdivided into many sub-windows each having a maximal length of l_w samples. Then, a feature vector $F_O^i = [O_1^i \ \dots \ O_F^i]$ made of F components can be extracted from the i^{th} window, representing an indirect indication of the action $a \in \mathcal{A}$ that was performed within the same window. Every F_O^i is computed by taking into account the mean and variance value of some inter-skeletal distances, refer to Figure 4.1. Additionally, the distance of the wrists w.r.t. the centres of goals $C_{1,\dots,m}$, representing the position of

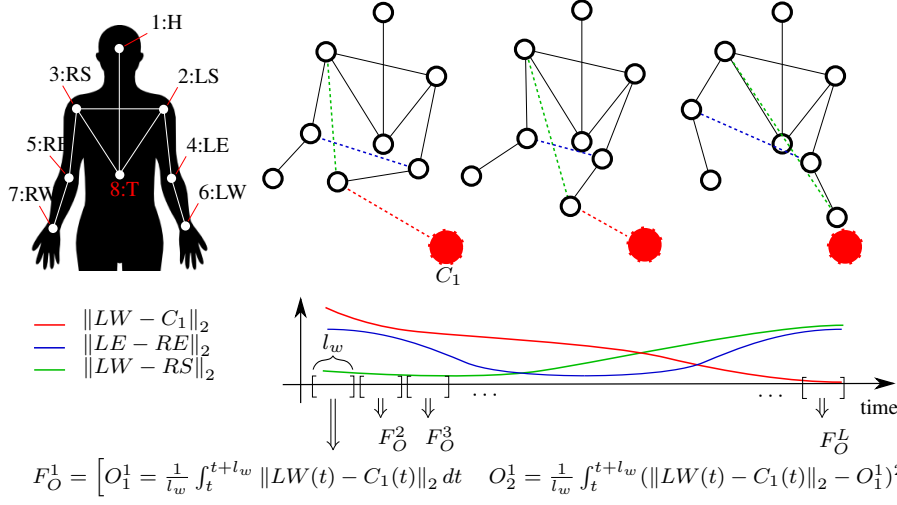


Figure 4.1: The approach followed to obtain the observations. The trajectory of the human is split into many windows of length equal to l_w . The mean and variance of some skeletal distance values are computed for obtaining the set of observations F_O^1, \dots, F_O^L .

buffers storing the parts to use, can be taken into account.

The hypothesis is that a specific action a will produce characteristic values for F_O . However, the main problem to overcome is that the sequence of actions performed by the human, their durations and their number is not known. Therefore, when dealing with a macro-window made of L segments of length equal to l_w , we have to find a way to determine the most probable sequence of actions that produced that observations.

The problem can be tackled considering a variable-structured Markovian Random Field (see Appendix C), having as hidden variables the sequence of actions actually performed by the human and as evidences the features $F_O^{1, \dots, L}$ contained in the window for which the segmentation of actions has to be computed. Vector $\rho = [\rho_1 \ \dots \ \rho_S]$ is adopted for describing the durations of the human actions as well as their number. Indeed, ρ_i indicates the percentage of time spent by the human when doing the i^{th} action in the sequence. Knowing ρ , it is possible to build the underlying factor graph, assuming to connect $\rho_i \cdot L$ observations to the node representing the i^{th} action¹, refer to Figure 4.2. The potentials involved in the structure reported in Figure 4.2 will be discussed in Section 4.1.1 and 4.1.2. Clearly, it must be ensured that $\sum \rho_i = 1$.

Since the real segmentation ρ^* describing the sequence of human actions is not available, the proposed approach considers many hypotheses ρ^1, ρ^2, \dots , which are iteratively compared with the aim of finding the best one, i.e. the one more in accordance with the observations retrieved from the sensors. More specifically, a genetic algorithm will be exploited as reported in Section 4.1.3.

4.1.1 Modelling the correlation existing between the observations and the human actions

The generic potential $\Psi_{O_j Y}$ (Figure 4.2) describes a correlation existing between O_j and Y , i.e. a variable representing a performed action. It is an exponential potential

¹Values are rounded to obtain integer quantities.

4.1. Evolving factor graphs for segmenting the human actions

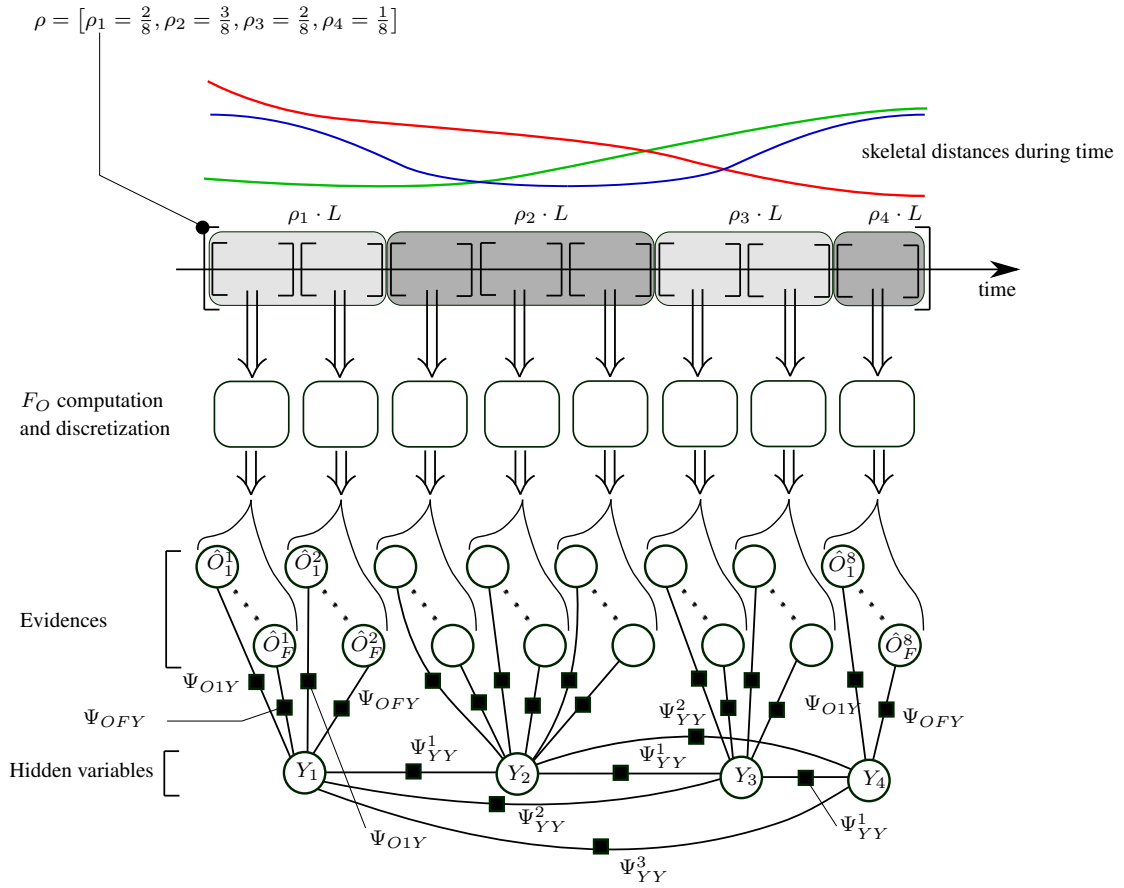


Figure 4.2: Example of graph construction. The observations are partitioned to actions, as described by the values contained in vector ρ . The observations $\hat{O}_1, \dots, \hat{O}_F$ are computed as indicated in Section 4.1.1.

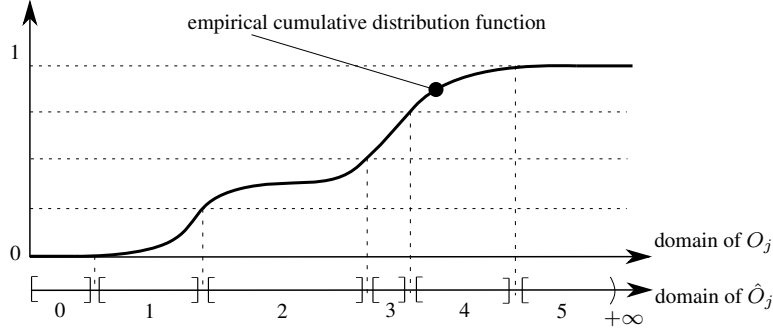


Figure 4.3: The intervals considered for computing \hat{O}_j from O_j , are obtained by considering equispaced portion of the image of the empirical cumulative distribution function describing the values that O_j can assume.

(see Appendix C) defined in this way:

$$\Psi_{O_j Y}(\hat{O}_j, Y) = \exp\left(w_{O_j} \cdot \Phi_{O_j Y}(\hat{O}_j, Y)\right) \quad (4.1)$$

where \hat{O}_j is a discrete projection of O_j , while w_{O_j} is a weighting coefficient whose value becomes known after performing training, as indicated in Section 4.1.5. \hat{O}_j is a categorical variable whose relationship with O_j is obtained by considering some statistics about O_j , retrieved from a training set made of recorded human actions. Indeed, each trajectory in the set is subdivided into many equispaced portions having a length equal to l_w and the value of O_j in each sub-portion is computed. In this way, a population of values for O_j is collected and an empirical cumulative distribution function can be built. The latter distribution is considered for defining some non-equispaced partitions of the domain of O_j , refer to Figure 4.3. \hat{O}_j assumes the value associate to the interval containing the real value of O_j . The value of $\Phi_{O_j Y}$ in a particular combination \hat{O}_j, Y is assumed to be equal to the times for which that pair is encountered in the training set.

4.1.2 Modelling the sequence of actions

The potentials $\Psi_{YY}^1, \dots, \Psi_{YY}^o$ (Figure 4.2) correlating the actions performed during time are assumed to be defined in this way:

$$\Psi_{YY}^i(Y_k, Y_{k-i}) = \exp\left(w_Y \cdot \Phi_{YY}^i(Y_k, Y_{k-i})\right) = \exp\left(w_Y \cdot \lambda_i \cdot Q_i(Y_k, Y_{k-i})\right) \quad (4.2)$$

where in the above equation λ_i and Q_i are the parameters of an Higher order Markovian model (Section 5.1) modelling the sequence of actions. $Q_i(Y_k, Y_{k-i})$ refers to the value in row Y_k and column Y_{k-i} ² of the i^{th} matrix in the model. The weight w_Y is shared by all the potentials connecting the hidden variables. The proportion between w_Y and w_{O_1}, w_{O_2}, \dots modulates the importance given to Markovian model w.r.t. the one assigned to the information coming from the observations.

²Notice that the variables involved in the graphs are all categorical

4.1.3 Segmenting the actions on a fixed window of observations

Suppose for the moment to perform the segmentation of actions, after having acquired all the observations F_O^1, \dots, F_O^L in a temporal window. A factor graph (Appendix C) can be built for each segmentation hypothesis ρ , as indicated before. Then, alternative hypothesis must be compared. We might think to perform the comparison by considering the following likelihood function (Appendix A):

$$L(\rho|F_O^1, \dots, F_O^L) = \mathbb{P}(F_O^1, \dots, F_O^L|\rho)\mathbb{P}(\rho)_{prior} \quad (4.3)$$

The computation of $\mathbb{P}(F_O^1, \dots, F_O^L|\rho)$ could be done by considering the underlying factor graph (the notation introduced at the beginning of Appendix C is adopted):

$$\begin{aligned} \mathbb{P}(\hat{O}_1^1, \dots, \hat{O}_F^1, \dots, \hat{O}_F^L|\rho) &= \sum_{\tilde{Y}} \mathbb{P}(\hat{O}_1^1, \dots, \hat{O}_F^1, \dots, \hat{O}_F^L, \tilde{Y}_{1,2,\dots}|\rho) \\ &= \sum_{\tilde{Y}} \frac{E(\hat{O}_1^1, \dots, \hat{O}_F^L, \tilde{Y}_{1,2,\dots})}{Z} \\ &= \sum_{\tilde{Y}} \frac{E(\hat{O}_1^1, \dots, \hat{O}_F^L, \tilde{Y}_{1,2,\dots})}{\sum_{\tilde{O}_1^1, \dots, \tilde{O}_F^L} E(\tilde{O}_1^1, \dots, \tilde{O}_F^L, \tilde{Y}_{1,2,\dots})} \end{aligned} \quad (4.4)$$

However, the above derivations are computationally intractable even for small sized graphs. For this reason, an approximation of the above quantity must be used. Such approximation will be treated as a fitness parameter by the genetic algorithm in charge of finding the optimal segmentation and it is defined as follows:

$$\varrho(\rho) = \frac{E(\hat{O}_1^1, \dots, \hat{O}_F^L, Y_{MAP}|\rho)}{\sup\{E(\rho)\}} \mathbb{P}(\rho)_{prior} \quad (4.5)$$

where Y_{MAP} is the maximum a posteriori estimation (Appendix C.0.1) of variables Y when assuming $\hat{O}_1^1, \dots, \hat{O}_F^L$ as evidences, while $\sup\{E(\rho)\}$ is the maximum possible value assumed by the energy function E , which is the product of the maximal possible value of the images of every potentials involved in the graph.

Priors $\mathbb{P}(\rho)_{prior}$ are computed considering $|\rho|$, i.e. the length of vector ρ . Indeed, the following Poisson distribution is assumed:

$$\mathbb{P}(\rho) = \frac{\exp(-\lambda)\lambda^{|\rho|}}{|\rho|!} \quad (4.6)$$

λ is tuned in order to have for $|\rho| > L$ a value equal to zero.

Get the optimal segmentation adopting a genetic algorithm

The fitness function $\varrho(\rho)$ is adopted by a genetic algorithm [40] for determining the optimal segmentation. This kind of algorithms progressively evolve a population of solutions, also called generations, till a certain number of maximal iterations. The generation at step k is a made of a set of solutions $\Omega^k = \{\rho^{k,1}, \dots, \rho^{k,G}\}$, representing segmenting hypothesis. The starting generation Ω^1 is obtained by randomly sampling³ some initial individuals. Then, the following steps are iteratively followed:

³Some random values for ρ are taken and then normalized. The number of values to consider is sampled too.

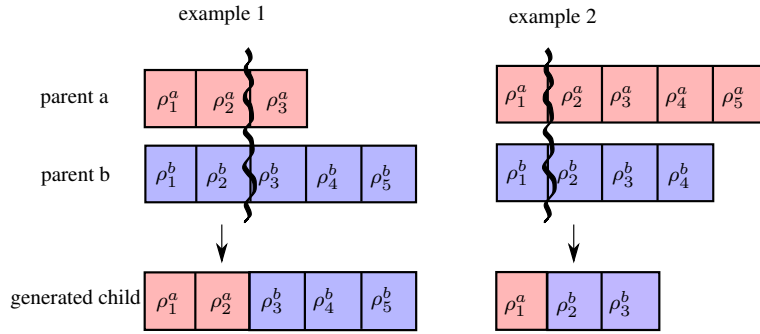


Figure 4.4: Examples of individual generation. At a random point, the solutions in the parents are broken and mixed to obtain the child.

1. The fitness values $\{\varrho(\rho^{k,1}), \dots, \varrho(\rho^{k,G})\}$ of individuals in Ω^k are computed.
2. G new individuals are determined for obtaining a new generation Ω^{k+1} . Every element in Ω^{k+1} is generated by combining two selected parents in Ω^k . The selection process can be done in many ways: tournament selection [84], roulette selection [72] and others. All the approaches have in common that the probability to be selected as a generating parent for an element in Ω^k is proportional to its fitness value. Once a pair of parents $\langle \rho^a, \rho^b \rangle$ is taken, a new solution ρ^{ab} is obtained by combining the values of the parents, refer to Figure 4.4.
3. With a certain frequency, the elements in the new generation are randomly muted. In the algorithm proposed here, the mutation can consist in: a variation in the values of a solution ρ ; an insertion of a new value ⁴.

The element in the final generation having the maximum ϱ is assumed as the best solution.

4.1.4 Segmenting the actions on evolving windows

The approach described so far is able to provide the most likely segmentation on a window of already acquired observations. However, in real cases, it is impossible to wait for the human to terminate all the assigned tasks and then execute the algorithm described before. Indeed, it might be desirable to determine partial results, considering the available observations.

The algorithm described in the previous Section can be invoked every time L new observations F_O^1, \dots, F_O^L become available. After the optimal segmentation is found for the first batch of observations, an initial graph is determined. Then, only a portion of this graph is kept for performing the segmentation on the second batch of observations. When assuming that the Markovian model used for computing $\Psi_Y^{1, \dots, o} Y$ is of order o , the last o hidden variables in the chain are kept for the subsequent computations. Then, the message passing algorithm (Appendix C.0.1) is exploited for determining the messages incoming into the surviving variables. Such messages are computed and stored as additional simple potentials (here the terminology introduced in Appendix C was used). The factors not contained in the sub-portion of the network that encompasses

⁴After the mutation, a solution is always re-normalized.

4.1. Evolving factor graphs for segmenting the human actions

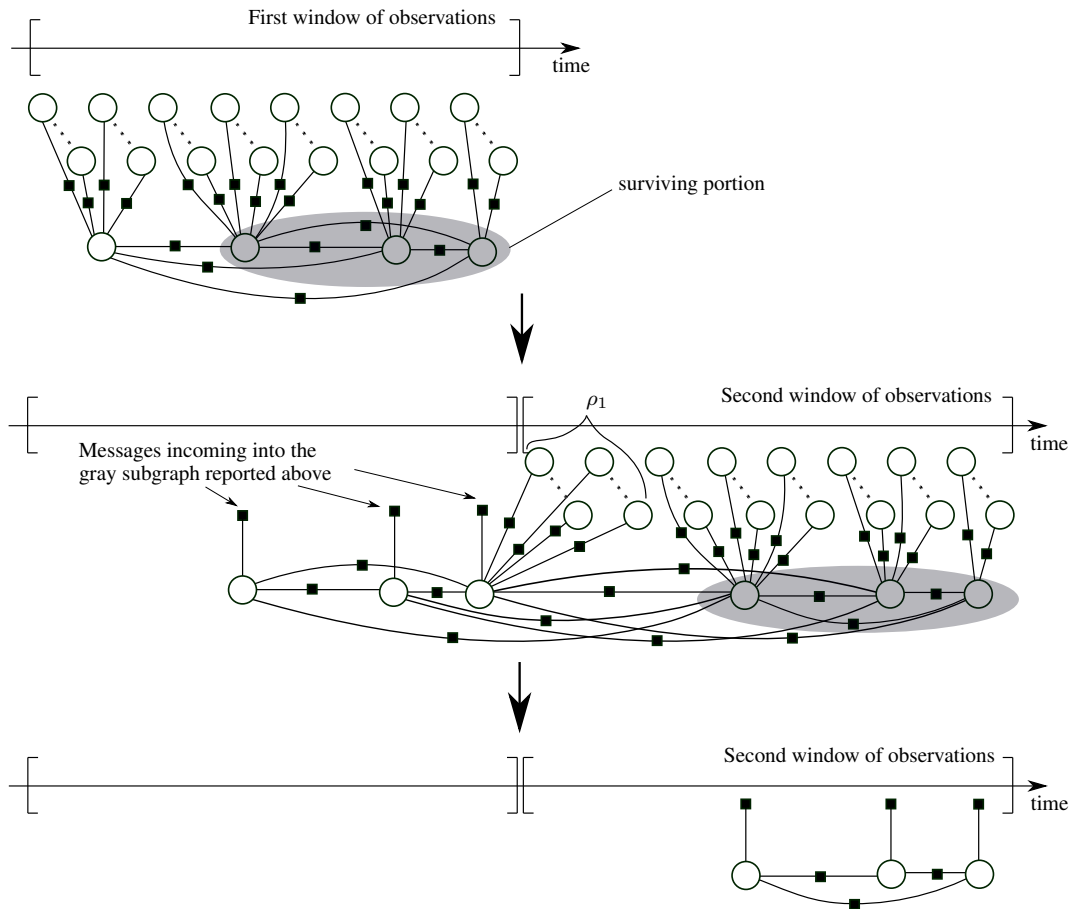


Figure 4.5: Mechanism adopted for performing a sliding window segmentation. The first segmentation is computed as described in Section 4.1.3. Then, the surviving variables are designated (the gray area in the picture at the top) and the incoming messages are computed. The surviving graph is kept constant and taken into account for performing the segmentation on the second window of observations (picture in the middle). When the optimal segmentation for the second step is determined, the new surviving variables are identified and the procedure is iterated.

the remaining variables are deleted.

The segmentation on the second batch is done with the same genetic algorithm previously introduced, with the only difference that the underlying factor graphs are computed in a different way. Indeed, the surviving portions of graph obtained at the previous step, are kept constant and are taken into account for connecting the new hidden variables. The first value of ρ indicates the portion of observations pertaining to the last hidden variable in the surviving chain. Figure 4.5 summarizes the above considerations.

After the optimal segmentation along the second window of observations is computed, the novel surviving portion of graph is designated, the messages incoming to the variables involved in such portion are computed and the procedure is iterated for the subsequent steps.

4.1.5 Tuning of the parameters

Weights w_{O1}, \dots, w_{OF} and w_Y must be learnt from a library of recorded human motions. We can assume to collect in a preliminary stage a certain number of samples for every action $a \in \mathcal{A}$. Each sample is then subdivided into many equally spaced temporal windows with a maximal possible length equal to l_w , with the aim of computing the corresponding feature vectors F_O .

The knowledge of the precedence constraints existing among the human actions can be exploited for producing a population of artificial sequences, as similarly done in Section 5.2.2. Such data can be exploited for fitting the Higher Order Markov model exploited for determining the potentials $\Phi_{YY}^{1,2,\dots}$ (Section 4.1.2).

The set of artificial sequences becomes the training set to use for training a graph containing weights w_{O1}, \dots, w_{OF} and w_Y . The values of the observations in the training set of the graph are sampled from the library of human motion described before. Since w_{O1}, \dots, w_{OF} and w_Y are shared by multiple potentials, the approach describe in Appendix C.0.3 is followed for computing the gradients of the likelihood function. Figure 4.6 summarizes the training procedure.

4.2 Experiments

The segmentation algorithm described so far has been adopted to recognize the actions performed by an operator when assembling the same torch described in Section 7.5.3. The operator is monitored while performing all the phases leading to the realization of a finite product, which comprises:

- a_1 : Perform the screwing of the first cap into the body of the torch
- a_2 : Insert the batteries into the battery case
- a_3 : Insert the light into the second cap of the torch
- a_4 : Insert the battery group and a spring into the body of the torch
- a_5 : Finalize the assembly by performing the screwing of the second cap

The steps described above are reported into Figure 4.9.

The assembling operations must be done with an order consistent with the precedence constraints reported at the bottom of Figure 4.9⁵. The experimental set-up reported in Figure 4.7 was exploited. The assembling operations are performed as approximately indicated in Figure 4.9. The RGB-d camera reported in Figure 4.7 is used to retrieve the human posture with a sampling time equal to 40 ms. Such data are saved into textual files for off-line processing. The positions of the buffers reported in Figure 4.7 are exploited for computing additional skeletal distances to consider for computing F_O (refer to Figure 4.1).

4 volunteers were enrolled for the experiments. They were asked to assemble 5 torches each, by following the order they prefer for the assembling steps. The acquired motion of the operators was off-line processed, assuming a l_w equal to 400 ms, while a new segmentation was computation every time 30 new observations become available, i.e. $L = 30$.

⁵The Markovian model describing the sequence of actions was computed off-line considering such constraints, in the way discussed in Section 4.1.5.

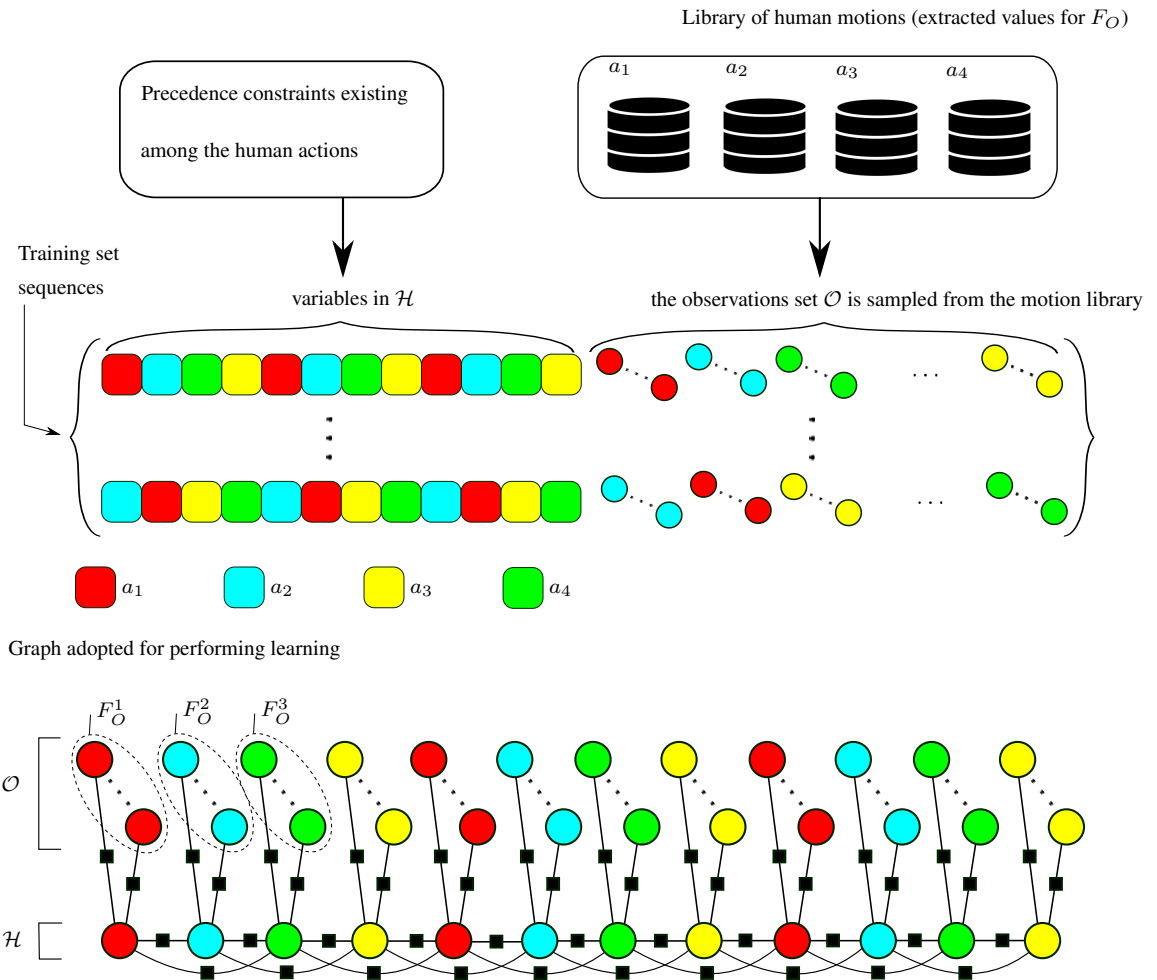


Figure 4.6: The procedure adopted for tuning the model parameters. The sequence in the training set (values for the hidden variables Y) are sampled from those consistent with a known set of precedence constraints. The values of the observed variables are sampled from the recorded examples of human motion (if a_i is sampled at a certain step as the performed human action, the value of the connected F_O is sampled from the ones associated with this particular action).

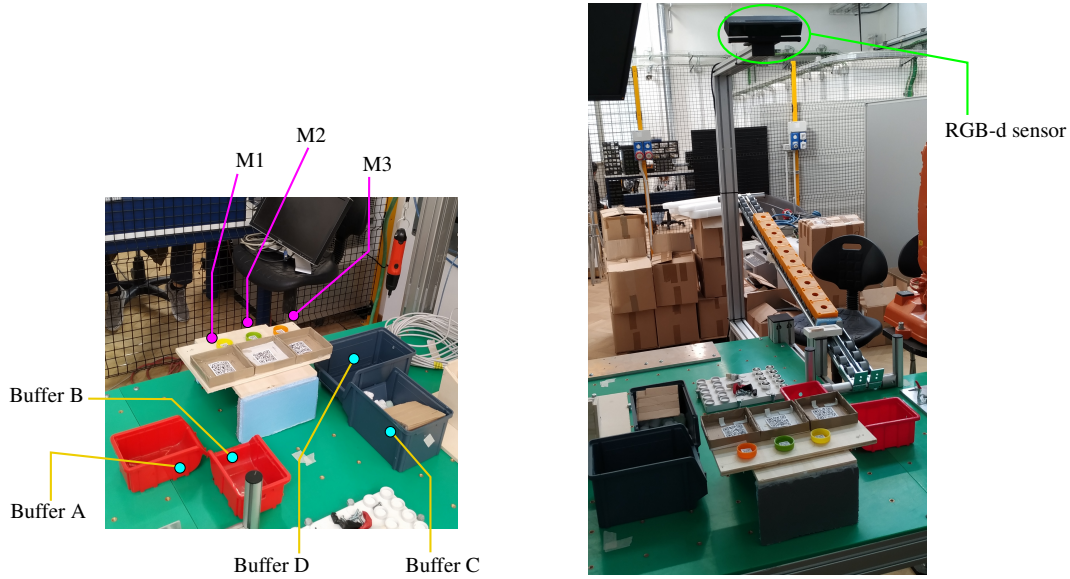


Figure 4.7: The experimental set-up taken by two distinctive perspectives. The operator's working area consists in 4 buffers storing parts and 3 assembly stations. Buffer A contains the first and the second kind of caps; B the springs and the lights; C the batteries and the batteries and the battery case while D the bodies of the torch.

4.2.1 Results

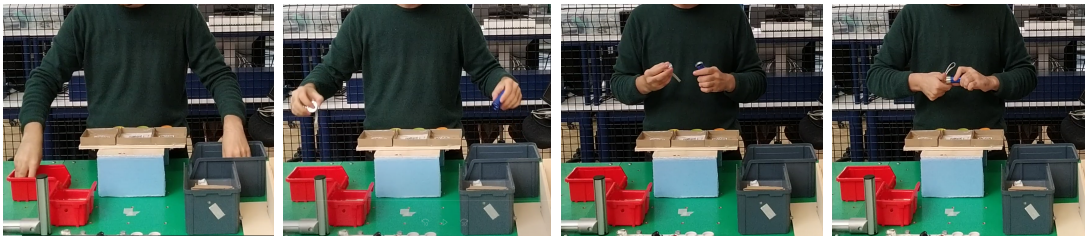
Every time a new batch of observations is available, the optimal ρ^* related to that temporal window is computed. Then, considering the graph structure induced by ρ^* (Section 4.1), the following marginal probabilities (see Appendix C.0.1) are computed:

$$X_i = [\mathbb{P}(Y_i = a_1) \quad \cdots \quad \mathbb{P}(Y_i = a_m)] \quad (4.7)$$

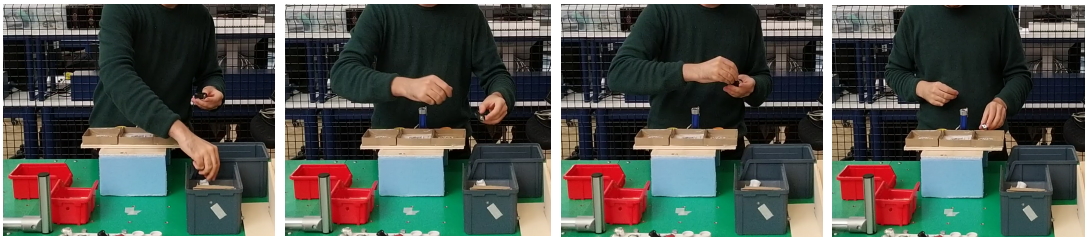
The above probabilities are associated to each hidden variable in the segmenting graph. The estimated duration of action Y_i is assumed to be equal to $T_i = l_w \cdot L \cdot \rho_i$. The time series represented by $X_{1,2,\dots}$ and $T_{1,2,\dots}$ is assumed to be the one describing the actions performed in time by the operator. The real sequence of actions happened within the same window is a time series $\hat{Y}_{1,2,\dots}$ and $\hat{T}_{1,2,\dots}$, with $\hat{Y}_i \in \mathcal{A}$. The segmentation error E_s can be evaluated considering the times for which the maximal values in X_i doesn't match with \hat{Y}_i , w.r.t the total number of observations retrieved from the camera. Statistics about E_s , on the data of the assembly task proposed in the previous Section, are reported in Figure 4.10. As can be seen, there is a positive effect in considering the positions of the goals storing raw materials w.r.t the case where only the inter-skeletal distances are exploited.

Figure 4.11, shows an example of segmentation obtained when considering the centres of the buffers of Figure 4.7: the segmentation is quite good at recognizing the starting and ending time of the assembly steps. Indeed, marginals of X_i are in good accordance with \hat{Y} .

Description of a_1 : After assembling the first kind of cap and the torch, the resulting work in progress is inserted into $M2$.



Description of a_2 : three batteries must be taken and inserted into the battery case. Then, the assembled components are put into $M3$.



Description of a_4 : the battery group is inserted into the body of the torch. Then a spring is taken and mounted.

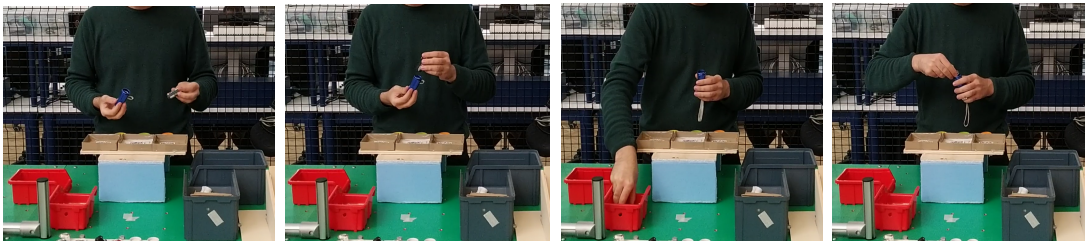
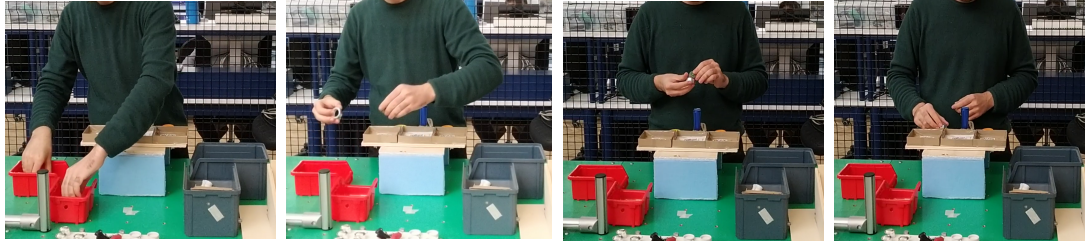


Figure 4.8: Description of a_1 , a_2 and a_4 . The locations of $M2$ and $M3$ is reported in Figure 4.7.

Chapter 4. Segmenting the human actions by analysing the upper body motion

Description of a_3 : after the light is assembled with the second kind of cap, the resulting work in progress is put into $M1$.



Description of a_5 : finalization of the product and storing in a buffer outside the working space.

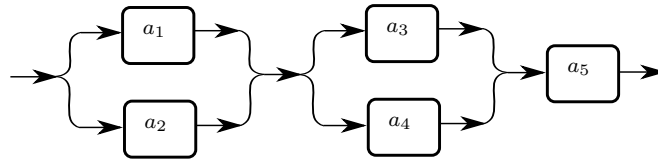
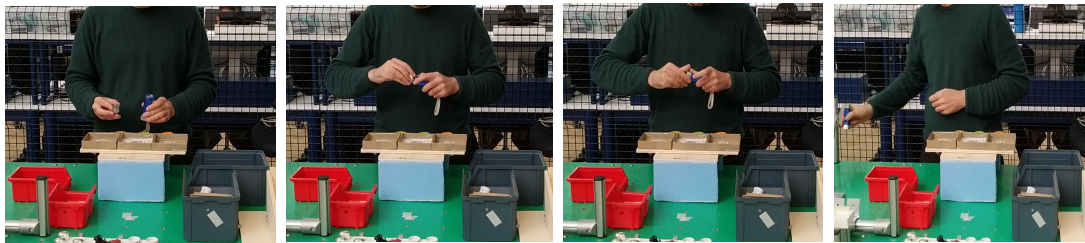


Figure 4.9: At the top, snapshots describing a_3 and a_5 , while at the bottom the precedence constraints existing among the actions. $M1$ is located as indicated in Figure 4.7.

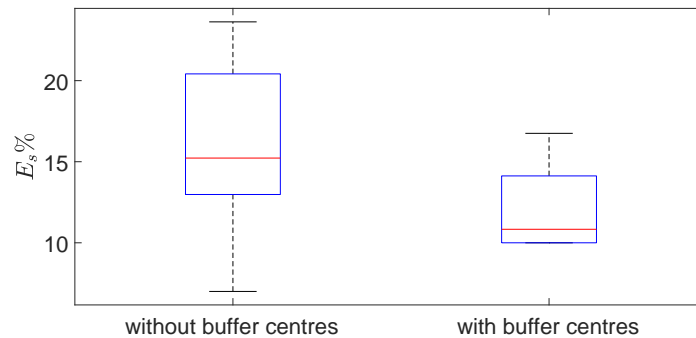


Figure 4.10: Distributions of the segmentation error E_s on the data acquired by the experiments when considering: on the right an approach considering all the combinations of possible inter-skeletal distances and the distances of the wrists (right and left) w.r.t. the center of the buffers indicated in the left part of Figure 4.7; on the left a similar approach not taking into account the centres of the buffers storing raw materials (only the distance of the wrists w.r.t. the camera origin is considered).

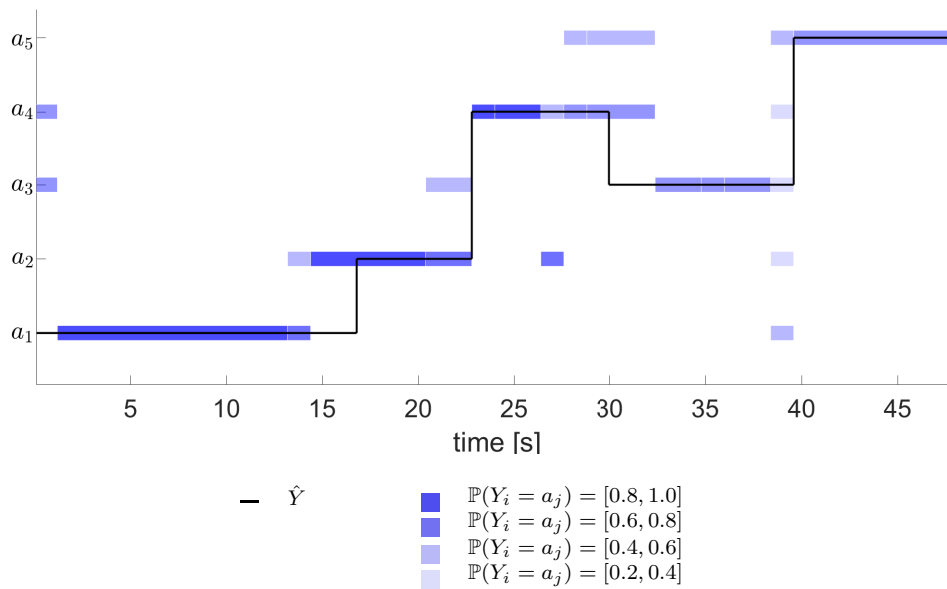


Figure 4.11: An example of obtained segmentation. The black curve refers to value assumed in time by the time series $\hat{Y}_{1,2,\dots}, \hat{T}_{1,2,\dots}$, while $X_{1,2,\dots}, T_{1,2,\dots}$ is reported in blue. Since X_i is a marginal distribution of probability, a color code is exploited for representing it, refer to legend on the right.

CHAPTER 5

Predicting the future activities

In typical collaborative scenarios, the human and the robot have to mutually synchronize and coordinate. For this reason, predicting the future human behaviour become crucial. The aim of this Chapter is to show how it is possible to predict human activity patterns. In particular, we are interested in evaluating the waiting time τ^a before seeing again a particular action $a \in \mathcal{A}$. Such a predicting problem is subdivided into two distinct ones: modelling the activity sequence and computing the waiting time. Two possible approaches will be proposed for modelling the activity sequence in Section 5.1 and 5.2, which can be equivalently used for predicting also the waiting time τ^a , whose computation is detailed in Section 5.3. The two predicting approaches will be compared in Section 5.4.

All the proposed methods are data driven: statistics about the duration of the human activities are collected during time as well as the past activities sequence, see Figure 5.1. In particular, the sequence of past actions are considered when updating the model describing the activity sequence (with the model of Section 5.1 or the one in Section 5.2), while the statistics about the time durations are exploited for predicting τ^a , see Section 5.3.

Human assembly sequences usually form quasi-repetitive patterns. In other words, the sequence of human activities can be modelled through a time series, which is the output of a certain dynamic process. Assume A_k as the ongoing activity at discrete time instant k , the behaviour of the human fellow co-worker can be modelled through the following discrete-time process

$$\begin{aligned} A_{k+1} &= f(A_k, A_{k-1}, A_{k-2}, \dots, A_{k-n}) \\ t_{k+1} &= t_k + g(A_k) \end{aligned} \tag{5.1}$$

where $t_k \in \mathbb{R}^+ \cup \{0\}$ represents the time instant corresponding to the transition from

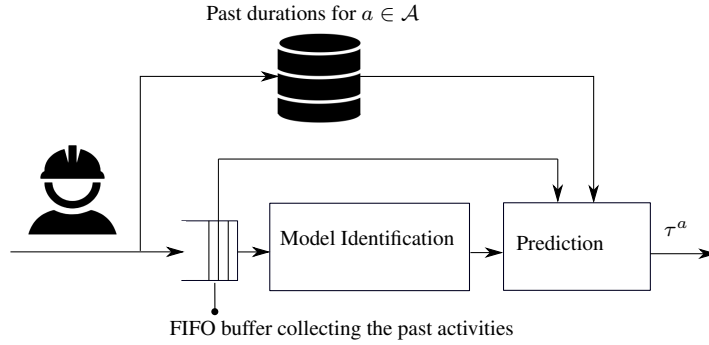


Figure 5.1: Approach followed for predicting the waiting times.

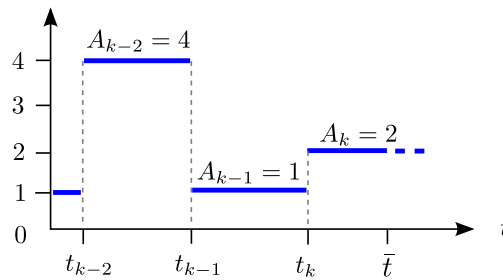


Figure 5.2: Example of behaviour of the human: t_k represents the time instant corresponding to the activation of activity A_k , while \bar{t} represents the current time-stamp.

A_{k-1} to A_k and $g(a) = T^a > 0$ is the duration of activity $a \in \mathcal{A}$. In the prediction model (5.1) the robot actions and their influence on the sequence of human actions are not explicitly accounted. However, as the parameters of model will be constantly updated using online data, the identified model will be indirectly influenced by robot actions in case they have influence on human ones. A possible time evolution of the process is represented in Fig. 5.2. The dynamic system in equation (5.1) is more easily identifiable in two stages. First, the identification of the underlying discrete event process governing the evolution of activities (*i.e.* the first equation) is addressed, regardless of their durations, which is the aim of this Section and the following one. Then, in Section 5.3 a model for the duration of the activities will be provided.

5.1 Higher Order Markov Model

The stochasticity of the underlying discrete process governing the sequence of activities can in principle be modelled by making use of Markov models (equivalently Markov Chain, MC). MC are able to express the evolution of a discrete time system, characterized by a finite number of possible states. In the case of predicting human patterns, the state at step k is represented by A_k . In a MC, the evolution of the system is governed

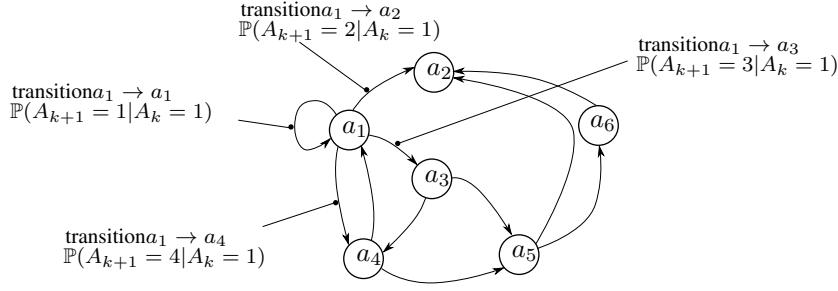


Figure 5.3: Example of transitions governing a Markov model. Transitions $a_1 \rightarrow a_1$, $a_1 \rightarrow a_2$, $a_1 \rightarrow a_3$ and $a_1 \rightarrow a_4$ are governed by the conditional probability distribution expressed by q^1 . q^1 has null values for rows 5 and 6, since the corresponding transitions are not allowed in this example.

by the transition probability matrix Q :

$$\mathbb{P}(A_{k+1} = i) = \sum_{j=1}^m \mathbb{P}(A_{k+1} = i | A_k = j) \mathbb{P}(A_k = j) \quad (5.2)$$

$$\begin{aligned} \begin{bmatrix} \mathbb{P}(A_{k+1} = 1) \\ \vdots \\ \mathbb{P}(A_{k+1} = m) \end{bmatrix} &= \begin{bmatrix} \mathbb{P}(A_{k+1} = 1 | A_k = 1) & \cdots & \mathbb{P}(A_{k+1} = 1 | A_k = m) \\ \vdots & \ddots & \vdots \\ \mathbb{P}(A_{k+1} = m | A_k = 1) & \cdots & \mathbb{P}(A_{k+1} = m | A_k = m) \end{bmatrix} \begin{bmatrix} \mathbb{P}(A_k = 1) \\ \vdots \\ \mathbb{P}(A_k = m) \end{bmatrix} \\ \begin{bmatrix} \mathbb{P}(A_{k+1} = 1) \\ \vdots \\ \mathbb{P}(A_{k+1} = m) \end{bmatrix} &= Q \begin{bmatrix} \mathbb{P}(A_k = 1) \\ \vdots \\ \mathbb{P}(A_k = m) \end{bmatrix} \end{aligned} \quad (5.3)$$

The j^{th} column of Q , q^j , is a discrete conditional probability distribution and the sum of the values in q^j must be equal to 1. Markov models are conceptually similar to state machines, with the important difference that the transitions between states happen randomly, refer to Figure 5.3.

Let $X_{k+1} = [\mathbb{P}(A_{k+1} = 1) \cdots \mathbb{P}(A_{k+1} = m)]^T$ be the probability distribution describing the state of the system at step $k + 1$. X_{k+1} can be computed using the following compact form (refer to equation (5.2)):

$$X_{k+1} = QX_k \quad (5.4)$$

The above equation is the equation of motion of a discrete time dynamical system having X as state. Propagating forward in time the above equation we can obtain:

$$\begin{aligned} X_N &= QX_{N-1} \\ X_N &= Q(QX_{N-2}) \\ X_N &= Q^N X_0 \end{aligned} \quad (5.5)$$

One important property characterizing MC is the memory absence¹. Indeed, refer to equation (5.2), the computation of the probability distribution of A_{k+1} is influenced only by A_k . Equivalently, the determination of X_{k+1} is made considering only X_k . In

¹The memory absence term is here used with a little bit of abuse, since the memory of a MC is represented by the state X_k itself, without taking into account the preceding ones.

other words, the transition that the system will do from step k to $k + 1$ depends only upon the state reached at step k , no matter the sequence of states in the trajectory leading to A_k .

Markov Chains, or in general Hidden Markov Models (HMM)s, have been extensively used in the literature to model and predict human behaviour in collaborative tasks, [47, 69]. However, common manufacturing or assembly activities are difficult to model as Markov Chains, *i.e.* implying that the next activity only depends on the current one (which is inherent in the Markov's assumption above introduced). This restricts the modelling capabilities of strictly Markovian processes, which are weakly able to capture periodic or repetitive patterns of actions which are however common in assembly stations. In fact, in manufacturing environments (and especially in assembly) the next activity to be performed does not depend solely on the current one. In other words the process has a memory longer than one step (the whole sequence of assembly steps). Also, the authors of [70] reported the same limitation of HMMs to model long-term causality dependencies between actions.

Better results can be achieved if the human behaviour is modelled with an higher-order Markov model (HOMM), as we are interested in computing the probability associated to the next activity (or more in general to the next sequence of activities, if evaluated recursively) given a set of previous ones, *i.e.* :

$$\mathbb{P}(A_{k+1} = a | A_k = k_0, A_{k-1} = k_1, \dots, A_{k-n} = k_n). \quad (5.6)$$

Differently from usual Markov Chains, generic higher-order Markov Chains require $m^{n+1} (m - 1)$ parameters to be estimated, resulting in an exponential complexity with respect to the order of the stochastic process to be identified.

The work from Raftery [103, 104] proposed an efficient way to describe higher-order Markov Chains using Mixture Transition Distribution (MTD) models. Specifically, the probability distribution in equation (5.6) is represented as

$$\begin{aligned} \mathbb{P}(A_{k+1} = a | A_k = k_0, \dots, A_{k-n} = k_n) &\approx \\ &\approx \sum_{i=0}^n \lambda_i \mathbb{P}(A_{k+1} = a | A_{k-i} = k_i) \end{aligned} \quad (5.7)$$

hence as a convex combination of multiple-steps transition probabilities, *i.e.* it is a mixture model (Section 3.1.2) involving discrete distributions. This model, that corresponds to usual Markov Chains for $n = 0$, requires only $m^2 (n + 1)$ parameters. According to the work from Raftery, [104], a prediction of the probability distribution \hat{X}_{k+1} at time $k + 1$ can be computed as

$$\hat{X}_{k+1} = \sum_{i=0}^n \lambda_i Q_i X_{k-i} \quad (5.8)$$

where $m \times m$ matrix Q_i denotes the i -steps transition probability matrix that can be simply evaluated through count statistics. As for the online estimation of the weights λ_i from data, differently from Ching et al. [26] who adopted an estimate of the stationary distribution X_∞ , [136] introduced a data-driven procedure. Using all the available evaluations until the present time instant one can evaluate the squared norm of the prediction error, *i.e.* $\left\| \hat{X}_{k+1} - X_{k+1} \right\|^2 = \left\| \sum_{i=0}^n \lambda_i Q_i X_{k-i} - X_{k+1} \right\|^2$. By stacking all

these evaluations available for different values of k , *i.e.*

$$\begin{bmatrix} Q_1 X_{k-2} \lambda_1 + Q_2 X_{k-3} \lambda_2 + \dots \\ Q_1 X_{k-3} \lambda_1 + Q_2 X_{k-4} \lambda_2 + \dots \\ \vdots \end{bmatrix} - \begin{bmatrix} X_k \\ X_{k-1} \\ \vdots \end{bmatrix} = A\lambda - b$$

the optimal solution for the λ_i 's parameters can be obtained by a non negative least-squares problem of the following type:

$$\min_{\lambda} \|A\lambda - b\|^2 \text{ subject to } \sum_{i=0}^n \lambda_i = 1, \text{ and } \lambda_i \geq 0 \quad (5.9)$$

where the column vector λ collects all the unknown parameters λ_i , while the regression matrix A and vector b can be simply evaluated from data.

The effectiveness of HOMM w.r.t. simpler Markovian models will be now discussed considering some artificial data adopted as benchmark. Set \mathcal{A} was assumed to have 5 elements and the sequence $a_1 \rightarrow a_2 \rightarrow a_3 \rightarrow a_1 \rightarrow a_2 \rightarrow a_4 \rightarrow a_3 \rightarrow a_5$ has been repeated for 75 times, and random mutations with probability 3% have been applied. The same resulting sequence has been processed with different algorithms.

Figure 5.4 reports the average 1-step ahead prediction error for the analysed methods: every models is constantly re-fitted, considering the data in the FIFO buffer (Figure 5.1), whose length is varied for comparing the performance. The error reported in Figure 5.4 refers to the times for which the maximal value in X_k was in a row different from the real action a_k performed at step k .

It can be noticed that HOMM sensibly outperforms the others, especially for high lengths of the FIFO buffer. The reason is due to the least-squared optimisation method which provide robustness to the algorithm in case of quasi-periodic patterns. Moreover, the limited prediction capabilities of Markov Chains in case of higher-order causality can be also appreciated. Finally, it is worth noticing that the memory storage required by the VOMM method proposed in [114] is linear with respect to the length of the FIFO buffer, and significantly higher then the one required by the methods based on Markov chains, which, in turn, does not depend on the length of the FIFO buffer. Anyway, for better performance in case of a sudden change of pattern, the length of the FIFO buffer should be kept at a minimum. It follows that a rough knowledge of length of the typical pattern is necessary for the algorithm to achieve the best prediction accuracy.

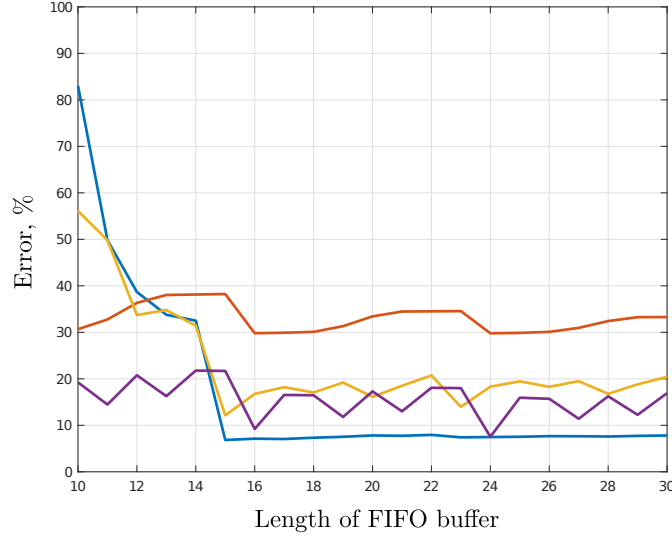


Figure 5.4: Comparison of different methods in terms of prediction error: the presented algorithm ($n = 7$, blue), the VOMM method proposed in [114] (purple), a Markov Chain model ($n = 0$, red), a higher-order Markov Chain models trained with the algorithm proposed by Ching et al. in [26] ($n = 7$, yellow).

5.2 Suffix Tree

It is reasonable to assume the existence of affine subsets $\mathcal{C} \subset \mathcal{A}$ of actions, for which the probability to be executed in sequence is high. The approach of Section 5.1 does not seem to allow considering such knowledge, without severe modifications. On the opposite, the aim of the approach proposed here is meant to predict the human actions by considering both the aforementioned a-priori knowledge as well as a data-driven model, whose evolution depends upon the actions performed during time by the operator.

For the rest of this Section, A_k^σ will be adopted for referring to the ordered sequence of actions $A_k^\sigma = A_{k-\sigma} \triangleright \cdots \triangleright A_{k-2} \triangleright A_{k-1}$ preceding A_k in the time series A .

5.2.1 The predictive model

The probability of $A_k \in \mathcal{A}$ conditioned to A_k^σ is characterized in the following way:

$$\mathbb{P}(A_k = a | A_k^\sigma) = \frac{\Psi(A_k^\sigma, a, t)}{\sum_{\tilde{a} \in \mathcal{A}} \Psi(A_k^\sigma, \tilde{a}, t)} = \frac{\Psi(A_k^\sigma, a, t)}{Z(t)} \quad (5.10)$$

The factors characterizing Ψ are all exponentials:

$$\Psi = \Psi_v(A_k^\sigma, a, t) \cdot \prod_{i=1}^{N_C} \Psi_{C_i}(A_k^\sigma, a) \quad (5.11)$$

$$\Psi_v = \exp\left(w_0 \Phi_v(A_k^\sigma, a, t)\right) \quad (5.12)$$

$$\Psi_{C_i} = \exp\left(w_i \Phi_{C_i}(A_k^\sigma, a)\right) \quad (5.13)$$

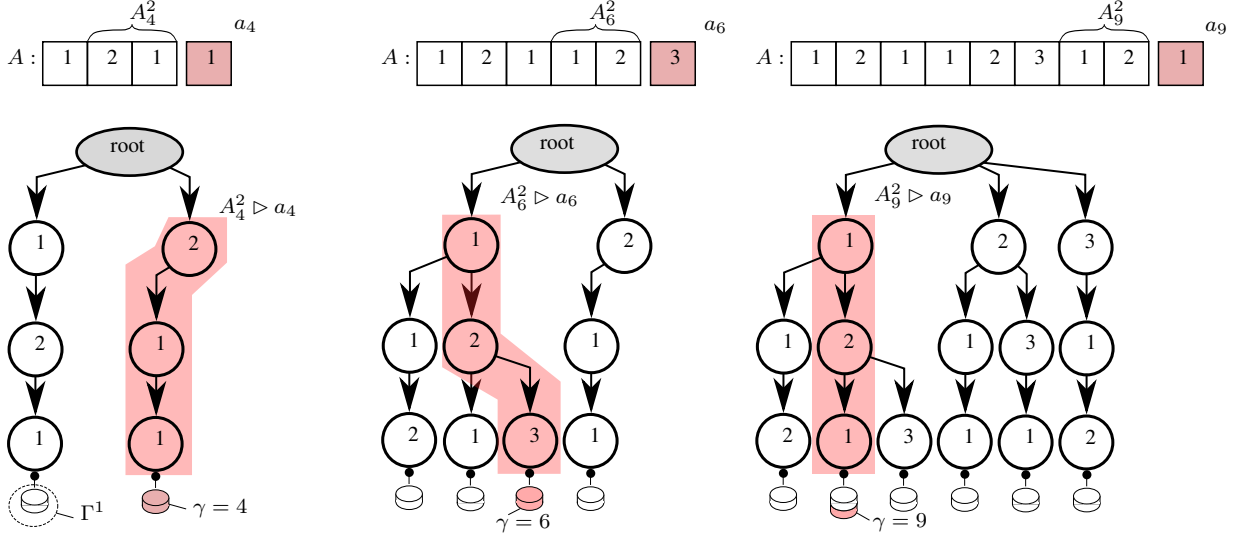


Figure 5.5: Examples of suffix tree updates. The structure of the tree after the update is reported for each example. The token sets Γ associated to the leaves are indicated in the lower part of the pictures containing the trees.

Φ_v is a piecewise time varying function, depending on the definition of a suffix tree, see Section 5.2.1. On the opposite, functions $\Phi_{c_1, \dots, C}$ remain invariant and are assumed as given. They model an a-priori knowledge to be used in the prediction process, see Section 5.2.1. Equation (5.10) is used for the single step prediction. Then, by recursively propagating it, the probability of a sequence $A_k \triangleright A_{k+1} \triangleright \dots \triangleright A_{k+L}$, conditioned to A_k^σ can be also evaluated. The computations for $A_k \triangleright A_{k+1}$ will be detailed, then it is easy to extend the reasoning to the general case:

$$\begin{aligned} \mathbb{P}(A_k = a_0, A_{k+1} = a_1 | A_k^\sigma) &= p_k \cdot p_{k+1} \\ p_k &= \mathbb{P}(A_k = a_0 | A_k^\sigma) \\ p_{k+1} &= \mathbb{P}(A_{k+1} = a_1 | A_{k-\sigma+1} \triangleright \dots \triangleright A_{k-1} \triangleright a_0) \end{aligned} \quad (5.14)$$

Both p_k and p_{k+1} are computable by making use of equation (5.10). On the other hand, the conditional probability of A_{k+L} w.r.t A_k^σ , regardless the intermediate values $A_{k, \dots, k+L-1}$ can be computed with the following summation:

$$\begin{aligned} \mathbb{P}(A_{k+L} = a_L | A_k^\sigma) &= \\ \sum_{\tilde{a}_0, \dots, \tilde{a}_{L-1} \in \mathcal{A} \times \dots \times \mathcal{A}} \mathbb{P}(A_k = \tilde{a}_0, \dots, A_{k+L} = a_L | A_k^\sigma) \end{aligned} \quad (5.15)$$

The above expression can be evaluated by making use of equation (5.14).

Definition of the suffix tree

A suffix tree (ST) is a time varying structure: every time a new value A_{k+1} is available, the tree should be updated. A ST describes in a compact way the information contained in the sequence $A_0 \triangleright \dots \triangleright A_k$. To every node, excluding the root, an action $a \in \mathcal{A}$ is assigned. The path connecting the root with the i^{th} leaf, also called branch, is denoted as B^j and is an ordered sequence of actions $a_{Bj1} \triangleright a_{Bj2} \triangleright \dots$. The population of all

$$Y = \{2, 1, 3, 3, 2\}$$

$$\begin{array}{c} \left[\begin{array}{c} \mathcal{I}[Y]_1 = 1 \\ \mathcal{I}[Y]_2 = 2 \\ \mathcal{I}[Y]_3 = 2 \end{array} \right] \qquad \left[\begin{array}{c|c} \mathcal{O}[Y]_1^1 = 2 & \mathcal{O}[Y]_1^2 = 0 \\ \mathcal{O}[Y]_2^1 = 1 & \mathcal{O}[Y]_2^2 = 5 \\ \mathcal{O}[Y]_3^1 = 3 & \mathcal{O}[Y]_3^2 = 4 \end{array} \right] \end{array}$$

Table 5.1: Results obtained when applying operators \mathcal{I} and \mathcal{O} on the series Y reported at the top.

the branches of the tree contains all the observed sub-sequences in A , up to step k . The construction of the tree is made considering a particular order σ . $\sigma + 1$ will be the length of every branch. A set of tokens $\Gamma^j = \{\gamma_1^j, \gamma_2^j, \dots\}$ is assigned to the j^{th} leaf, whose meaning will be clear later.

Every ST is initialized with the presence of the sole root. The sequence $A_1 \triangleright \dots \triangleright A_{\sigma+1}$ is inserted as first branch B^1 at step $\sigma + 1$, i.e. after observing the first $\sigma + 1$ values of A . At the same step, set Γ^1 is initialized with a single token $\gamma_1^1 = \sigma + 1$. Then, at the generic step k the ST is updated in this way:

- Case a): $A_k^\sigma \triangleright A_k$ is already contained in the ST, i.e. there exists a branch $B^j = A_k^\sigma \triangleright A_k$. In this case, a token equal to k is added to Γ^j , i.e. $\Gamma^j = \Gamma^j \cup k$.
- Case b): $A_k^\sigma \triangleright A_k$ is not present in the ST. In this circumstance, a new branch $B^M = A_k^\sigma \triangleright A_k$ is inserted in the tree, whose corresponding set Γ^M is initialized with the value k .

Fig. 5.5 reports some examples. Function Φ_v , equation (5.12) depends on the structure of a ST. Prior to define Φ_v , the operators $\mathcal{I}[\cdot]$ and $\mathcal{O}[\cdot]$ must be introduced. \mathcal{I} describes the actions contained in a sequence $Y = y_1 \triangleright \dots \triangleright y_s$, regardless their order, and is defined as follows:

$$\mathcal{I}[Y]_{a_i} = \sum_{j=1}^s L(y_j)_{a_i} \quad a_i \in \mathcal{A} \quad (5.16)$$

where the indicator function L is here defined:

$$L(y_j)_a = \begin{cases} 1 & \text{if } y_j = a \\ 0 & \text{otherwise} \end{cases} \quad (5.17)$$

On the opposite, \mathcal{O} aims at describing the order of actions in a sequence:

$$\mathcal{O}[Y]_{a_i}^K = \begin{cases} 0 & \text{if } \mathcal{I}[Y]_{a_i} < K \\ k & \text{if } \mathcal{I}[Y_k^k \triangleright y_k]_{a_i} = K \wedge y_k = a_i \end{cases} \quad (5.18)$$

Refer to the example reported in Table 5.1. Two possible distances, d_I and d_O can express the similarity existing between two sequences X and Y . They are defined

according to the two previously introduced operators:

$$d_I(X, Y) = \sum_{i=1}^{m=|A|} \left| \mathcal{I}[X]_i - \mathcal{I}[Y]_i \right| \quad (5.19)$$

$$d_O(X, Y) = \sum_{j=1}^{+\infty} \sum_{i=1}^{m=|A|} \left| \mathcal{O}[X]_i^j - \mathcal{O}[Y]_i^j \right| \quad (5.20)$$

The domain of Φ_v is divided into three disjoint regions $\mathcal{D}_I(ST)$, $\mathcal{D}_{II}(ST)$, $\mathcal{D}_{III}(ST)$ (refer to equation (5.12)):

$$\Phi_v(A_k^\sigma, a|ST) = \begin{cases} \Phi_{vI} & \text{if } \{A_k^\sigma, a\} \in \mathcal{D}_1 \\ \Phi_{vII} & \text{if } \{A_k^\sigma, a\} \in \mathcal{D}_2 \\ \Phi_{vIII} & \text{if } \{A_k^\sigma, a\} \in \mathcal{D}_3 \end{cases} \quad (5.21)$$

Set \mathcal{D}_1 contains those sequence already existing in the ST. More formally:

$$\mathcal{D}_1 = \{A_k^\sigma, a \mid \exists B^j \in ST \text{ s.t. } B^j = A_k^\sigma \triangleright a\} \quad (5.22)$$

Then, the complement of \mathcal{D}_1 is divided into two parts: the first one contains all those sequences for which in the ST there exists at least one branch having the same actions (with a different order) while the second one contains all the remaining ones. Assume operator \mathcal{V} defined in the following way:

$$\mathcal{V}[X, ST] = \{B^j \in ST \mid d_I(B^j, X) = 0\} \quad (5.23)$$

Then, it holds that:

$$\mathcal{D}_2 = \{A_k^\sigma, a \mid \mathcal{V}[A_k^\sigma \triangleright a, ST] \neq \emptyset\} \quad (5.24)$$

$$\mathcal{D}_3 = \{A_k^\sigma, a \mid \mathcal{V}[A_k^\sigma \triangleright a, ST] = \emptyset\} \quad (5.25)$$

Figure 5.6 summarizes the above considerations, reporting an example of domain partition.

We are now in position to discuss the definition of Φ_{vI} , Φ_{vII} and Φ_{vIII} . To this purpose, the activation function f_{act}^Γ must be introduced:

$$f_{act}^\Gamma(n) = \sum_{\gamma \in \Gamma} \exp\left(-\alpha(n - \gamma)\right) \quad (5.26)$$

The parameter α is determined in order to verify that $f_{act}^\Gamma(N) \cong 0$ for $N > \frac{5}{\alpha}$, with N a desired forgetting time. Φ_{vI} is defined as follows:

$$\Phi_{vI}(A_k^\sigma, a) = f_{act}^{\Gamma^j}(k) \quad (5.27)$$

The Γ^j in the above equation is the one related to branch $B^j = A_k^\sigma \triangleright a$. Therefore, the aim of tokens is to activate more those sequences recently seen. However, since a summation is present in equation (5.26) a high activation value is provided also by

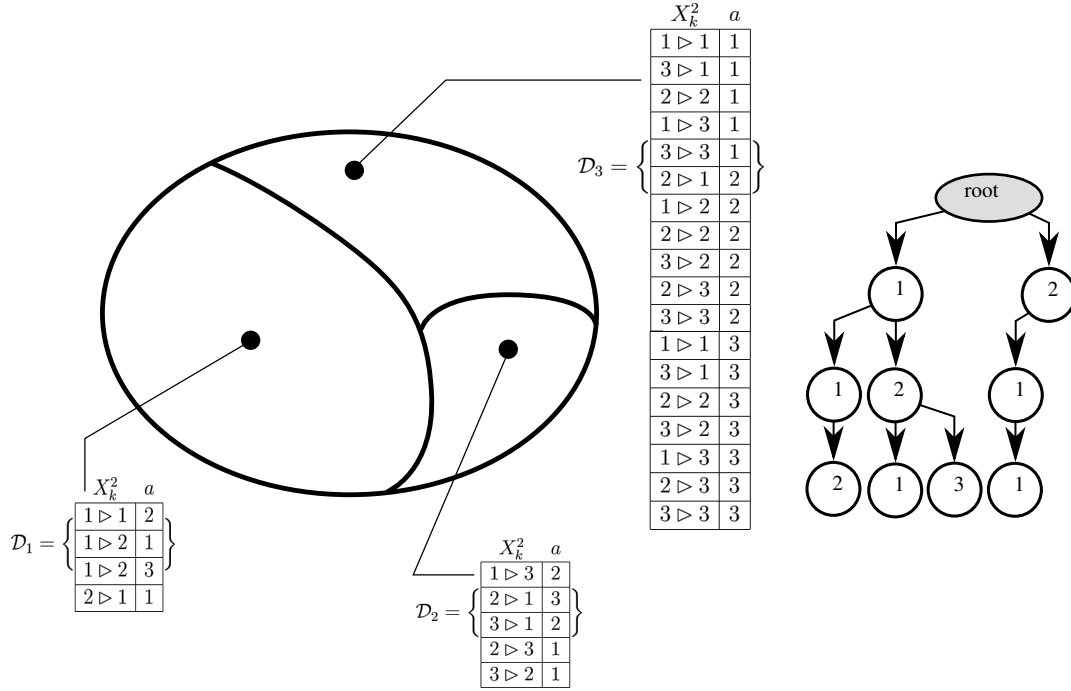


Figure 5.6: Example of domain ripartition. The left part of the Figure reports sets $\mathcal{D}_{1,2,3}$ when considering the suffix tree on the right.

those sequences seen many times. Φ_{vII} is defined in this way:

$$\Phi_{vII}(A_k^\sigma, a) = \frac{1}{|S|} \left(\sum_{B^i \in S} \frac{1}{\beta} f_{act}^{\Gamma^i}(k) \right) \quad (5.28)$$

$$\text{where } S = \mathcal{V}[A_k^\sigma \triangleright a, ST] \quad (5.29)$$

$$\beta = d_O(B^i, A_k^\sigma \triangleright a) \quad (5.30)$$

Finally, the definition of Φ_{vIII} is as follows:

$$\Phi_{vIII}(A_k^\sigma, a) = \frac{1}{|ST|} \left(\sum_{B^i \in ST} \frac{1}{\delta_i} f_{act}^{\Gamma^i}(k) \right) \quad (5.31)$$

$$\text{where } \delta_i = d_I(B^i, A_k^\sigma \triangleright a) + d_O(B^i, A_k^\sigma \triangleright a) \quad (5.32)$$

Handling the prior knowledge of the process

As humans, we are easily able to make predictions by exploiting contextual information. For instance, when someone takes a screwdriver, we naturally think that a subsequent action will involve screws. Similarly, when we see an operator gluing a surface, we guess that in the near future something will be attached. For this reason, the approach proposed here was developed for managing some prior information regarding the process to predict. More formally, the generic activation function Φ_{C_i} (equation (5.10)), expresses the circumstance that a subset of actions $\mathcal{C}_i \subseteq \mathcal{A}$ are affine. The

evaluation of Φ_{C_i} , is done as follows:

$$\Phi_{C_i}(A_k^\sigma, a) = \sum_{j=1}^{\sigma} L_{C_i}(A_{k-j}, a) \cdot \exp(-\alpha \cdot j)$$

$$\text{where } L_{C_i}(A, a) = \begin{cases} 1 & \text{if } A \in C_i \wedge A \neq a \\ -\frac{1}{|C_i|-1} & \text{if } A = a \\ 0 & \text{if } A \notin C_i \end{cases} \quad (5.33)$$

where α in the above equation has the same meaning of the one in equation (5.26). We assume the sub-sets $C_{1,2,\dots}$ as given: they can be easily determined by clustering actions with a strong ontological similarity (an extensive review of this topic can be found in [28]). The importance of the information provided by the a priori knowledge w.r.t the one contained in the predictive suffix tree (i.e. the data-driven one) discussed in the previous Section, is determined by tuning weights $w_{0,1,2,\dots}$ (equation (5.10)), which is the aim of training, see Section 5.2.1.

Tuning the model

The weights $w_{0,1,2,\dots}$, see Section 5.2.1, can be determined through learning. In fact, they can be determined in order to maximize the likelihood of A , up to a step K , i.e. considering all the known realizations $A_{1,\dots,K}$. The logarithm of the joint probability of all the values in A till K (equation (5.10)) can be determined as the following product:

$$\begin{aligned} L &= \log\left(\mathbb{P}(A_{\sigma+1}|A_{\sigma+1}^\sigma) \cdots \mathbb{P}(A_K|A_K^\sigma)\right) \\ &= \sum_{j=\sigma+1}^K \left(w_0 \Phi_v(A_j^\sigma, A_j|ST) + \sum_{C_i} w_{C_i} \Phi_{C_i}(A_j^\sigma, A_j|ST) - \log(Z(j)) \right) \end{aligned} \quad (5.34)$$

Since it is impossible to find the value maximising L in a closed form, a gradient ascent strategy can be adopted. To this purpose, the derivatives $\left[\frac{\partial L}{\partial w_0} \quad \frac{\partial L}{\partial w_{C_1}} \quad \frac{\partial L}{\partial w_{C_2}} \quad \dots\right]$ must be evaluated. It is not difficult to prove that their expressions are as follows (the computations are analogous to those discussed in Appendix C.0.2):

$$\frac{\partial L}{\partial w_0} = \sum_{j=\sigma+1}^K \left(\Phi_v(A_j^\sigma, A_j|ST) - \sum_{a \in \mathcal{A}} (\mathbb{P}(a|A_j^\sigma, ST) \cdot \Phi_v(A_j^\sigma, a|ST)) \right) \quad (5.35)$$

$$\frac{\partial L}{\partial w_{C_i}} = \sum_{j=\sigma+1}^K \left(\Phi_{C_i}(A_j^\sigma, x_j) - \sum_{a \in \mathcal{A}} (\mathbb{P}(a|A_j^\sigma, ST) \cdot \Phi_{C_i}(A_j^\sigma, a)) \right) \quad (5.36)$$

In principle, it is possible to re-train a model every time a new action A_k is observed, as indicate in the pipeline of Figure 5.1. In such a case, the update of the ST and a new learning of weights $w_0, w_{C_1}, w_{C_2}, \dots$ are done for every step. This is reasonable for the most of real contexts, since the human activity durations (which are in the order of seconds) are higher than the time required for performing the gradient ascent described (in the order of the milliseconds). However, an approach where learning is done only sporadically is also possible.

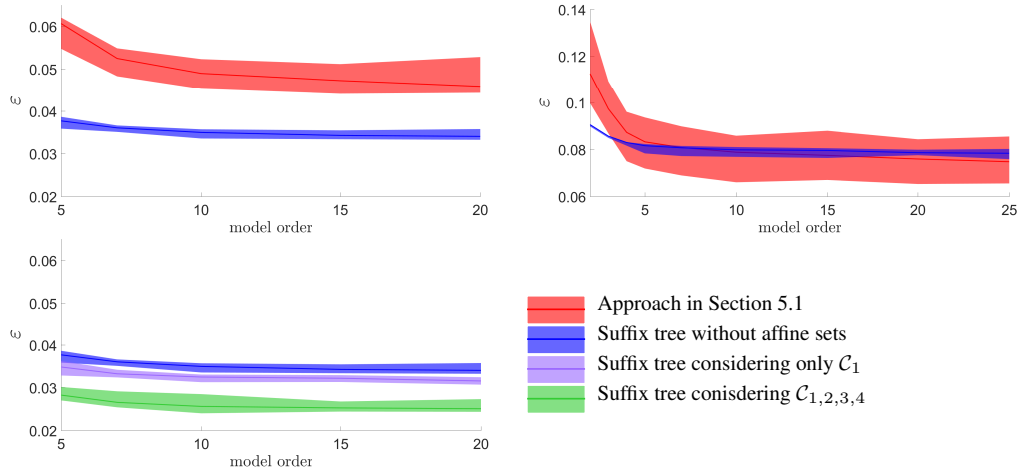


Figure 5.7: Statistics of the prediction error obtained from the simulations. In all the figures, model order refer to the number of previous actions taken into account for computing the one-step probability prediction (in case of suffix trees is clearly σ) and the curve of the 50th quartile is inserted into a shaded area delimited by the 80th quartile and the 20th one. The legend of reported in the right lower part applies. The pictures on the left part consider the complete assembly in Fig. 5.8, while the one the right takes into account the simplification reported in Fig. 5.9.

5.2.2 Comparison of the proposed predictive models

With the aim of comparing the developed approach described in Section 5.1, some off-line simulations were performed, considering the assembly of the emergency button reported in Fig. 5.8. Steps involved for the completion of a finite product are reported in the same Figure and are made of a series of forks and joints. All the operations in the same fork must be done before the succeeding ones, without a particular order (for instance the screws can be taken before the screwdriver and vice-versa). A population of artificial series A were created, by alternating 20 assembly cycles. Each cycle is a random sequence of operations consistent with the precedence constraints expressed in Fig. 5.8. Then, an error simulating the non perfect segmentation of human actions was introduced: the 5% of the elements of a A were replaced with random numbers. A total amount of 100 artificial series were generated for producing the results reported in the following. Comparisons are made computing the mean prediction error $\left\| \hat{X}_{k+1} - X_{k+1} \right\|^2$, see Section 5.1. The affine sets are made by considering the actions in the same fork ² leading to the definition of the following sets: $\mathcal{C}_1 = \{a_1, a_2\}$; $\mathcal{C}_2 = \{a_4, a_5, a_6\}$; $\mathcal{C}_3 = \{a_7, a_8\}$ and $\mathcal{C}_4 = \{a_9, a_{10}\}$.

The left pictures in Fig. 5.7 reports the results when considering the complete assembly process, while the result on the right part of the same Figure reports similar statistics but considering the simplified assembly reported in Fig. 5.9, for which the existence of the affine sets was ignored. As it can be seen, in the first kind of comparison, the curve of the mean prediction error of the proposed approach is completely below the one of the approach in Section 5.1. Moreover, the dispersion is lower (curves of the quantile are closer). Performance are significantly improved when introducing the affine sets and the best performance are achieved when all of them are taken into account. It is in-

²This can be also the systematic criterion to follow in industrial contexts.

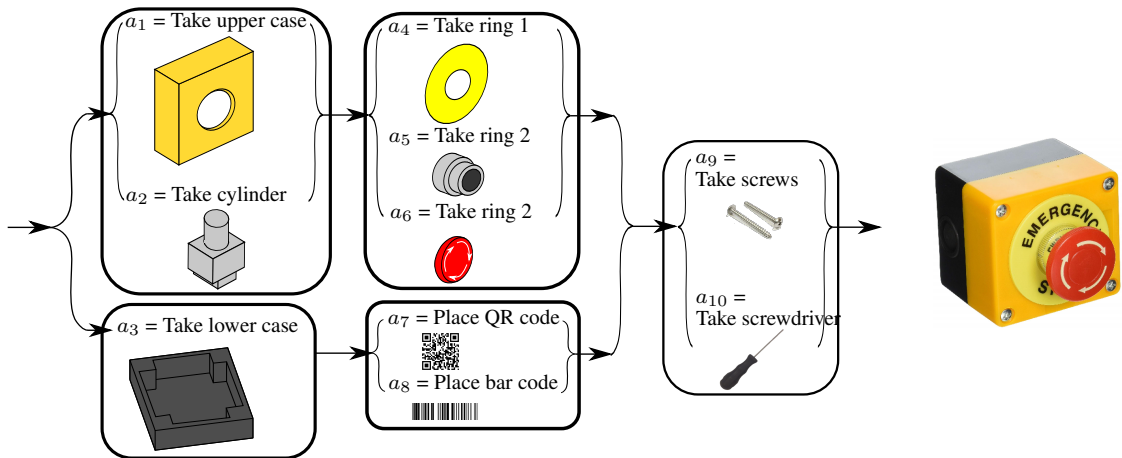


Figure 5.8: On the top left part the complete sequence of actions required for assembling an emergency button: the actions contained in a box can be done with no particular order, but before the actions contained in the boxes following in the sequence. A total number of 10 actions are needed to finalize the product. The top right part of the Figure reports the emergency button to assemble.

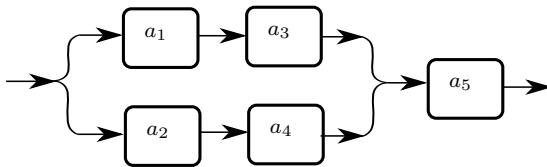


Figure 5.9: The Figure reports a simplification of the assembly reported in Figure 5.8, for which the size of set \mathcal{A} is equal to 5.

interesting to notice that the performance gap is reduced when considering the assembly simplification.

As a general consideration, the proposed approach seems to perform well also for low values of σ . Indeed, even with a low order, the suffix tree is able to represent the temporal correlations among the actions, by simply including more branches. For this reason a repeating pattern of actions with a length higher than the model order, would be anyway handled.

5.3 Evaluating waiting times

Both in Section 5.1 and 5.2, the human behaviour has been modelled as a sequence of activities, regardless of their duration. In order to predict in the most effective way when a certain activity is undertaken by the human, it is necessary to account for their time durations as well. We here assume that the duration of activity $a \in \mathcal{A}$, *i.e.* T^a , can be modelled as a stochastic variable with a strictly positive lower bound, *i.e.* $T^a \geq T_{min}^a > 0$.

In order to estimate the waiting time needed for the certain activity a to show up, say τ^a , we can combine this information with the models describing the activity sequence. In particular, at the present continuous time instant \bar{t} , given the sequence of the last activities (possibly including the currently running one) $A_k, A_{k-1}, \dots, A_{k-n}$, we would like to estimate the probability distribution of the waiting time for the beginning of a certain activity a , *i.e.* $\mathbb{P}(\tau^a \leq t | A_k, A_{k-1}, \dots, A_{k-n})$.

The key idea is to construct a predictive reachability tree. Then, evaluating the time spent to traverse each possible branch in the tree, terminating with the desired activity $a \in \mathcal{A}$, it is possible to estimate τ^a as will be discussed. Since the reachability tree is, in principle, infinite, a prediction horizon ΔT must be defined, meaning that the given probability will be computed up to the instant $t = \bar{t} + \Delta T$.

The probability associated to each branch can be simply computed by multiplying the probability of each arc of the branch, *i.e.*

$$p_{branch} = \prod_{(i,j) \in branch} P^{(i,j)}.$$

As for the waiting time associated to each branch τ_{branch} , this is simply the sum of the duration of each activity T^a , *i.e.*

$$\tau_{branch} = \sum_{j:(i,j) \in branch} T^j.$$

Notice that the elapsed time of the ongoing activity as well as the tails of the activities exceeding the prediction horizon ΔT have to be removed. The time associated to each branch is computed as the sum of stochastic variables which are not, in general, identically distributed. Moreover, neither the associated distribution nor its parameters are a priori known. Since it may turn out to be difficult to select a model to describe the probability distribution of the duration of each activity, the statistics associated to recently acquired samples can be directly used. Figure 5.10 also reports an example of distribution of duration of a certain activity.

Finally, given the distributions of the times associated to each branch, the overall distribution of the waiting time of the activity a can be simply computed as a weighted sum of the waiting times associated to each branch, *i.e.*

$$\begin{aligned} \mathbb{P}(\tau^a \leq t | A_k, A_{k-1}, \dots, A_{k-n}) &= \\ &= \sum_{branch} p_{branch} \mathbb{P}(\tau_{branch} \leq t). \end{aligned} \tag{5.37}$$

Figure 5.10 reports an example of the application of the developed method, refer also to the pipeline reported in Figure 5.1.

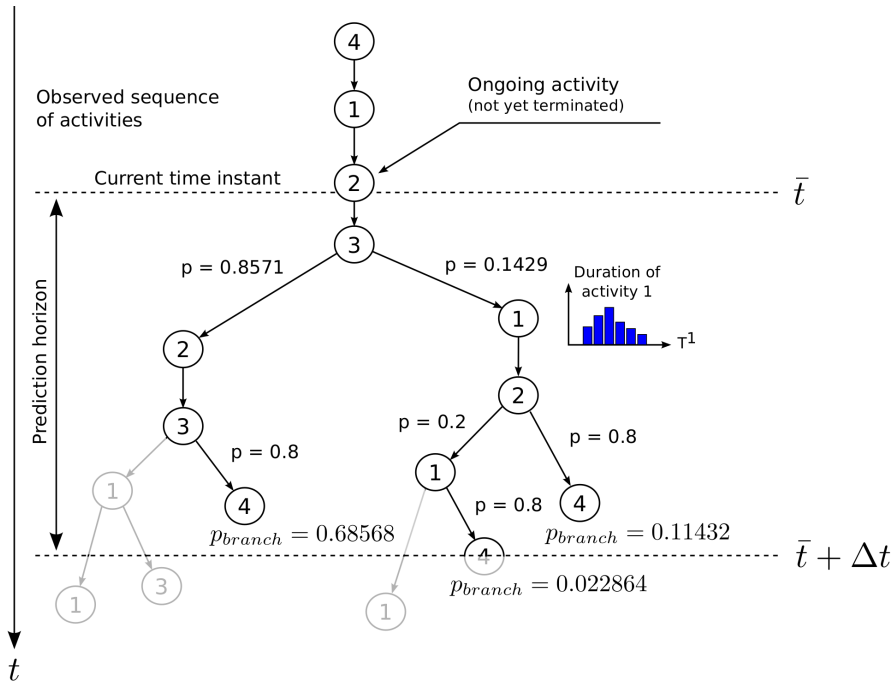


Figure 5.10: Example of prediction of human future activities. The transition probabilities associated to each arc are evaluated using a model for the activity sequence (see Section 5.1, 5.2). The lower bounds on the duration of each activities are used to prune branches of the tree that surely exceed the given prediction horizon ΔT . For all the remaining branches (three in the reported example), the corresponding distributions of waiting times τ_{branch} are computed and used within equation (5.37) to estimate the distribution of the waiting time needed for a certain activity to show up. In this example, the probability distribution of the waiting time of activity 4, i.e. τ^4 , is computed.

Algorithm 1 Reachability Tree Expansion

```

1: procedure TREEEXPAND(Activity sequence model ,  $\Delta T$ ,  $a \in \mathcal{A}$ )
2:   while true do
3:     set the root as an expanded leaf;
4:     if all leaves expanded then
5:       return;
6:     else
7:       pick a non expanded leaf;
8:       if current leaf corresponds to activity  $a$  then
9:         mark current node as expanded;
10:      else
11:        compute  $\{\mathbb{P}(a_1|branch), \dots, \mathbb{P}(a_m|branch)\}$ ;
12:        append  $m$  leaves to the current node;
13:        for each leaf do
14:          set  $\tau_{branch} = \sum_{j:(i,j) \in branch} T_{min}^j$ ;
15:          evaluate  $p_{branch}$ ;
16:          if  $\tau_{branch}^{min} > \Delta T$  or  $p_{branch} < \varepsilon$  then
17:            mark current node as expanded;
    
```

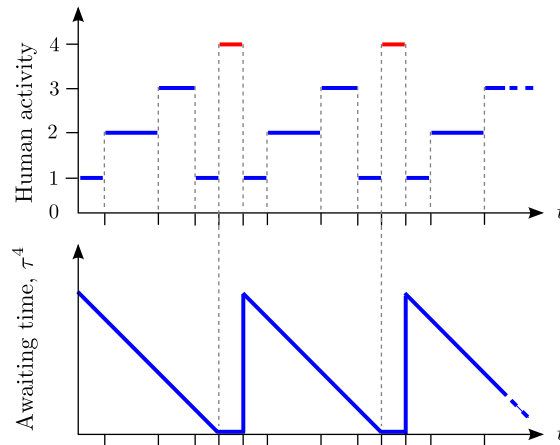


Figure 5.11: Example of sequence of activities (top) and corresponding typical behaviour of the estimate of the waiting time of activity 4, i.e. τ^4 , (bottom) evaluated and continuously updated during time.

When the described algorithm is run continuously at a certain frequency, an updated estimation of the waiting time for a certain activity to arrive is available at each iteration. As an example, Fig. 5.11 reports the typical behaviour of the output of the algorithm corresponding to a certain activity sequence.

5.4 Performance comparison

In this Section the effectiveness of the proposed predictive algorithms will be tested in various use cases, representing realistic human-robot collaborative assemblies, involving ABB dual-arm robot YUMI. A MICROSOFT KINECT depth camera is used to acquire the positions of the human's hands in order to recognize the sequence of performed operations.

5.4.1 Use case a

In this case the human and the robot actively cooperate to perform the assembly of a PCB board to be accommodated within an IP 54 plastic enclosure. A picture of the experimental setup is shown in Fig. 5.12.

Task description and implementation

The human is responsible for an autonomous task which consists in assembling an integrated circuit into a socket already soldered onto a PCB. In turn, the robot is responsible for verifying the quality of the resulting assembly. The different phases of the assembly procedure are shown in Fig. 5.13.

The method described in Section 5.3, adopting HOMM for modelling the activity sequence, Section 5.1, is considered for computing the probability distribution of the waiting time for a certain human activity to show up. Should this activity require some kind of assistive behaviour from the robot, it is essential for the robot task planner to know whether a subtask can be initiated or not. Within the present use case, the robot is responsible for an autonomous activity (quality check) but also for being of assistance to the human in holding, like a third hand, the cap while the operator is fixing the

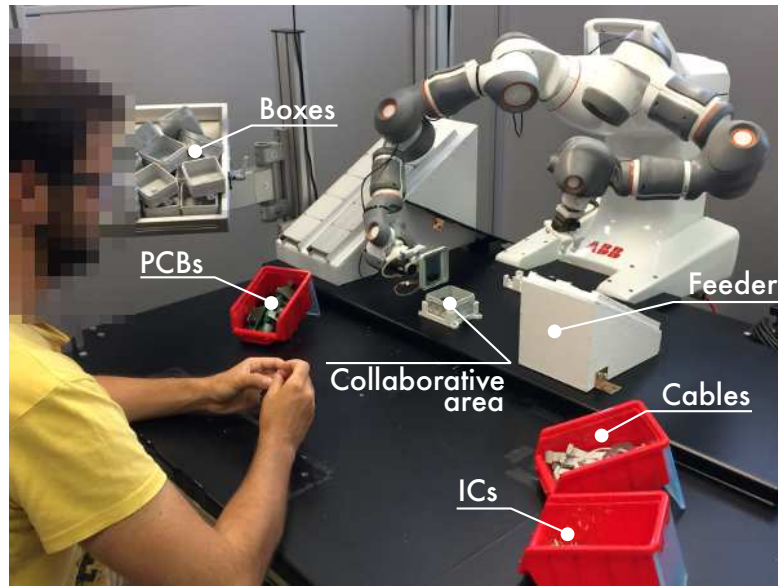


Figure 5.12: Layout of the experimental setup: the human can access six stations, the central one being dedicated to the collaboration with the robot.

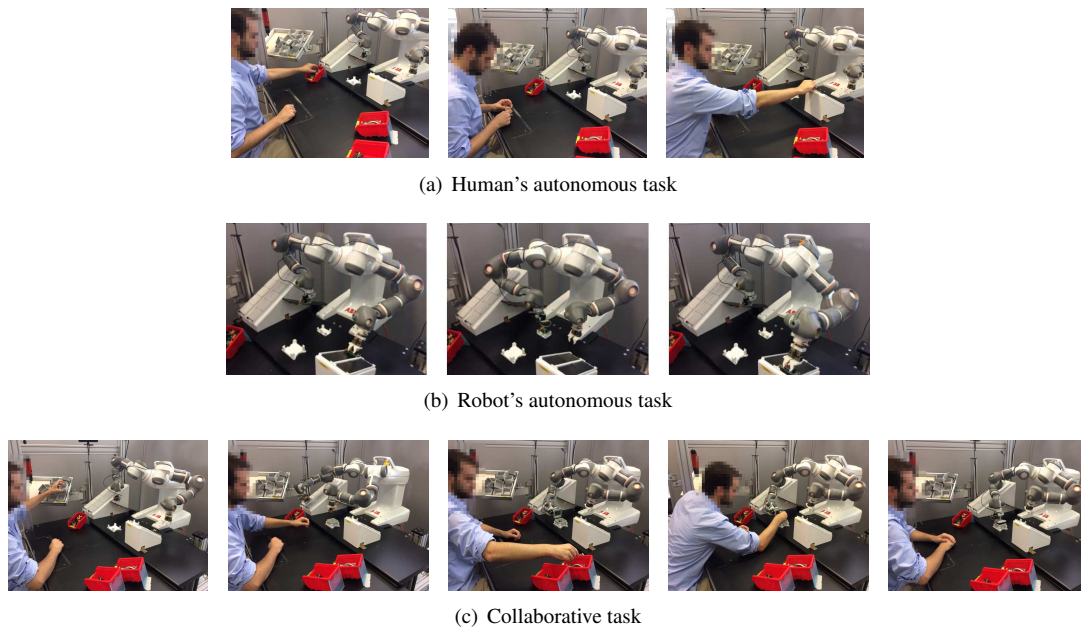


Figure 5.13: Different phases of the assembly procedure. **IC insertion** (top left): the human takes a PCB board from the red box on the left and an IC from the red rightmost box, inserts the IC in the pre-soldered socket, and finally fills the feeder. **Quality check** (top right): the robot takes a PCB from the feeder, accommodates it within a fixture, then it takes a picture of the PCB using the in-hand camera, and finally drops it on the feeder. **Flat assembly and finalisation** (bottom): the human takes a plastic enclosure from the left tray and places it in the fixture in front of the robot within the collaborative area, the robot picks a verified PCB and places it inside the enclosure, the human takes a flat cable from the right red box, meanwhile the robot takes the cap from a feeder and assists the human while fixing the cable on it, the robot accommodates the cap on the enclosure and finally stores the finished part.

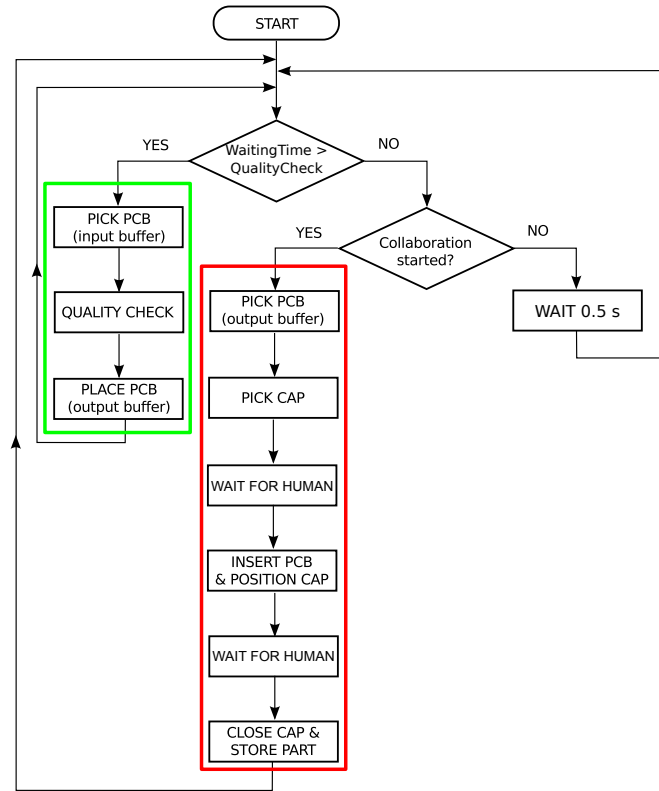


Figure 5.14: Workflow of the robot program: based on *WaitingTime*, i.e. the p -percentile t_p returned by the algorithm, the first decision the robot takes is whether to wait for the human to initiate the collaborative operation (on the right, grey box) or to start the autonomous subtask (on the left, green box). The collaborative operation (in the middle, orange box) starts when initiated by the human.

flat cable. Therefore, the waiting time for the collaborative operation is constantly estimated and, when needed, compared to the execution time of the quality check. If the time remaining before the collaborative operation is larger than the time the robot needs to complete the quality inspection of one part, the robot initiates its autonomous task. Otherwise, the robot waits in order to be ready to assist the human during the collaborative operation. This behaviour has been coded within the robot programming language and the corresponding flow chart is reported in Fig. 5.14. This is a simplified scheduling approach that will be greatly extended by the approaches described in Part II. Every time a human activity is terminated, the HOMM model parameters Q_i and λ_i are updated.

Experiments and discussion

For validation, two different experiments have been run. The second experiment has been run for comparison: the predictive algorithm has been disabled and the robot keeps executing its autonomous task, unless the human operator has already initiated the collaborative operation. In other words, the robot implements a purely reactive strategy. On the contrary, the strategy exploiting the predictive algorithm can be considered a proactive one.

5.4. Performance comparison

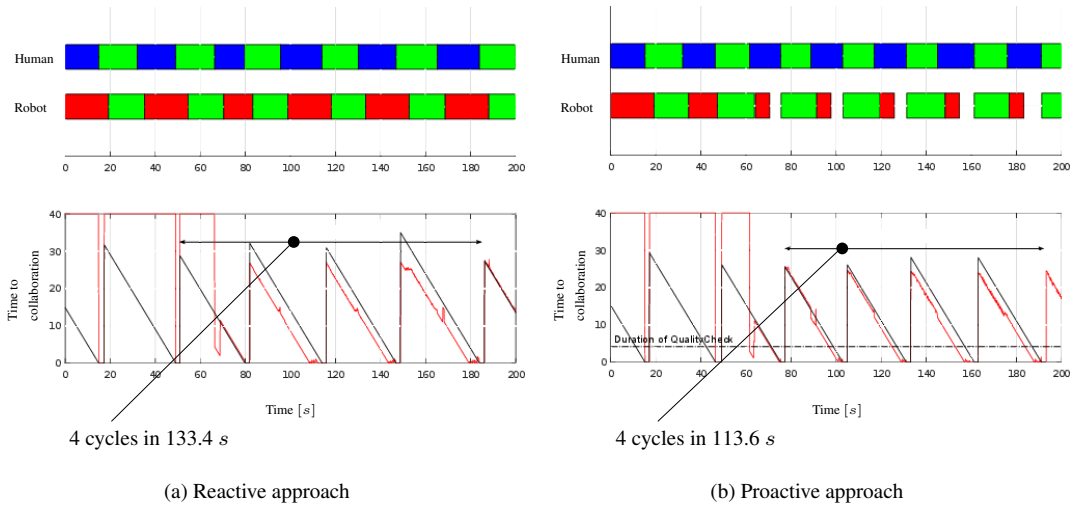


Figure 5.15: Execution of the collaborative assembly experiment with the reactive (left) and the proactive (right) approach. The top Figures shows the sequence of activities of the human left hand and of the robot (blue and red represent autonomous activities, while the collaborative operation is marked in green). The bottom figures show the predicted time to collaboration (picking a box from the left tray, see Fig. 5.12) as compared to the ground truth (black).

Figure 5.15 reports the sequence of activities performed by the robot and the human, together with the estimate of the waiting time until the request for collaboration. As one can see, after a training phase lasting around 60 s, which is required for the method to collect enough data, the robot is able to schedule the right operation, *i.e.* to wait for the human to initiate the collaborative task instead of initiating its autonomous assignment, which would have caused the human to wait before being assisted.

For comparison, during the second experiment the same assembly task is executed without a prediction about the human behaviour, but using a purely reactive approach. Differently from the previous case, the robot is always assigned to the autonomous task, unless the human has already initiated the collaborative part. As one can notice from Fig. 5.15 the overall execution of the last complete four assembly cycles takes around 20 s more (133.4 s vs. 113.6 s with the proposed approach), which corresponds to an increase of 17% in terms of throughput, thus confirming that predicting the human behaviour is able to let the robot plan in a better way the assigned actions. Moreover, in collaborative applications, and because of safety limitations, robots are typically slower than caged industrial manipulators. Thanks to the developed technology, the possibility to reduce the cycle time and thus improve the efficiency of the assembly cycle would further boost the return on investment (ROI) of collaborative robots. As a further confirmation, Fig. 5.16 reports the distribution of the cycle time corresponding to the reactive and the prediction based approaches. As already stated, the second outperforms the reactive one in terms of a reduced cycle time (Wilcoxon signed rank right-tail test, $r = 0.9848$), Fig. 5.16 also shows that the variability can be reduced (F-test, $r = 0.9705$) by adopting a proactive behaviour. Overall, the prediction algorithm based on a HOMM performs well in predicting the time before the next demand for collaboration from the human. As one can see from Fig. 5.15, the estimated waiting time is slightly underestimated with respect to the ground truth value, and results in a saw-toothed profile with respect to time, as expected.

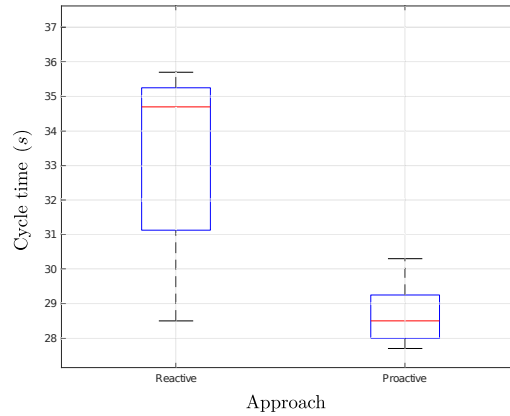


Figure 5.16: Distribution of cycle times of the whole assembly sequence with the two approaches. The approach considering the predictive algorithm is responsible of a higher throughput as well as a reduced variability in cycle times.

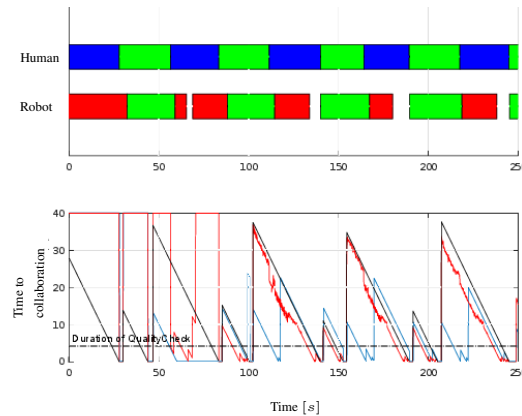


Figure 5.17: Execution of the collaborative assembly when the human adopts a different pattern which consists in two consecutive IC insertions and two consecutive collaborative operations. The notation is identical to the one of Fig. 5.17, except from the blue curve which represents a purely data-drive approach.

So far the, a *one piece flow* pattern has been adopted by the human operator. In different production scenarios, some other patterns can be also adopted. Another experiment has been performed to test the capabilities of the algorithm. In particular, the human adopted pattern which consisting in two consecutive IC insertions and two consecutive collaborative operations. The results are reported in Fig. 5.17. It is worth noticing that the duration of the time interval between the beginning of two consecutive collaborative operations now assumes a bimodal distribution. For this reason, any other approach based solely on this information will be surely less precise than any other method that attempts to model what happens between two consecutive events with a higher granularity. In Fig. 5.17 a comparison between the method based upon HOMM and a purely data-drive approach is reported. The latter is obtained by collecting the time intervals between two consecutive requests for collaboration (up to the present time instant), and the prediction is made extracting the same percentile from the obtained distribution. As one can notice the proposed method significantly outperforms the other in predicting the remaining time before the next demand for collaboration from the human operator.



Part II

Assistive scheduling

Scheduling of the robotic actions

Common industrial plants are populated by many machines, handling different kind of dedicated operations. Machines are seen as resources of the system and the jobs to perform compete for their usage. In order to maximise the throughput, scheduling algorithms must be applied. This is done typically considering every machine of the plant as a controllable agent, whose actions can be imposed during time, with the aim of reacting to both forecasted and unexpected events. To this aim, cyber-physical approaches are becoming popular [67], [87]. Essentially, a model of the plant, called digital twin, is considered for representing the plant state and evaluate the consequences of alternative scheduled plans, assuming to have the complete control of all the agents of the plants. When considering collaborative stations, the above approach is not easy to implement, due to the weakly controllable nature of human operators. Those techniques conceiving the human as an additional quasi-controllable agent which receives during time notifications about the work to do, are ineffective. Moreover, in such cases the human is treated no more than a high cognitive robot, performing the operations that are difficult to be executed by common industrial robots.

On the opposite, in this work the whole system of a collaborative station (*i.e.* all the robots constituting the cell) is intended to assist human workers, performing during time complementary operations. In this way, the operator is allowed to drive the interaction. The problem becomes essentially to decide the most convenient assistive action to impose to cobots, taking into account the future human behaviour. Such behaviour will be predicted by making use of the algorithms discussed in Chapter 5, which provide an estimation about the time at which the operator would prefer to perform every assigned action. Then, according to the preferences of the operator, a scheduled plan for the system is produced and robots are instructed to perform the actions in this plan. Therefore, with the proposed approach, the human is directly influencing the future be-

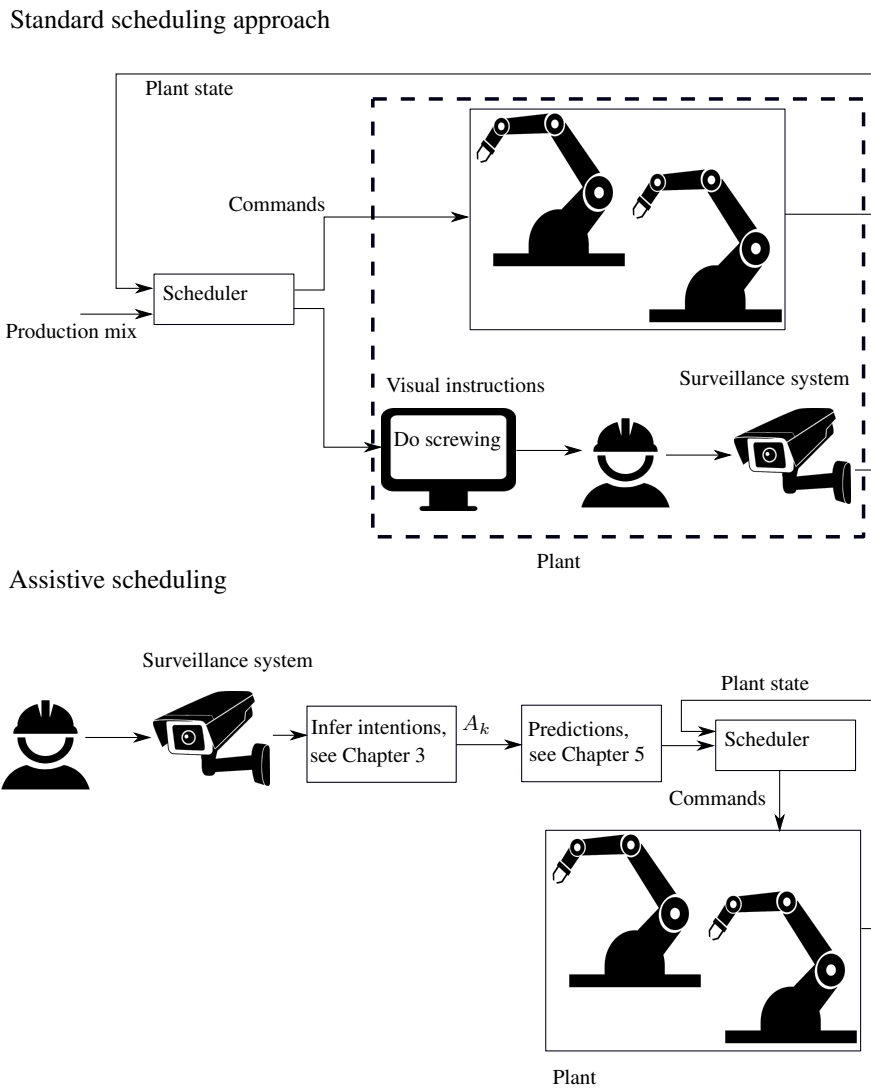


Figure 6.1: Differences between a standard scheduling approach, on the top, and an assistive scheduling, on the bottom.

haviour of the plant, refer to Figure 6.1 (Assistive scheduling).

All the proposed approaches have been implemented in a receding horizon fashion. Indeed, new predictions for the human behaviour are constantly produced. Accordingly, new plans are computed and the first actions in the plans are applied to the robots, see Section 7.1.

The next part of this Chapter will review some general concepts, which will be exploited in Chapter 7 for explaining the developed scheduling approaches.

6.1 Petri Nets as scheduling tools

When performing collaborative assemblies, humans and robots compose a certain number of parts into complex finite products, executing many intermediate operations like, for instance, taking parts from intermediate buffers, performing the screwing of an object into another, and so on. Prior to performing any kind of scheduling, tasks have to be allocated to agents, *i.e.* to the humans and to the robots of the cell. A dynamic approach is in principle possible, solving on line and simultaneously the allocation and the scheduling problems. However, this work assumes a static task allocation: the assembly flow¹ is identified and actions are off line assigned to agents, taking into account their specific capabilities. The problem amounts to optimally schedule the operations assigned to robots, since, as already discussed, the human is treated like a non controllable agent.

An example of assembly flow is reported in Figure 6.2, which describes the operations required to assemble three distinct products. The AND of Figure 6.2 indicates some compositional tasks, requiring to take multiple work in progress (WIP) from preceding buffers. The actions reported in Figure 6.2, will be executed concurrently by agents, assuming the task allocation reported in Figure 6.3. Although this is not the case in the example provided, a single action may require to be executed with the collaboration of multiple agents (robots-robots, or human-robots).

The presence of intermediate buffers storing WIP, see Figure 6.2, must be taken into account when scheduling. In particular, the non infinite capacity of such buffers is something that the scheduler must be aware of. This is not completely true when considering the buffers containing raw materials, *i.e.* the ones not having a predecessor in the assembly flow. Indeed, such buffers are assumed to be constantly refurnished and therefore we can assume they persistently contain an infinite number of items.

When scheduling, alternative evolutions of the system are compared, in order to understand the best actions to do. For this purpose, the collaborative cell can be modelled as a Timed Petri Net (TPN). This modelling tool is suited for describing the temporal concurrencies in a system, which are in case of collaborative assemblies, the actions performed during time by both humans and robots. The Petri Nets modelling the system are built according to the assembly flow characterizing a particular collaborative assembly. Section 6.3 will propose a systematic way to translate a large class of assembly flows into PN.

As it is well known, Petri Nets are adopted for modelling discrete event systems, having agents that share finite resources. Some formal definitions will be now provided.

Petri Nets (PN) [90] are bipartite graphs made of transitions and places. Places are marked with tokens, describing in a compact way the state of the system. The firing of a transition in a PN, represents the occurrence of an event, leading the system to a new state, since the firing of a transition alters the positions of the tokens inside the net. A transition may fire only if enabled, in the sense that some preconditions must be met. More formally, a PN is a tuple $\langle P, T, Pre, Post, m_0 \rangle$, where: P is the set of places; T the set of transitions; Pre and $Post$ are matrices such that $P \times T \rightarrow \mathbb{N}$ defines the flow

¹The assembly flow is an intuitive formalism adopted for indicating the precedence constraints existing among actions, see the examples provided in the following.

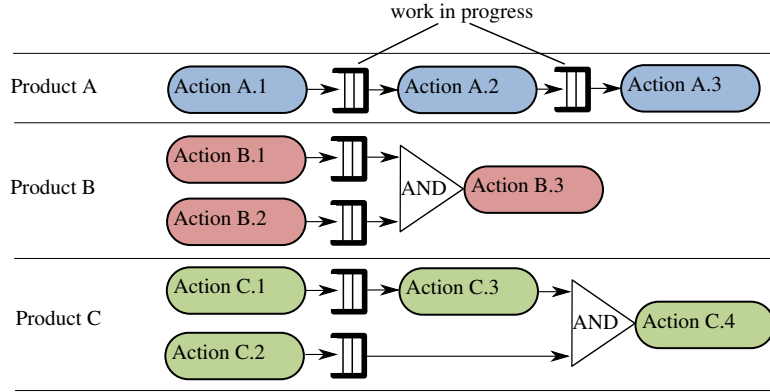


Figure 6.2: The assembly flow of a mix involving three different products. All actions with the same color have to be performed to produce a single finite product.

Human	Robot 1	Robot 2
Action A.2	Action A.1	Action A.3
Action B.3	Action B.1	Action B.2
Action C.1	Action C.3	Action C.2
Action C.4		

Figure 6.3: Example of task allocation. Actions refer to the assembly flow of Figure 6.2

of tokens through the net, while m_0 is the initial marking. m is a vector of numbers specifying the number of tokens that all places have for a particular state. Pre and $Post$ are incidence matrices defining the preconditions to be met for enabling every transition, as well as the token flow deriving from the firing events. More precisely, the j^{th} transition in T , is enabled if and only if for every place i it is satisfied that:

$$Pre^{ij} < m^i \tag{6.1}$$

where Pre^{ij} refers to the i^{th} row and j^{th} column of matrix Pre , while m^i is the number of token present in the i^{th} place. Suppose transition j is enabled in the marking m_k . Then, in case that transition fires, the new marking m_{k+1} reached is computable as:

$$m_{k+1} = m_k + Post^j - Pre^j \tag{6.2}$$

where $Post^j$ and Pre^j refer to the j^{th} column of the respective matrices.

6.2 Temporal Petri Nets: main concepts

Time Petri Nets [120] are a particular class of PNs, embedding the concept of time. In such kind of nets, transitions not necessarily fire instantaneously, but a particular firing delay for each transition is assumed. Such delays model the fact that the actions represented by transitions take a certain amount of time to be accomplished. Every transition in the net has a distinctive firing delay. More formally, TPNs are PNs, for which to each transition T a firing delay d is assigned. d can be a deterministic number, or more in

general a quantity suitable to represent uncertainty, like for example a probability distribution. When a transition becomes enabled, its firing must occur after a delay d from the enabling event. Otherwise, the firing of a transition can be disabled only by the firing of another one, which makes the preconditions of the first one no longer met.

Many kinds of TPN were proposed in the literature. However, this work will address only partially controllable TPNs. They are a particular class of TPN specifically derived for describing collaborative assembly. Two particular subclasses will be discussed: TPNs having the generic delay d characterized by a probability density function (PDF) or described by a fuzzy number.

Among all kind of TPN for which all the firing delays are described by a PDF, Stochastic Petri Net (SPN) [86] are the most popular one. In particular, every transition in a SPN is described by an exponential PDF. This implies, see Section 6.2.1, that SPNs have the memory absence property, making the temporal evolution of these nets equivalent to Markovian Processes. However, it was already discussed in Chapter 5.1 that when dealing with collaborative assembly, the Markovian property is not realistic, meaning that SPN cannot be adopted. Indeed, the framework proposed is able to deal with any kind of distributions, generalizing SPN.

A partially controllable Timed Petri Net (pcTPN) is a TPN for which the set T is partitioned as $T = T_c \cup T_u, T_c \cap T_u = \emptyset$, as similarly done in Ramadge and Wonham [109]. T_u contains all the uncontrollable transitions, for which the firing mechanism is the one described so far, valid for any kind of TPN. Instead, T_c contains all the controllable transitions, for which firing occurs at a relative time with respect to the enabling event, that is a control variable, *i.e.* its value can be decided within an interval $[0, +\infty)$.

For modelling collaborative assemblies, controllable transitions must be introduced alternatively to the uncontrollable ones. Indeed, transitions in T_c will be associated to the activation of certain tasks, while those in T_u are introduced to model their possible durations. For example, an autonomous task performed either by a robot or the human, will be represented as a transition in T_c followed by a single transition in T_u , see the nets reported in Section 7.2.1, 7.3.1 and 7.5.2. On the other hand, tasks shared between more than one agent can be represented by a controllable transition, followed by several others in T_u , to be fired in sequence or in parallel, see Section 7.2.1. In this way, the fact that some agents can finish their subtasks before the others is modelled.

In the following, the generic transition of a net will be indicate with the letter t . Then, for referring to a controllable transition, the notation \bar{t} will be used.

6.2.1 Reachability Tree

Timed Petri Nets are exploited to evaluate possible alternative evolutions of the system. With this aim, the Reachability Tree (RT) of the system is computed. RT is a collection of states $S_{1,2,\dots}$, that are reachable within a certain temporal range from the initial state of the system. A connection between two states in a RT, implies that the firing of a certain transition allows to reach one state from the other.

It could be in principle possible to compute an RT, by considering in a parametric way the firing delays of controllable transitions. This amounts essentially to consider possible different versions of the RT, one for every combination of delays imposed for the transitions in T_c . Such an approach would end up to be too much computationally de-

manding and will be therefore not followed. On the opposite, we'll assume transitions in T_c to have only two possible firing delays: 0 or $+\infty$. This implies that controllable transitions may be fired or not, but in case of firing they are instantaneously fired.

The states in a RT are reached from the root by firing a particular sequence of transitions. Apart from the root, every state S_i have a father $S_f = fath(S_i)$, which represents the preceding node in the RT, *i.e.* the state to reach before S_i . After firing a transition t from S_f , node S_i is reached. Formally, states $S_{0,1,2...} \in RT$ are characterized in this way:

$$S_i = \langle m_i, \alpha_i \rangle \quad (6.3)$$

m_i is the marking of the modelling Petri Net associated to S_i , while α_i is the corresponding arrival time. The root S_0 of a RT is a node describing the state of the system at the present time, therefore $\alpha_0 = 0$ for any kind of nets. Figure 6.4 reports an example of RT. Notice that the tree reported in Figure 6.4 is similar to the one of Figure 5.10. However, when dealing with TPN, the computation of arrival times as well as the probability to get to the end of a particular path is more articulated, as it will be discussed. If node S_a is the father of node S_b , this implies that the transition t_{ab} leading from S_a to S_b is enabled in S_a , but at the same time it does not imply that t_{ab} is enabled just upon arriving in S_a , since it could have been enabled earlier. Let $(S_a; S_c)$ be the path connecting S_a to S_c . The farthest enabling ancestor of the generic node S_i , is the node $S_e \in (S_0; S_i)$, such that $(S_e; S_i)$ is the largest subpath that contains only nodes for which the transition leading to S_i is enabled. In other words, S_e is the node in the RT for which the transition leading to S_i becomes to be enabled. The above definition clearly does not exclude the root S_0 to be the farthest enabling ancestor of some other nodes. The following notation will be adopted: $S_e = enab(S_i)$. When considering deterministic firing delays, the RT is a simple chain of states, whose arrival time is a given real number. However, both the presence of controllable transitions, as well as the uncertainties related to the firing delays of the uncontrollable ones, make alternative evolutions of the system possible, which leads to a tree structure. In particular, a RT will be populated by conflicts, which can be controllable or not.

A controllable conflict arises when reaching a state for which multiple controllable transitions are enabled². On the opposite, when the only transitions enabled are uncontrollable, an uncontrollable conflict arises. When dealing with controllable conflicts, the alternative future evolutions of the system must be considered in order to decide which transition to fire. Performing the latter choice is in essence the aim of scheduling. Since the firing of controllable transitions is instantaneous, the node reached in the RT after firing has the same arrival time of its father.

On the other hand, handling of uncontrollable transitions is more complex. In fact, most of the time it is not true that all transitions in such conflicts become enabled just after arriving in the same state. This implies that the arrival time α of the furthest enabling ancestor of every enabled transition, plays a fundamental role for computing the arrival time in the children of a node having an uncontrollable conflict. As an introductory example, consider the Example A of Figure 6.5. Both transitions 1 and 2 are enabled in the initial state of the net. The arrival times in S_1 and S_2 (Figure 6.6) is an

²It is irrelevant to consider the non controllable ones in such cases, since controllable transitions fire instantaneously.

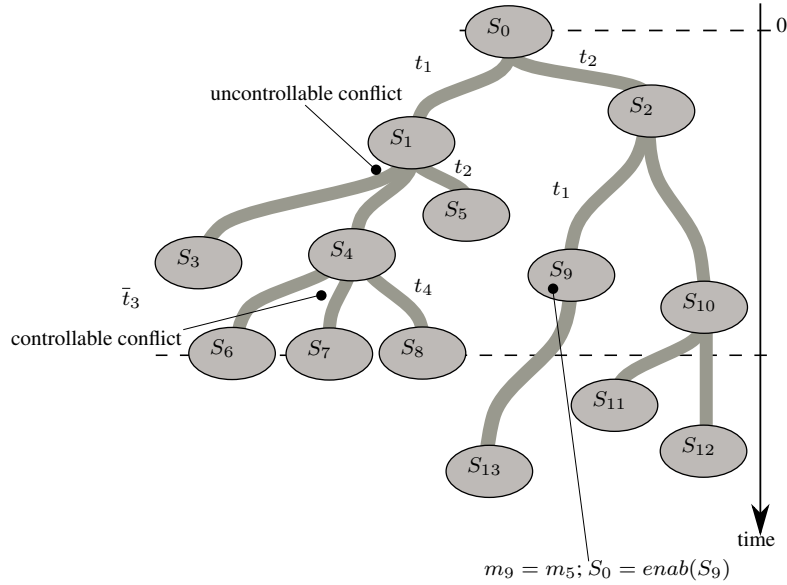


Figure 6.4: Portion of a reachability tree. Nodes $S_{6,7,8}$ are reached by firing controllable transitions, i.e. $\alpha_4 = \alpha_{6,7,8}$. $m_9 = m_5$ since the same kind of transitions lead to the corresponding nodes. However, S_9 and S_5 are reached with a different order of firing, implying that the arrival times to that nodes are different.

uncertain quantity, governed by a PDF. When not considering the presence of the conflict, the support of α_1 is simply assumed as the interval $[0, 1]$, while for α_2 is $[0, 0.75]$. However, an uncontrollable transition may be disabled only by the firing of another one. Therefore, after a time equal to 0.75, transition 2 is forced to fire. Therefore, it is not possible to get to S_1 at a time equal to 0.8 for example, since prior to that instant transition 2 would have already fired, leading to state S_2 . Indeed, the real support of α_1 is $[0, 0.75]$.

Example B of Figure 6.5 depicts a similar net. However, in this new example, transitions 1 and 2 are preceded by transitions 3 and 4, which are deterministic. Considering the reachability tree of this new net, Figure 6.6, another uncontrollable conflict arises between transition 1 and 2. Anyway, for this conflict, the transitions involved become enabled in different times. This is something to consider when computing the support (or, more in general, when characterizing the entire arrival time distribution, see the following Section) of α_1 . The supports of the unconditioned arrival times in S_1 and S_2 , would be equal in this case to $[0, 1]$ and $[0.5, 1.25]$ respectively. Then, for similar considerations exposed for the previous example, the real support of α_2 would end up to be equal to $[0.5, 1]$.

Basically, the examples provided, tell us that it is important to characterize for a state S_i , not only the marking m_i , but also the particular trajectory (i.e. the particular sequence of transitions fired, controllable or not) leading to S_i , since it is crucial for computing the arrival time in that node as well as the one of the successors in the RT. As will be shown, the only class of nets for which the latter consideration is not true are SPN. Indeed, when considering the RT of a SPN, the particular trajectory followed to get to a state does not influence the further evolution of the system. Therefore, for the only

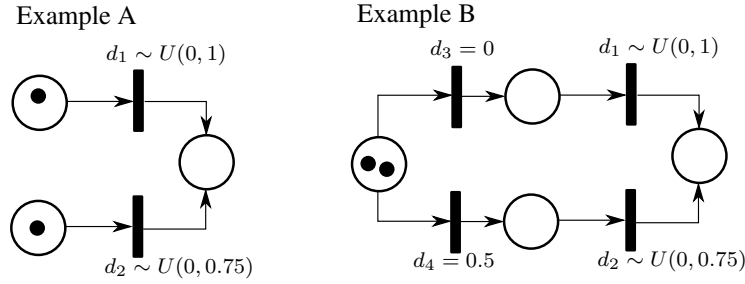


Figure 6.5: Examples of TPN. All the transitions reported are uncontrollable. Transitions 3 and 4 in Example B are deterministic, while for both the examples transitions 1 and 2 are uniformly distributed.

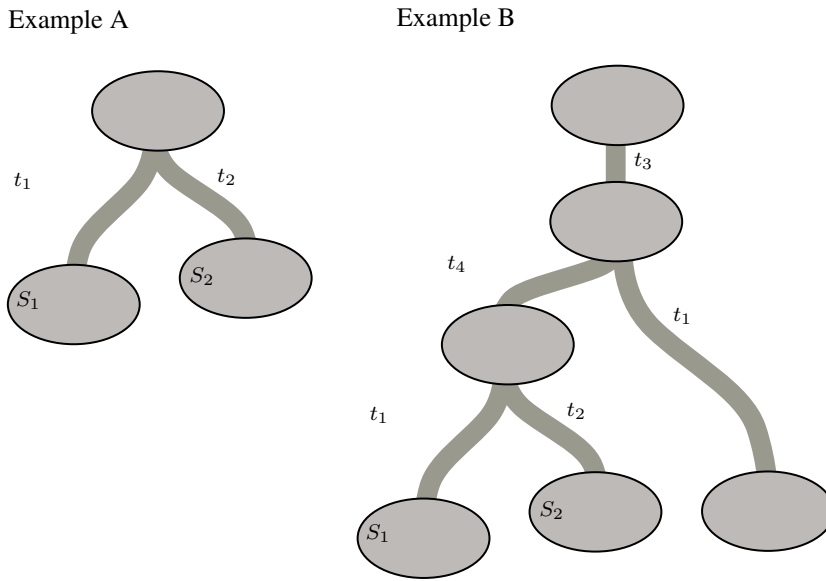


Figure 6.6: Portions of the reachability trees of the temporal nets in Figure 6.5.

case of SPN, we can get rid of α and consider only m . For all the other cases the above considerations about the enabling time of transitions are valid, no matter the way delays are modelled (using PDF or with other methods).

Arrival probabilities

In order to present some characteristics of a RT we assume in this Section to deal only with firing delays modelled as PDFs. The concepts presented here will be then extended to fuzzy numbers in Section 7.4.

Assume then to have an uncontrollable conflict, see Figure 6.7. For every state S_{C_i} , both the arrival time α_{C_i} as well as the conditioned probability P_{f_i} , representing the probability to reach S_{C_i} having reached S_f , have to be characterized. P_{f_i} represents the probability of transition t_{C_i} to win the conflict, *i.e.* fire before all the others involved, leading the system to m_{C_i} . Instead, α_{C_i} is a conditioned probability distribution, describing the possible arrival time in S_{C_i} , conditioned to the fact that S_{C_i} is reached from the root of the RT.

Nodes $enab(S_{C_{1,2,\dots}})$ must be taken into account for computing $\bar{\alpha}_{C_i}$, *i.e.* the unconditioned arrival time. $\bar{\alpha}_{C_i}$ is an arrival time which does not consider the existence of the

conflict. Assuming α_{E_i} as the arrival time in $enab(S_{C_i})$ and d_{C_i} as the firing delay of the transition leading to S_{C_i} , the following is true:

$$\bar{\alpha}_{C_i} = \alpha_{E_i} + d_{C_i} \quad (6.4)$$

Indeed, when not considering the conflict, the arrival time in a node is simply the sum of the arrival time of the furthest enabling ancestor and the firing delay of the transition leading to that node. $\bar{\alpha}_{C_i}$ is an uncertain quantity, modelled by a PDF in this case.

For computing P_{f_i} , it can be noticed that firing events in a conflict are statistically independent. Therefore, the probability of having some specific arrival times as realizations for $\bar{\alpha}_{1,\dots,u}$, as well as an arrival time in node S_f as a consequence of the preceding conflicts, is computable as a product of the probabilities of independent events:

$$\begin{aligned} \mathbb{P}(\alpha_f = a_0, \alpha_{E_1} + d_{C_1} = a_1, \dots, \alpha_{E_u} + d_{C_u} = a_u) &= \\ &= \mathbb{P}(\alpha_f = a_0) \prod_{i=1}^u \mathbb{P}(\bar{\alpha}_{C_i} = a_i) \end{aligned} \quad (6.5)$$

P_{f_1} is computed considering all the combinations of arrival times, for which $\bar{\alpha}_{C_1}$ is lower than all the others $\bar{\alpha}_{C_2,\dots,u}$ (the corresponding transition fires before all the others) and is greater than the arrival time in S_f , *i.e.* the probability of having the transition leading to S_{C_1} as the winner of the conflict. Therefore it holds that:

$$\begin{aligned} P_{f_1} &= \int_0^{+\infty} \mathbb{P}(\bar{\alpha}_{C_1} = a_1, a_1 > \alpha_f, a_1 < \bar{\alpha}_{C_2,\dots,n}) da_1 \\ &= \int_0^{+\infty} \bar{\alpha}_{C_1}(a_1) \left(\int_{-\infty}^{a_1} \alpha_f(a_0) da_0 \right) \left(\prod_{i=2}^u \int_{a_1}^{+\infty} \bar{\alpha}_{C_i}(a_i) da_i \right) da_1 \\ &= \int_0^{+\infty} \bar{\alpha}_{C_1}(a_1) \left(1 - G_f(a_1) \right) G_{C_2}(a_1) \cdots G_{C_u}(a_1) da_1 \\ &= \int_0^{+\infty} \hat{\alpha}_{C_1}(a_1) da_1 \end{aligned} \quad (6.6)$$

where in the above equation $G_{C_i}(a_1)$ refers to $\mathbb{P}(\bar{\alpha}_{C_i} > a_1)$, which is the complement of the cumulative distribution function of $\bar{\alpha}_{C_i}$ ³. Clearly, $\sum_{i=1}^u P_{f_i} = 1$.

Regarding the conditioned arrival time distribution α_{C_1} , it is equal to $\hat{\alpha}_{C_1}$ in equation (6.6), apart from a normalization coefficient:

$$\alpha_{C_1}(a_1) = \frac{\hat{\alpha}_{C_1}(a_1)}{P_{f_1}} \quad (6.7)$$

since for the Bayes law it holds that:

$$\begin{aligned} \mathbb{P}(\alpha_{C_1} = a_1) &= \mathbb{P}(\bar{\alpha}_{C_1} = a_1 | a_1 > \alpha_f, a_1 < \bar{\alpha}_{C_2,\dots,n}) \\ &= \frac{\mathbb{P}(\bar{\alpha}_{C_1} = a_1, a_1 > \alpha_f, a_1 < \bar{\alpha}_{C_2,\dots,n})}{\int_0^{+\infty} \mathbb{P}(\bar{\alpha}_{C_1} = a_1, a_1 > \alpha_f, a_1 < \bar{\alpha}_{C_2,\dots,n}) da_1} \end{aligned} \quad (6.8)$$

As can be noticed, computations in equation (6.6) are difficult to be performed in a

³ G_f is a similar complement, considering the distribution α_f

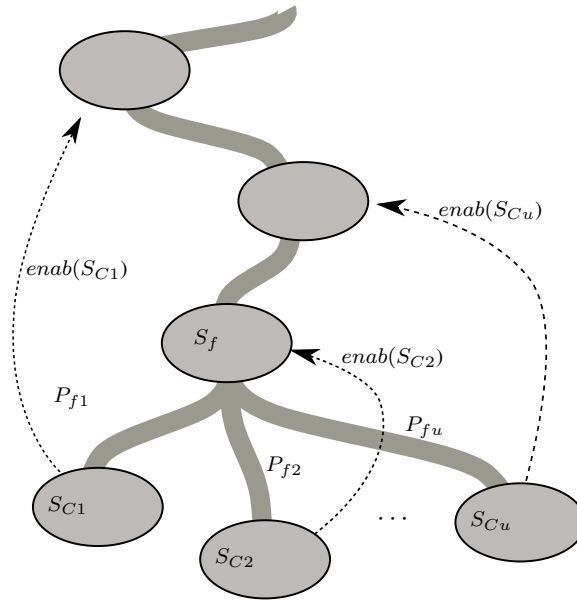


Figure 6.7: The depicted conflict is uncontrollable: all transitions leading to $S_{C1,2,\dots}$ are uncontrollable.

closed form, even when considering simple expressions for the PDFs modelling the firing times.

Similarly to the tree of Figure 5.10, the probability of reaching a leaf in the reachability tree is the product of the probabilities of winning all the conflicts in the path leading to that node. Every single winning probability is computed by making use of equation (6.6). The difference w.r.t. the tree of Figure 5.10 is that the transitions probabilities are not constant, but have to be recomputed every time, taking into account the particular trajectories leading to the nodes from which conflicts departs.

Steps involved for determining the arrival time distribution α of some nodes in the trees shown in Figure 6.6, are reported in Figure 6.8.

Exponential distributions

When considering stochastic Petri Nets, the computations discussed in the previous Section are easy to perform. The generic $d_i(t)$ of a transition in a SPN is an exponential distribution, *i.e.*:

$$d_i \sim \lambda \exp(-\lambda_i t) \tag{6.9}$$

Such distributions have the memory absence property as will be here shown. Let a distribution in a SPN begin to be enabled at a time equal to 0. Then, knowing that it hasn't fired within $[0, T]$ is not meaningful for determining the real firing time, as the

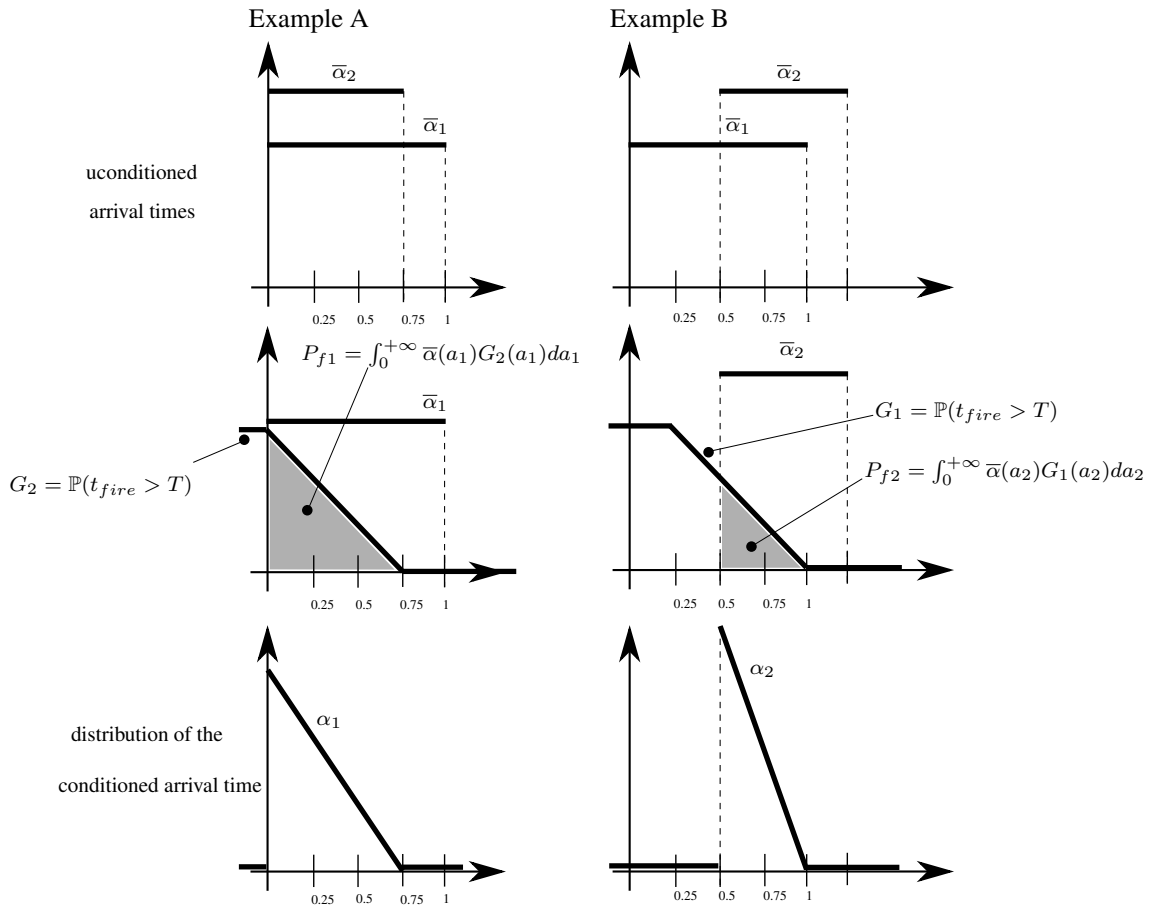


Figure 6.8: Steps involved for determining the distribution of the arrival time in S_1 of the Example A of Figure 6.6, and the same for S_2 of Example B. Notice that distributions G_f of the arrival time in the node preceding, equation (6.6), is not considered because for both the examples above, the transition leading to the node of interest begin to be enabled after arriving in S_f .

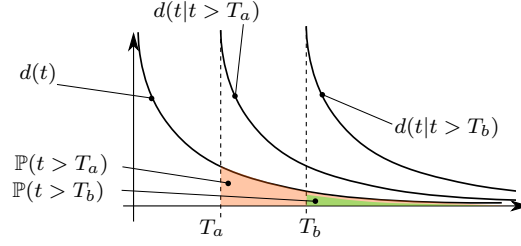


Figure 6.9: The conditioned density of an exponential distribution is in turn the same exponential distribution.

conditional distribution of the firing time is equal to the initial one. Indeed:

$$\begin{aligned}
 \mathbb{P}(t|t > T) &= \frac{\mathbb{P}(t)}{\mathbb{P}(t > T)} \\
 &= \frac{\lambda_i \exp(\lambda_i t)}{\int_T^{+\infty} \lambda_i \exp(\lambda_i t) dt} \\
 &= \frac{\lambda_i \exp(\lambda_i t)}{\exp(\lambda_i T)} \\
 &= \lambda_i \exp(\lambda_i (t - T)) \tag{6.10}
 \end{aligned}$$

Notice that equation (6.10) is identical to (6.9) setting $\hat{t}_{fire} = t - T$, *i.e.* a scaled time, see also Figure 6.10. The important consequence of what reported above, is that knowing the time at which an uncontrollable transition begins to be enabled is irrelevant. For this reason, the reachability tree of SPN is actually a finite graph, were nodes are characterized only by the state reached, no matter the trajectory leading to the nodes, as similarly happens for Markovian Processes, see Section 5.1.

When computing the transition probabilities characterizing the conflict of a set of exponentially distributed firing delays, leading to nodes $S_{C_1, \dots, u}$, we can assume all the transitions in the conflict as enabled from the time the system arrived into the state from which the conflict departs, thanks to the memory absence property, see equation (6.10). Indeed, applying equation (6.6) to these situations leads to compute the transition probability P_{f_1} in this way⁴:

$$\begin{aligned}
 P_{f_1} &= \int_0^{+\infty} \lambda_{C_1} \exp(-\lambda_{C_1} t) \left(\exp(-\lambda_{C_2} t) \exp(-\lambda_{C_u} t) \right) dt \\
 &= \lambda_{C_1} \int_0^{+\infty} \exp\left(-\sum_{i=1}^u \lambda_{C_i} t\right) dt \\
 &= \frac{\lambda_{C_1}}{\sum_{i=1}^u \lambda_{C_i}} \int_0^{+\infty} \left(\sum_{i=1}^u \lambda_{C_i} \right) \exp\left(-\sum_{i=1}^u \lambda_{C_i} t\right) dt = \frac{\lambda_{C_1}}{\sum_{i=1}^u \lambda_{C_i}} \tag{6.11}
 \end{aligned}$$

where λ_{C_i} is the parameter of the exponential distribution characterizing the transition leading to S_{C_i} .

⁴The contribution of G_f is equal to 1 for all t , since, due to the memory absence property, for all the transitions in the conflict it can be assumed they started to begin enabled after arriving in the node from which the conflict departs.

6.3 Task specification

One of the challenges in collaborative assembly operations is the maximization of the throughput, regardless of the high variability of humans in executing assembly tasks, as compared to purely automated solutions. Due to this variability, off-line scheduling and predefined assembly sequences appear to be suboptimal and unable to minimize the cycle time in a robust manner. Moreover, in realistic production scenarios, actual collaborative operations may be sporadic and interleaved with other activities performed autonomously by either the robot(s) or the human(s). Although these activities might be considered less relevant from a human-robot collaboration point of view, they should actually be regarded as equally important when modeling the system behavior. For example, if the robot has to autonomously complete an operation previously initiated by the human, an interaction between these two agents actually takes place, although the two actions are not in the same temporal span. The possibility to model and account for this kind of interaction is then crucial for the control of the overall flow of the production process. In view of this, together with the possibility for robotic non-experts to specify even complex processes from a system point of view, i.e. without focusing on the actual execution of the cycle, a systematic way to analytically describe a collaborative production process, emphasizing all types of interaction between agents, is here proposed. This approach has the aim to build in a systematic way modelling partially controllable Timed Petri Nets (pcTPNs), starting from some high level describing information, like the assembly flow for instance.

A fundamental difficulty for practitioners in modeling complex assembly sequences is represented by the modeling formalisms, which are either suitable for describing the sequence of actions, or designed for their computational capabilities, [42,75,112,115]. In the following, a simple formalism is introduced that allows to describe a certain assembly process from a high-level point of view. The properties characterizing the proposed class of nets can be analyzed by adopting formal techniques available in the literature and the reader may refer e.g. to [20] for an overview.


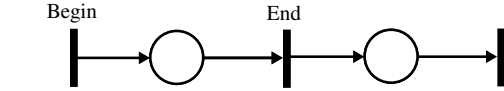
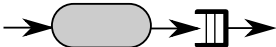
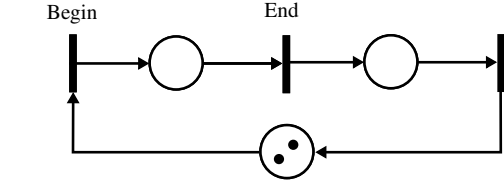
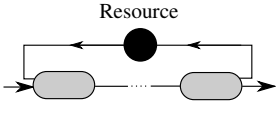
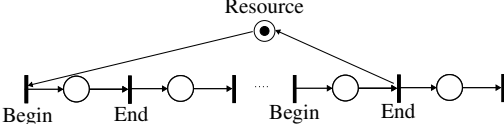

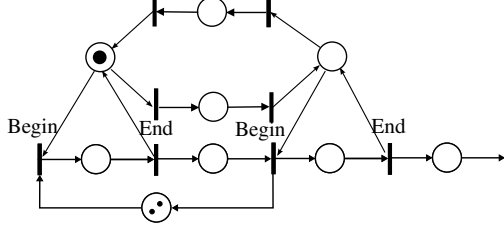
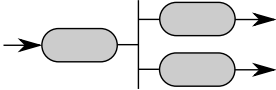
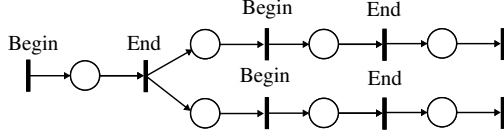
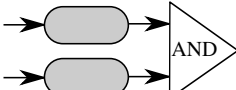
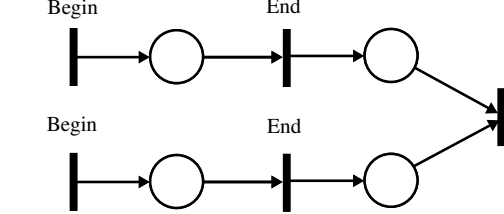
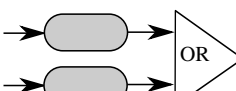
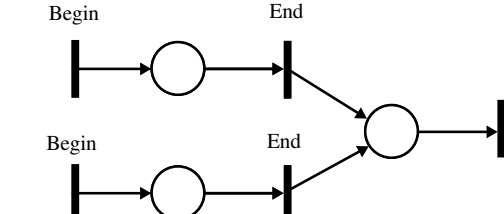
The descriptive formalism includes some basic building blocks that can be modelled as Petri Nets (see Table 6.1) and can be assembled together using standard sequential, parallel and alternative connections. Differently from the commonly adopted modeling strategies, a clear distinction between shared resources and buffers is done. In the former case, the token identifying the resource is returned when the corresponding action is terminated, whereas in the latter the token identifying an occupied spot in the buffer is returned only when the subsequent action starts, taking the corresponding part from the buffer.

In order to be executed, the actions in Tab.6.1 require one or more agents. When dealing with human-robot collaboration in assembly processes, the agents considered herein are the human and the robot. The main difference to deal with is that robots are completely controllable, while humans are only partially controllable and non deterministic.

Controllable transitions are such that their firing can be prevented by a supervisor, whereas uncontrollable transitions cannot be prevented from firing, see Section 6.2. An ad hoc formalism is now introduced to take into account the human behaviour. In the following, controllable transitions may be graphically represented with thick bars.

Chapter 6. Scheduling of the robotic actions

Table 6.1: Descriptive formalism adopted in this work and equivalent PN blocks for automatic translation.

Name	Notation	Petri Net
Action		
Buffered action		
Use of a resource (e.g. a tool or space)		
Load, transport and unload (e.g. with a mobile robot)		
FORK		
AND		
OR		

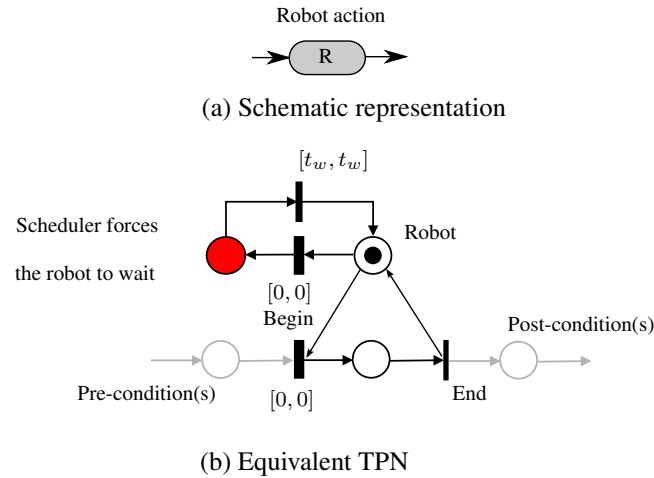


Figure 6.10: *PN model for actions executed by the robot.*

Since it was assumed, differently from the usual practice with TPNs, [7], that controllable transitions may fire only instantaneously after the enabling event, the act of waiting has to be modeled explicitly as an alternative action that can be scheduled, leaving it to the scheduler to resolve the possible generated conflicts (if multiple conflicting controllable transitions are simultaneously enabled, only one of them fires). This is made possible by explicitly defining the waiting of a predefined quantum of time t_w as an action. This working hypothesis simplifies the control architecture, in that the scheduler simply decides which of the enabled controllable transitions to fire at each iteration. To make this decision it can employ a version of the reachability tree of the TPN that avoids propagating parametric time-related information, as opposed to more sophisticated concepts such as the State Class Graph, [4]. Some modeling rules are now suggested, discriminating between actions executed by the human and the robot.

6.3.1 Modeling robot actions

The availability of the robot to perform an action is represented by a marked place, as a standard resource. The beginning of a certain action performed by the robot corresponds to the firing of a controllable transition that consumes the corresponding robot token, which is later returned when the action itself is over (the action termination being modeled as an uncontrollable transition). An alternative controllable transition can consume the token, marking a special place which explicitly represents the robot being idle (waiting action). An uncontrollable transition which fires after a predefined quantum of time allows the token to return to its original place. Figure 6.10 illustrates the PN adopted to model the execution of a certain action by the robot. The robot has to decide whether to wait, marking the red place, or to start the action. Notice that the decision to wait will be accounted for as a cost, and the total amount of inactivity will be minimized by the scheduler, so to maximize the throughput.

6.3.2 Modeling human actions

In order to allow non-expert users to model a potentially complex behavior, the human is modeled only in terms of actions that are relevant from the point of view of the robot

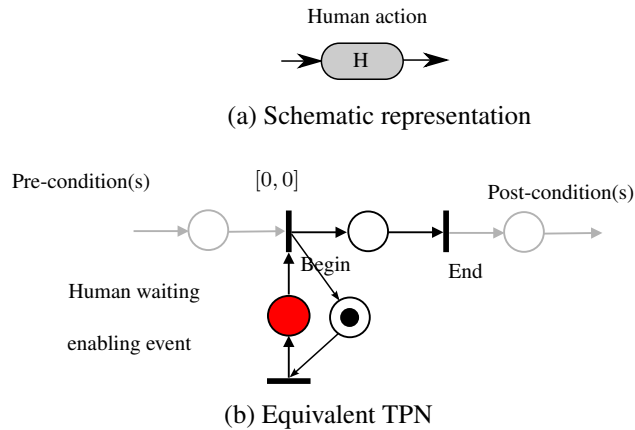


Figure 6.11: PN model for actions executed by the human.

and hence to the overall process. For example, it is not necessary to model whether the action of dropping a piece in a buffer is performed by the human with the right or the left hand, but rather its net effect on the process, i.e. that a new piece is available to be processed by the robot.

The human, differently from the robot, is not assumed to be a controllable agent. Accordingly, the intention to start an action assigned to the human is modeled with an uncontrollable transition followed by a special place which represents the human waiting for some event to be verified in order to actually proceed with the intended action. In case this pre-condition is satisfied, the corresponding transition fires immediately. Conversely, if the pre-condition is not met, the special place will remain marked, causing the human to remain idle. Figure 6.11 illustrates the PN adopted to model the execution of a certain action by the human operator.

6.3.3 Modelling collaborative actions

A collaborative action is performed simultaneously by the robot and the human. The PN describing the corresponding behavior can be simply obtained by superposition of the robot action and the human action, as described previously, see Figure 6.12. Notice that the robot is responsible for initiating the actual collaborative task, while both the agents are allowed to wait when the two red places are marked.

6.3.4 Modelling mobile robots

A common action in assembly processes is the transport of items using a mobile robot. This action is typically composed of two atomic and subsequent activities: loading and unloading. As an example, the PN reported in Figure 6.13 details the example of a manual loading action and an unloading activity performed by a robot. Notice that, while the inactivity of either the human or the robot causes a cost, the inactivity of a mobile robot, which is only responsible for transporting items, does not directly represent a cost. However, if the unavailability of the mobile robot at the unload station forces the robot to wait, a cost for the inactivity of the robot is accounted for.

A pcTPN, compliant with the building rules described above, is the one represented

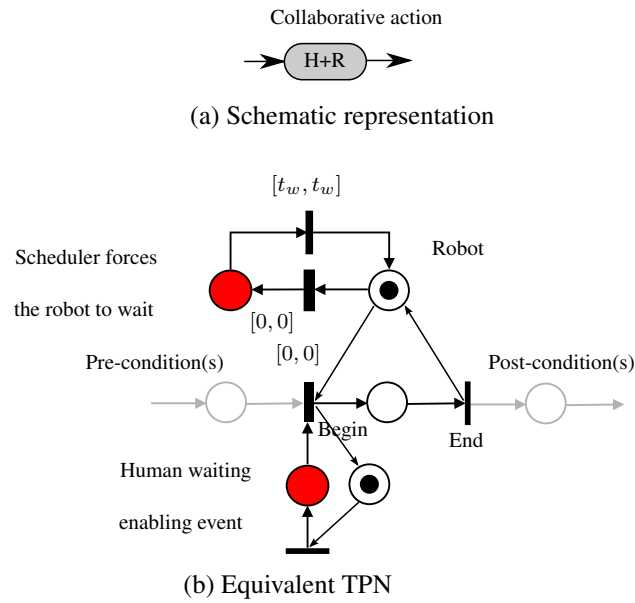


Figure 6.12: PN model for collaborative actions.

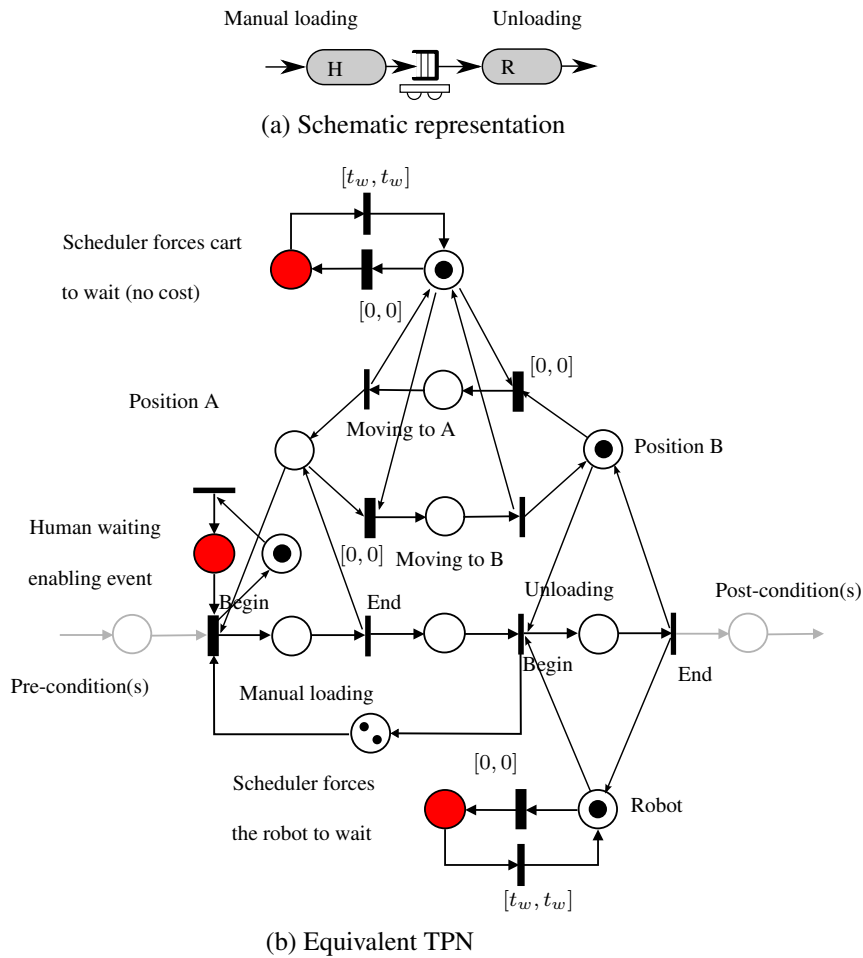


Figure 6.13: PN model for a mobile robot (cart) that transports items that are loaded manually and unloaded using a robot.

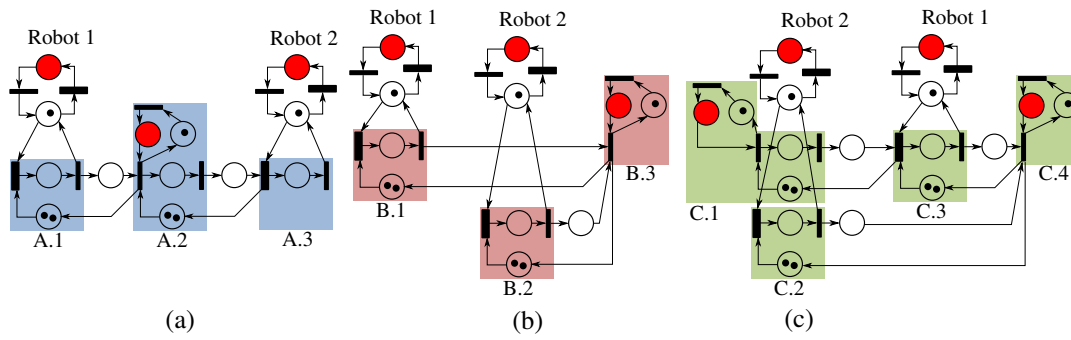


Figure 6.14: By superimposing the above nets, the pcTPN modelling the assembly in Figure 6.2 is obtained.

in Figure 6.14, which models the assembly described in Figures 6.2 and 6.3.

CHAPTER 7

Scheduling approaches

Three possible scheduling approaches for assistive scheduling will be presented in this Section. All the proposed strategies are based on a receding horizon approach, which is extensively described in Section 7.1. Therefore, in the subsequent Sections, it will be specified only how the reachability tree of the system is computed and the criteria adopted for selecting the optimal plan. All the proposed approaches are validated in realistic situations for which a cobot (or more than one) cooperate with a human for performing the assembly of some industrial products.

In particular, Section 7.2 will present a best case approach. The approaches in Section 7.3 and 7.4 extend the one in Section 7.2, balancing, when computing the optimal plan to be imposed to the system, the achievable performance with the risk of a plan to become infeasible due to the uncertainties characterizing the system. In particular, the method in Section 7.3 is a numerical method, based upon a Monte Carlo simulation, while the one in Section 7.4 is based on fuzzy theory.

7.1 Receding horizon scheduling

In this Section, the general approach followed by the three methods proposed in Sections 7.2, 7.3 and 7.4, for controlling a collaborative cell will be discussed.

The scheduling problem ultimately consists in deciding which activity (also including a decision to wait, see Section 6.3.1) to perform for the robots of the cell. Since all controllable transitions are assumed to fire immediately when enabled, the scheduling algorithm boils down to the problem of deciding which transitions to fire in case of controllable conflicts. Accordingly, the output of the scheduler consists in the list of the controllable transitions to fire: actually the corresponding commands are sent to robots.

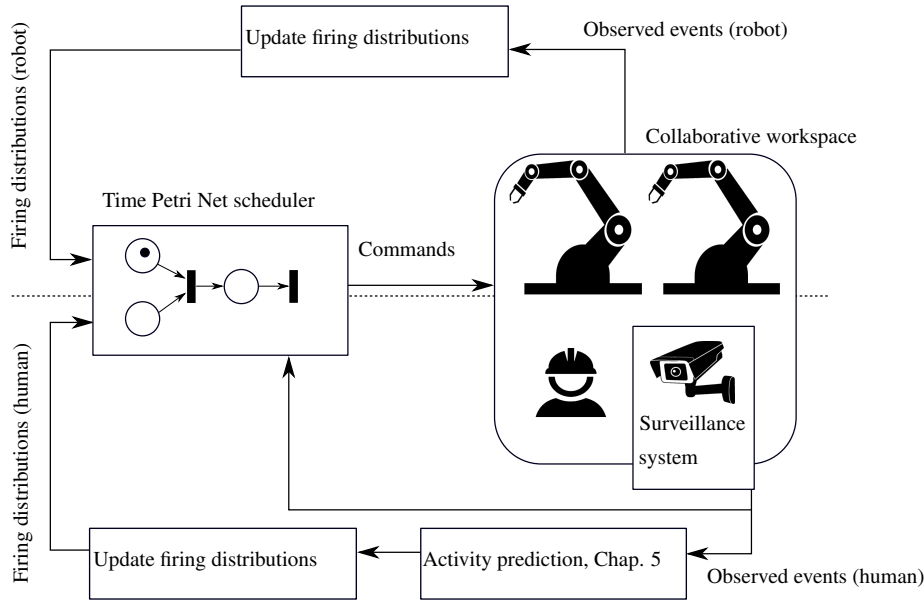


Figure 7.1: Schematic representation of the control architecture adopted for scheduling collaborative cells.

The control architecture is sketched in Figure 7.1. The main element is the scheduler, which interacts with the real robotic cell through events of two types: outgoing commands (associated to the firing of controllable transitions) issued to the robots, and incoming events (associated to the firing of uncontrollable transitions), reporting task terminations. The information regarding the occurrence of events allows the scheduler to update the state of the TPN that models the system. At the same time, the system records the duration of the tasks executed by the robots, using this information to update d , *i.e.* the corresponding distributions (a PDF or a fuzzy quantity) that model the timing behaviour of the uncontrollable transitions. Actually this is exactly the approach followed by both the methods described in Sections 7.3 and 7.4, which assume a poor prior knowledge for d : the samples collected during time allow the system to refine the distribution modelling d . On the opposite, the approach in Section 7.2 is similar for all other aspects, but assumes to know in a precise way the support of d prior to the execution of the scheduler. This is because in 7.2 is not necessary to characterize the entire distribution of d , but only the domain extremals, which can be easily determined according to few off line measured times, also adopting conservative values. Anyway, the approach described in Section 7.2 is amenable to an on line correction of the quantity describing d .

The human actions are similarly monitored, using a surveillance system to detect events. The timing of these events is also collected to build and update the statistical model employed by the predictive algorithm, Chapter 5, to forecast the human intentions. In turn, these predictions are used to update the firing distributions of the uncontrollable transitions associated to the human tasks.

The scheduler is run in an asynchronous way and specifically when one or more controllable agents become available, upon the firing of an uncontrollable transition. The ultimate goal of any kind of scheduling algorithm is to minimize the time wasted by

agents, consequently maximising the throughput. To this aim, the reachability tree of the underlying TPN is built with the aim of comparing alternatives plans, since a plan can be seen as a path in the reachability tree, made of a sequence of transitions to fire: some of them are controllable, some other uncontrollable. Every particular scheduling strategy is characterized by its own approach adopted for building the RT as well as the criteria adopted for selecting the optimal plan in the tree. Then, what is common to the three approaches in Sections 7.2, 7.3 and 7.4, is that a receding horizon approach is adopted: every time a new plan is computed, only the first controllable transitions in such plan are applied as commands to the system. Then, the system evolves randomly till other agents become available. In this case, a new plan is computed and commands are dispatched to agents.

Since a reachability tree is in principle infinitely large, a finite scheduling horizon T_{sch} must be adopted: every time only a portion of the reachability tree is built, starting from a root representing the current state of the system. The selection of the proper scheduling horizon is crucial. Clearly, T_{sch} must be big enough such that at least one uncontrollable event happens, in order to evaluate the consequences of the applied controls. Therefore, the selection of T_{sch} must be done considering also the supports of d , *i.e.* the firing delays. On the opposite, the size of the reachability tree may grow exponentially with T_{sch} , affecting severely the computational times. For these reasons, T_{sch} must be tuned decided according to the particular application for which the scheduling is required.

Notice that the developed framework generalizes the control approach adopted in Generalized Stochastic Petri Nets (GSPNs) [27], a variant of pure SPN having controllable transitions firing instantaneously. As clarified in Section 6.2.1, the equations governing the evolution of a SPN are the same of a Markovian process. On the other hand, GSPNs are analogous to Markov supervisors [6]: the optimal control policy is computed off line with algorithms like value iterations. This off line solution is simply applied on line for solving controllable conflicts. This is possible because GSPN contains uncontrollable transitions all exponentially distributed. The price to pay when dealing with generic distributed transitions, which is the aim of the proposed scheduling strategies, is to re-evaluate on line the possible future evolution of the system, deciding every time which is the best action to undertake.

7.2 Best scenario approach

Prior to describe the approach, the use case adopted in Section 7.5.1 for validating it will be presented, thus allowing the reader to better follow the subsequent technical part.

7.2.1 Description of the use case

The workspace is shared by one human operator and a dual-arm robot, YUMi of ABB, which actively cooperate to perform the assembly of a PCB board (named board A) to be inserted in an IP 54 plastic enclosure. A second PCB board (board B) is fixed to the cap of the enclosure. It is assumed that all items required for the realisation of the assembly are externally fed at the proper time. The following operations are executed

to obtain one finished product (refer also to assembly flow reported in Figure 7.2):

1. fix one fuse to board A;
2. fix one integrated circuit to the same board;
3. perform a quality control to check the correctness of the previous operations;
4. put one assembled board A into the plastic enclosure;
5. pick one cap and connect with a cable board A to board B;
6. fix the cap on the plastic enclosure.

Steps 4), 5) and 6) are performed within a single collaborative macro-operation involving the human operator and both robotic arms. As for the tasks executed autonomously, operation 2) is assigned to the human, while 1) is executed by the left arm of YUMI. Finally, operation 3) requires both robotic arms to be used. The right arm of YUMI has no autonomous tasks assigned.

The time when the human is likely to start every assigned task (either autonomous or collaborative) can be forecasted using one of the algorithms in Chapter 5. Such algorithms are able to characterize the distribution of the waiting time for seeing again a certain human action, see Section 5.3. However, the approach presented in this Section only needs to bound it within two percentiles (e.g. the 10th and the 80th).

Every time the left arm of YUMI becomes available, it is instructed to perform a new operation (actions in cooperation with the right arm can be performed only if the right arm is also available). To select the proper operation to execute, the estimates of some P^{th} percentiles of the waiting time to see again certain human activities are taken into account: P_{a1} and P_{a2} will denote the percentiles about operation 2), while P_{c1} and P_{c2} refer to the percentiles of the collaborative task. The designed control policy takes into account three possible cases:

1. $P_{a1} < P_{c1}$ and operation 2) is not enabled at the current time. In such case, a command to execute operation 1) is sent to the left arm of YUMI. In this way, operation 2) is enabled as soon as possible;
2. $P_{a1} < P_{c1}$ and operation 2) is enabled at the current time. In this case a non trivial scheduling of operations is required to select the proper command to be sent to the robot;
3. $P_{c1} < P_{a1}$. As for case (2), a non trivial scheduling of operations is required.

In cases (2) and (3), the scheduling algorithm detailed in Section 7.2.2 is invoked to compute the optimal sequence of robotic actions to undertake before the collaboration can take place, with the aim of maximising the throughput, while at the same time avoiding a prolonged inactivity of the human. In case the first action scheduled is the collaborative task, the robots wait for the human to begin the collaboration, without starting to execute other tasks. This idle status persists for a maximum waiting time (further detail will be given in Section 3), after which the robots are available again.

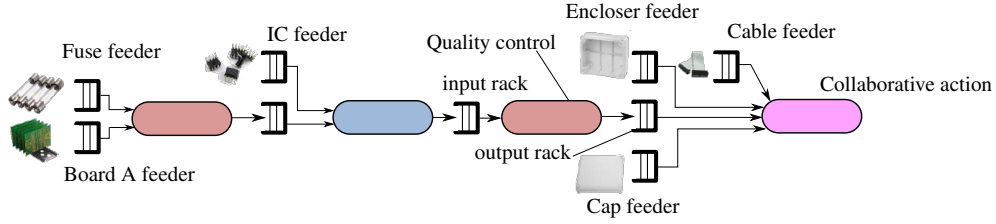


Figure 7.2: Sequence of operations required to obtain a finished product for the use case adopted to validate the best scenario approach scheduling. Red boxes refer to operations assigned to YUMI, while the blue one is the autonomous task of the human. The pink box indicates the collaborative action, executed simultaneously by both the human and the arms of YUMI. Feeders contain an infinite number of items, since they are externally fed when needed.

7.2.2 Selection of the best plan

The pcTPN of Figure 7.3 is assumed as a model for the system. Uncontrollable transitions fire at a random time, having PDF whose shape is not modelled for the aim of this work. Indeed, for every firing delay distribution d , only the corresponding support $[d, \bar{d}]$ is taken into account. The same applies for the arrival times into nodes of the reachability tree, which is within an interval $[\underline{\alpha}, \bar{\alpha}]$.

The net in Figure 7.3, was not built according to the guidelines detailed in Section 6.3, since for the aim of this kind of scheduling, the only inactivity time to evaluate is the waiting time before initiating the collaborative action. In particular, the firing of \bar{t}_c indicates that the robots are ready to begin a new collaboration, but the collaborative operation actually takes place only after the human is also ready, *i.e.* after firing t_{ch} . In this way, when the robots receive the command to begin the collaboration, they are no longer available and start waiting for the human to be ready. However, after transition t_c is fired, the robots involved in the collaboration remain in an idle state until a maximum waiting time, after which the collaboration command is revoked and consequently the robots return available. To model this behaviour, transition t_W is introduced. t_W begins to be enabled after the firing of t_c and has a firing delay equal to T_{wait} , where T_{wait} is the maximal time for which robots can remain inactive, waiting for the human to initiate the collaborative task.

Notice also that in case the collaboration takes place, Figure 7.3, left arm of YUMI finishes its subtask before the right one. In this way, the left arm can potentially start to execute a new operation, while the right one is still involved in the collaboration.

Scheduling aims at finding an optimal plan, consistent with the interval $[P_{c1}, P_{c2}]$. If the collaboration is constrained to start at a time within $[P_{c1}, P_{c2}]$, the involved robots must be ready before P_{c1} , which implies that t_c should be fired prior to P_{c1} . This worst case approach may turn out to be extremely conservative. Instead, the predicted interval $[P_{c1}, P_{c2}]$ are used only as a reference to evaluate the inactive time spent by the robots or by the human waiting for the collaboration to take place. The scheduling problem is solved under the following assumption. Since the robots are the sole controllable agents, when computing the optimal plan it is assumed that transition t_h is not allowed to fire (*i.e.* it has a firing delay d_h equal $+\infty$). In this way, the operations that the human executes simultaneously with the robots before the collaborative one are not taken into account.

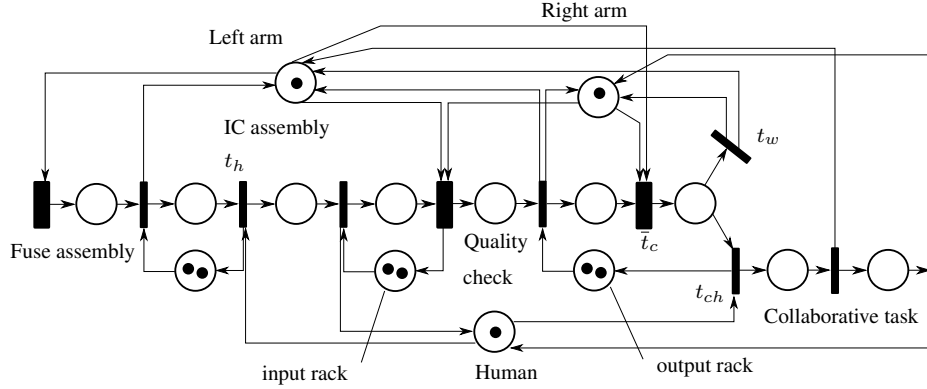


Figure 7.3: The *pcTPN* modelling the use case. Thick rectangles denote controllable transitions. Transitions \bar{t}_c and t_{ch} refer to the collaborative operation. As can be seen, the limited capacity of intermediate buffers is taken into account.

Branch and bound strategy

A feasible plan v is a path in the RT described by a firing order: $v = t_{v1}; t_{v2}; \dots; t_c$, representing an evolution of the net which leads from m_0 (the marking of the root) to a final state ¹, reached by firing as a last transition t_c . Notice that the sequence represented by v is uniquely associated to the node reached at the end of that sequence of events. Therefore, we'll refer equivalently to plans and nodes in the RT. Among all feasible plans, the aim is to find v^* , which is the optimal one (in a sense that will be formalized in the following). Set \mathcal{F} contains all the nodes S_f at the end of the feasible plans computable in a RT. A cost C can be assigned to every v , whose computation is now discussed. Assuming that the exact firing time of all the uncontrollable transitions present in the path leading to a node $S_f \in \mathcal{F}$ is known, it could be possible to exactly compute the value of α_f , which is the arrival time in S_f and, in this case also represents the time at which the robots are ready to begin a collaboration with the human. Then, knowing the exact value of α_f , the cost C_f can be computed. C_f is proportional to the forecasted waiting time spent by either the robots or the human, before the collaboration actually takes place:

$$\text{cost}(\alpha_f) = \begin{cases} c_R(P_{c1} - \alpha_f) & \text{when } \alpha_f < P_{c1} \\ 0 & \text{when } P_{c1} \leq \alpha_f \leq P_{c2} \\ c_H(\alpha_f - P_{c2}) & \text{when } \alpha_f > P_{c2} \end{cases} \quad (7.1)$$

The above conditions are summarised by the cost curve indicated in Figure 7.4. Then to account for the fact that the exact value of α_f is uncertain, the support of the probability distribution of the arrival time $[\underline{\alpha}_f, \bar{\alpha}_f]$ is considered: the integral mean of function $\text{cost}(\alpha_f)$ over that support is assumed as cost C :

$$C = \frac{1}{\bar{\alpha}_f - \underline{\alpha}_f} \int_{\underline{\alpha}_f}^{\bar{\alpha}_f} \text{cost}(\alpha_f) d\alpha_f \quad (7.2)$$

v^* is the plan having the lowest possible C among all feasible plans.

¹The extension in time of the RT is infinite. However, this approach will not explore the RT further to encounter the collaborative action, see also Algorithm 2

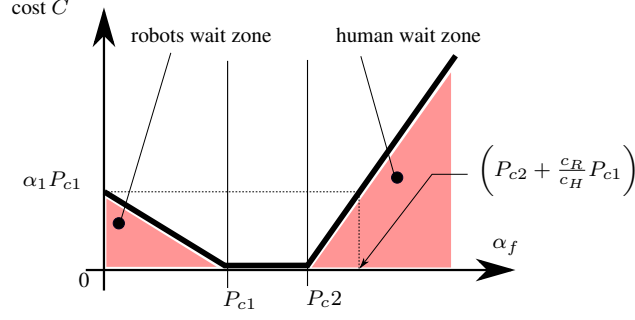


Figure 7.4: Cost curve for a feasible path, in case the arrival time s_f is known. c_H can be interpreted as the unit cost, for example $\$/s$, paid when the human is kept inactive (similarly c_R when maintaining the robots inactive).

The developed algorithm constructs the RT in a depth-first way with the aim to find the optimal plan v^* . At every iteration a new node is created and processed by computing the corresponding marking and arrival set. For every branch, the exploration is arrested when \bar{t}_c is met or if it is evident that further exploring will lead only to find suboptimal feasible paths, *i.e.* feasible nodes with a cost higher than the minimum cost related to feasible nodes already explored.

To this purpose, for a generic node $S_i \notin \mathcal{F}$, cost \hat{C}_i is computed. \hat{C}_i is a lower bound for the cost C_j associated to a generic feasible node $S_j \in \mathcal{F}$, reached by further exploring the RT from N_i , *i.e.* $\hat{C}_i \leq C_j$.

Before explaining the computation of \hat{C}_i , one key observation must be done. If S_j is a successor of S_i , then it holds that $\underline{\alpha}_i \leq \underline{\alpha}_j$, which means that the lower bound of the minimum possible arrival time monotonically increases when descending levels of the RT. The same consideration is not valid for the upper bounds $\bar{\alpha}_i, \bar{\alpha}_j$.

For this reason, when considering the cost curve shown in Figure 7.4, it is not difficult to derive that \hat{C}_i can be computed as $\hat{C}_i = 0$, if $\underline{\alpha}_i \leq P_{c2}$, and $C_i = (\underline{\alpha}_i - P_{c2})c_H$, otherwise. The complete scheduling algorithm is reported in Algorithm 2. It starts by initialising the RT with the root S_0 . Then, for every iteration, a new node is selected and all of the possible successors are computed. Algorithm 2 employs the procedure described in Algorithm 3 when processing new nodes. Algorithm 2 can be seen as a branch and bound strategy to compute the optimal plan v^* .

It is worth pointing out that Algorithm 2 returns a single optimal path v^* . However, many different paths having a cost equal to C^* can be admissible. To overcome this problem, it is not difficult to slightly modify Algorithm 2 to return a list of paths $\{v_1^*, \dots, v_L^*\}$ having the same cost C^* . Then it is possible to select from this list one single v_i^* .

The selection can be made according to different possible criteria, such as: the number of non collaborative operations performed by the robots before reaching \bar{t}_c , the minimisation of time at which one new scheduling operation will be required² to maximise responsiveness, etc.

The first controllable transitions scheduled to fire in v^* are the actions that the robots

²This implies to minimise the time required to execute the first action contained in v^* .

Chapter 7. Scheduling approaches

must begin to execute from the current time. If the first transition in v^* is \bar{t}_c , the robots wait for the human to start a new collaboration for the maximum possible waiting time.

Algorithm 2 Scheduling algorithm

```
 $v^* = \text{undefined}$  and  $C^* = \text{undefined}$ 
 $RT \leftarrow S_0$ 
tag  $S_0$  as unexplored
while at least one node in  $RT$  is tagged unexplored do
  pick an unexplored node  $S_i \in RT$ 
  for every  $t_k \in T$  enabled in  $S_i$  do
    Compute a new node  $S_k$  by using Algorithm 3
    if arrival set of  $S_k$  is tagged as possible then
      if  $v^*$  is undefined then
         $RT \leftarrow RT \cup S_k$ 
        if  $t_k = t_c$  then
           $v^* = S_k$ 
          compute  $C_k$ 
          tag  $S_k$  as explored
           $C^* = C_k$ 
        else
          if  $t_k = t_c$  then
            compute  $C_k$ 
            if  $C_k < C^*$  then
               $v^* = S_k$ 
               $C^* = C_k$ 
               $RT \leftarrow RT \cup S_k$ 
              tag  $S_k$  as explored
            else
              compute  $\hat{C}_k$ 
              if  $\hat{C}_k < C^*$  then
                 $RT \leftarrow RT \cup S_k$ 
          tag  $S_i$  as explored
return  $v^*$ 
```

7.2.3 Remarks

The approach presented is a best scenario one. Indeed, the partial controllability of the system implies that in a plan v there will be always controllable transitions alternating with uncontrollable ones. For this reason, when selecting a certain plan, only the initial controllable transitions characterizing that plan can be actually fired. Indeed, it is not ensured that after the first uncontrollable transition the system will follow the optimal plan. This fact happens because this approach tries to maximize the performance, not considering at all the probability to get at the end of a scheduled plan or more in general not considering the alternative evolutions of the system after imposing the first package of controllable actions. On the opposite, the approaches presented in Section 7.3 and 7.4 will perform the scheduling by considering all the possible evolutions of the system, not only the best one.

Anyway, it is worth to point out that the receding horizon approach (see Section 7.1) provides anyway some kind of robustness to the approach. In fact, no matter that the system evolves or not differently from the scheduled expected way, a new plan is always

Algorithm 3 New node computation. Inputs: N_i , preceding t_{ik}

```

 $m_k \leftarrow (Post - Pre)_{ik} + m_i$ 
find  $S_e = enab(S_k)$ 
if  $t_{ik} \in T_c$  then
     $\underline{\alpha}_k \leftarrow \underline{\alpha}_e$ 
     $\overline{\alpha}_k \leftarrow \overline{\alpha}_e$ 
else
     $\underline{\alpha}_k \leftarrow \underline{\alpha}_e + \underline{d}_{ik}$ 
     $\overline{\alpha}_k \leftarrow \overline{\alpha}_e + \overline{d}_{ik}$ 
if  $\overline{\alpha}_k < \underline{\alpha}_k$  then
    tag arrival set of  $S_k$  as impossible
else
    tag arrival set of  $S_k$  as possible
if  $\underline{\alpha}_k > \underline{\alpha}_i$  then  $\underline{\alpha}_i \leftarrow \underline{\alpha}_k$ 

```

recomputed by considering the reached state.

7.3 Monte Carlo scheduling

7.3.1 Selection of the best plan

The aim of the scheduling proposed here is to minimize the global inactivity times of both the human and the robots. In particular, according to the modelling principles explained in Section 6.3, the idle time can be easily evaluated considering the amount of time spent by the tokens in the colored places of the net. Therefore, the overall cost C of a plan can be computed by summing the time spent waiting by the robot(s) $W_{R,i}$ and by the human W_H , *i.e.* $C = c_R \sum_i W_{R,i} + c_H W_H$, where coefficients c_R and c_H can be used to assign a different cost to the two types of agents (as similarly done in the approach of the previous Section). It is important to remark that, since the TPN evolves in a non deterministic manner, the total waiting time C is a random variable itself.

Every time one or more agents become free, new actions to do are computed. The scheduling proposed in [19] consists in two main stages:

1. generation of a partial state reachability tree and evaluation of the costs through a Monte Carlo simulation;
2. propagation of the costs and evaluation of the optimal policy;

which are further detailed in the following. The firing distribution d of the net are assumed to be modelled by the empirical distributions that consider the samples of durations collected from the real system (see Section 7.1).

Generation of the state reachability tree and cost evaluation

A partial reachability tree is constructed by running N Monte Carlo independent simulations starting from the current marking of the net. Each trial of the Monte Carlo is a possible trajectory of the system, *i.e.* a sequence of firing events (with their associated times). If an uncontrollable transition is fired in the simulation, its duration is sampled from previous observations. If multiple conflicting uncontrollable transitions are enabled, only the transition that is evaluated to fire first (according to the extracted

duration samples for the unconditioned arrival times $\bar{\alpha}$ in the children reachable after the conflict, Section 6.2.1) is actually fired. On the other hand, when the simulation reaches a marking where multiple conflicting controllable transitions are enabled, the transition that fires is extracted randomly, according to a uniform distribution. Each simulation is protracted until time T_{sch} is met, implying that events occurring after this time horizon are not considered. Notice that all simulations start with a controllable transition, since the scheduling algorithm is invoked when (at least) one agent becomes available.

When the tree construction is completed, aggregating the samples associated to the same leaf, see Figure 7.5, one can estimate the cost of a certain evolution of the system. Notice that the percentage of simulations reaching a particular node S_i is an estimate of the arrival probability in that node, while samples accumulated as possible arrival times, are used to empirically describe the conditioned arrival time distribution α_i .

Cost propagation and evaluation of the optimal policy

To determine which of the (controllable) transitions enabled in the current state, *i.e.* at the root node of the tree, it is best to fire at the present time instant, the expected costs associated to the firing of each of them must be evaluated. To this aim, the probability distributions of the costs associated to the leaf nodes are back-propagated to the first children nodes of the tree (nodes immediately following the root). In this back-propagation process, the following rules must be applied if multiple nodes have the same father node. Precisely:

- a) if this branching follows from an uncontrollable conflict, the probability distribution of the cost of the father node is obtained by combining all the children's distributions (all the possible evolutions of the systems must be taken into account);
- b) if the branching results from a controllable conflict, the cost is inherited from the child node having the cost distribution with the lowest β -percentile (indeed, one would choose to fire the controllable transition leading to the smallest cost, at least in a probabilistic sense);

Notice that this back-propagation process can be effectively implemented through a simple recursive function, which calculates the cost distribution of a father node from those of its children, which in turn are calculated based on their own children, and so on, unless the children nodes are leaves. Algorithm 4 contains the pseudocode implementing the reduction strategy. The algorithm can be called recursively from the root node.

Figure 7.5 summarizes the above propagation rules.

An illustrative example

Consider the assembly process illustrated in Fig. 7.6 and modeled by the pcTPN of the same Figure: three operations are performed by the robot (R), the human (H), and in a collaborative manner (H+R), respectively. Notice that every time the robot is available to initiate an activity, a conflict arises. The scheduler will be responsible for deciding which controllable transition to fire among those conflicting. In the example the options

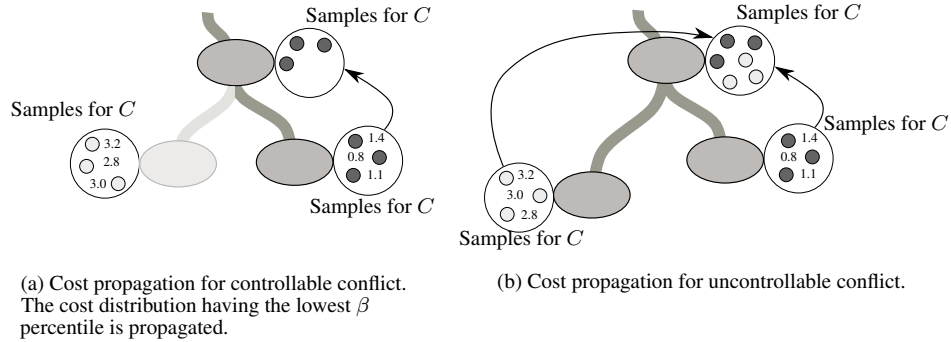


Figure 7.5: The two rules adopted for back propagating particles of cost in a reachability tree. When considering the controllable conflict on the left, the edge leading to the node having the lowest β percentile of cost will be followed in case the system would arrive in the indicated node. Therefore, the other edge is in some way disabled by the optimal control policy.

Algorithm 4 Reachability Tree Reduction

```

1: procedure EVALUATECOSTS(node  $S$ , percentile  $\beta$ )
2:   if ISROOT( $S$ ) then
3:      $B \leftarrow +\infty$ ;
4:      $S.Costs \leftarrow \emptyset$ ;
5:     for all child node  $c$  of  $S$  do
6:        $cTemp \leftarrow$  EVALUATECOSTS( $c$ ,  $\beta$ );
7:        $bTemp \leftarrow$  GETPERCENTILE( $cTemp$ ,  $\beta$ );
8:       if  $bTemp < B$  then
9:          $B \leftarrow bTemp$ ;
10:         $S.Costs \leftarrow cTemp$ ;
11:     return  $S.Costs$ ;
12:   else if ISLEAF( $S$ ) then
13:     return  $S.Costs$ ;
14:   else
15:     if ISCONTROLLABLE( $S$ ) then
16:        $B \leftarrow +\infty$ ;
17:        $S.Costs \leftarrow \emptyset$ ;
18:       for all child node  $c$  of  $S$  do
19:          $cTemp \leftarrow$  EVALUATECOSTS( $c$ ,  $\beta$ );
20:          $bTemp \leftarrow$  GETPERCENTILE( $cTemp$ ,  $\beta$ );
21:         if  $bTemp < B$  then
22:            $B \leftarrow bTemp$ ;
23:            $S.Costs \leftarrow cTemp$ ;
24:       return  $S.Costs$ ;
25:     else
26:        $S.Costs \leftarrow \emptyset$ ;
27:       for all child node  $c$  of  $S$  do
28:          $cTemp \leftarrow$  EVALUATECOSTS( $c$ ,  $\beta$ );
29:          $S.Costs \leftarrow S.Costs \cup cTemp$ ;
30:     return  $S.Costs$ ;

```

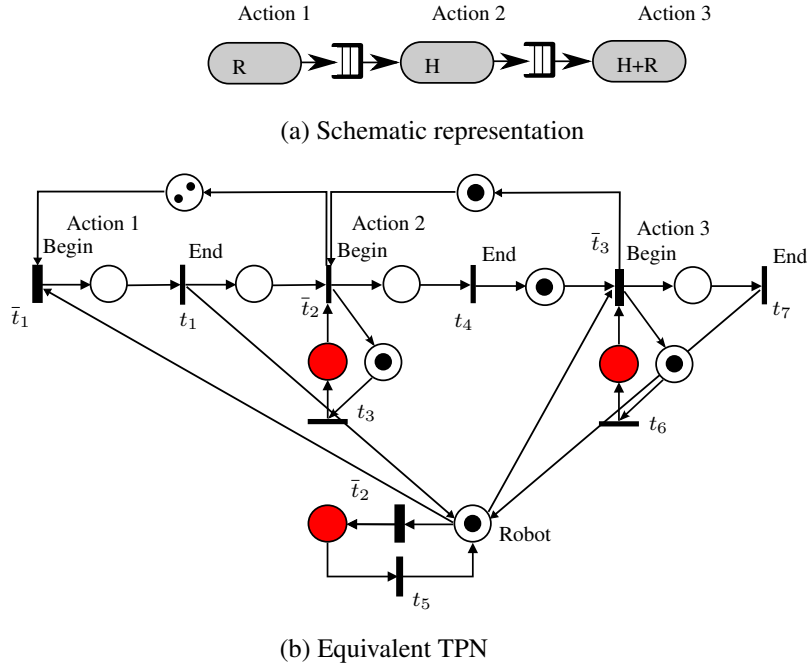


Figure 7.6: Example of a PN compliant with the proposed modeling strategy.

Table 7.1: Timing parameters of the transitions of the TPN of Fig. 7.6.

Transition	Type	d
\bar{t}_1	controllable	$U[0, 0]$
\bar{t}_2	controllable	$U[0, 0]$
\bar{t}_3	controllable	$U[0, 0]$
τ_1	uncontrollable	$U[3, 3]$
τ_2	uncontrollable	$U[0, 0]$
τ_3	uncontrollable	$U[2, 4]$
τ_4	uncontrollable	$U[1, 1]$
τ_5	uncontrollable	$U[1, 1]$
τ_6	uncontrollable	$U[2, 5]$
τ_7	uncontrollable	$U[5, 7]$

are \bar{t}_1 (performing Action 1), \bar{t}_3 (performing Action 3) and \bar{t}_2 (remaining idle).

The timing parameters of the net's transitions are reported in Tab. 7.1. Notice that all controllable transitions have a firing time interval equal to $[0, 0]$, implying that they must fire immediately upon being enabled. As for the uncontrollable transitions, if the firing interval does not coincide with a single point, the corresponding type of distribution is also reported.

The first steps of the Monte Carlo simulation leading to the determination of the reachability tree for the example of Figure 7.6 will be now reported. A scheduling horizon T_{sch} of 4 s is assumed. The reachability tree is built in an incremental way, according to the trajectories sampled during the Monte Carlo simulation. Initially, only one node is contained in the tree, representing the actual state of the system.

In each trial, a trajectory for the system is sampled, starting from the root: at every step a single transition is fired, considering a sampled firing delay. New states are reached as a consequence of transitions firing. A single trajectory is characterized by

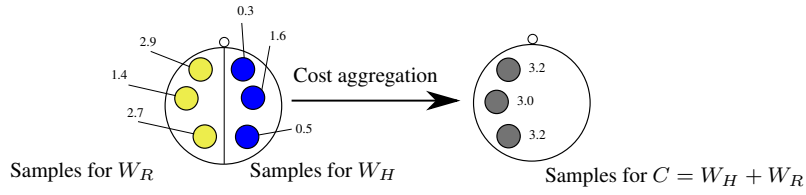


Figure 7.7: Cost aggregation for the single leaf of a reachability tree.

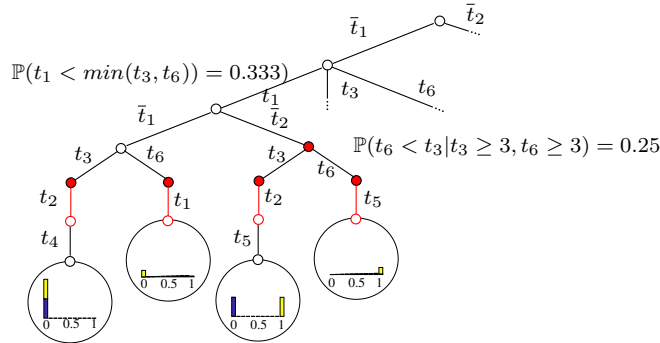


Figure 7.8: Excerpt of the reachability tree for the example in Fig. 7.6. Nodes correspond to firing events. Leaves are labeled with the probability distributions of waiting times for the human W_H (blue bars) and the robot W_R (yellow bars). Red segments correspond to the states where some places associated to waiting conditions of the human or the robot are marked (incurring in a cost). Filled [Empty] red nodes correspond to the firing of transitions that will mark [unmark] a red place.

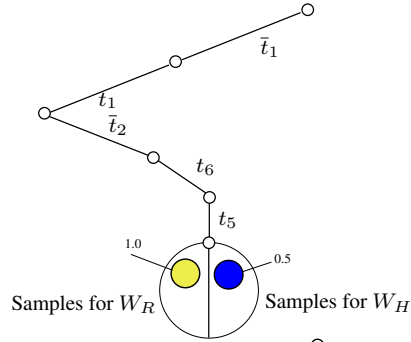
the aforementioned sequence of events together with the times reached after every firing. Situations of conflict are handled as described before.

Suppose to sample as first trajectory the one reported at the top of Table 7.2. It is easy to prove that with that sequence of events, the overall waiting times for the robot and human are $W_R = 1s$ and $W_H = 0.5s$, respectively. The reachability tree is updated, assuming the aforementioned waiting costs as additional samples for the distribution describing the possible waits (right picture at the top of Table 7.2). Another simulation is performed with the same approach, collecting a new sampled trajectory for the system, reported at the middle of Table 7.2. In this case, the human is allowed to begin the assigned action as soon as he/she is ready, since the necessary preconditions are already true. Accordingly, the waiting costs are equal to $W_R = 1s$ and $W_H = 0.0s$, respectively. The new nodes reached when performing this trial of the Monte Carlo, are added to the reachability tree (picture on the right of the middle part of Table 7.2). Now, consider the trajectory reported at the bottom of Table 7.2: the sequence of transitions fired is the same as in the first trial, but with different values for the firing delays, leading to waiting costs $W_R = 1s$ and $W_H = 0.9s$. Such quantities are considered as additional samples used to correct the distribution of the waiting times associated to the leaf reached by this trajectory.

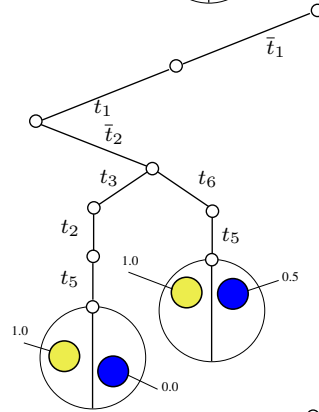
After many trials, the detailed reachability tree reported in Figure 7.8 is obtained. Observe that the leaves of the tree are endowed with the empirical probability distributions of the robot's and human's waiting times, obtained aggregating the samples for the waiting cost. Samples of W_R and W_H can be in turn aggregated for producing samples of the entire cost C , see Figure 7.7.

With reference to the same example, Figures 7.9 and 7.10 depict the back-propagation

fired transition	reached time [s]
\bar{t}_1	0
t_1	3
\bar{t}_2	3
t_6	3.5
t_5	4



fired transition	reached time [s]
\bar{t}_1	0
t_1	3
\bar{t}_2	3
t_3	3.6
t_2	3.6
t_5	4



fired transition	reached time [s]
\bar{t}_1	0
t_1	3
\bar{t}_2	3
t_6	3.1
t_5	4

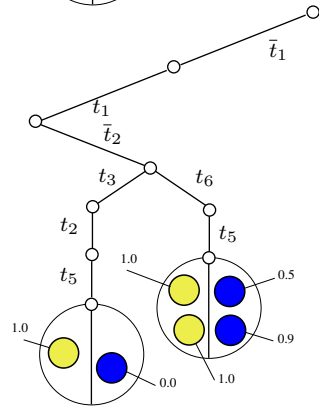


Table 7.2: Sampled trajectories for the system in Figure 7.6. On the left the sequence of events involved in every single trajectory, with the corresponding times. On the right the progressive update of the reachability tree.

process for the calculation of the cost distributions at non-leaf nodes. More precisely, Figure 7.9 illustrate the application of rule (a) to an uncontrollable conflict, while Figure 7.9 the application of rule (b) to a controllable conflict.

The computational time absorbed for computing a new plan to execute for this example is in the order of milliseconds. In particular, when assuming a scheduling horizon equal to $T_{sch} = 10$ s, the average time is equal to 23.9 ms. The latter quantity increases to 35.4 ms when considering an horizon of $T_{sch} = 20$ s, and to 85 ms for an horizon of $T_{sch} = 30$ s.

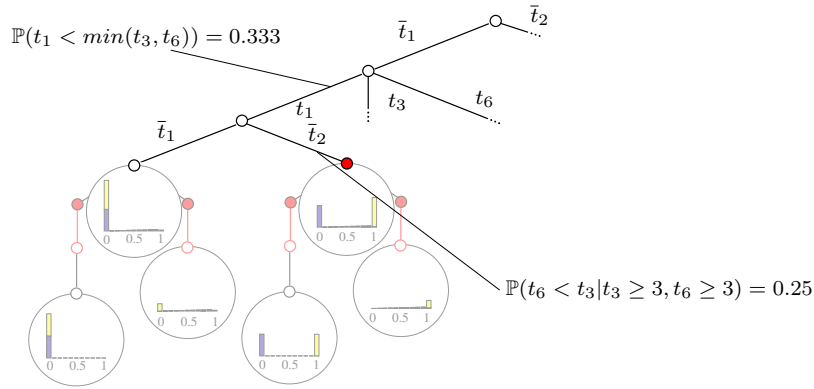


Figure 7.9: Propagation of the costs for uncontrollable transitions: the probability distribution of the parent node is the union of the probability distributions of its children nodes.

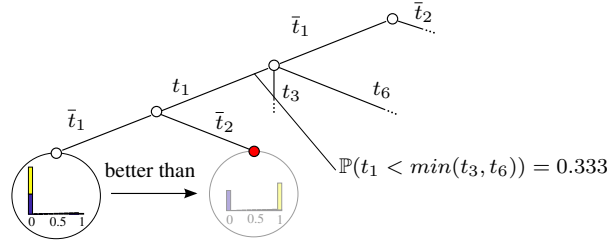


Figure 7.10: Propagation of the costs for controllable transitions: the distribution of the costs is inherited from the child having the lowest cost. In this case firing transition t_1 is evaluated to be more convenient than firing t_2 , therefore the branch starting from the latter is pruned from the tree.

7.3.2 Off line simulations

The performance of the Monte Carlo approach so far described were compared both with other state of the art scheduling methods, through off line simulations, and with a simpler approach in a realistic human robot collaborative assembly. The results of the off line simulations are presented in this Section, while Section 7.5.2 describes the performance achieved in the real experiments.

The simple assembly process depicted in Fig. 7.11, consisting of two sequential jobs with three actions each, will be considered. Tasks are labeled with the agent that has to execute them (robots R_1 and R_2 and the human H). Notice that each agent is in charge of an action in each job. The task durations are modeled by Gaussian distributions with a mean equal to 15 s and a standard deviation of 1.67 s. The human persistently repeats the same assembly pattern, alternating the execution of the task in job A twice with the execution of the task in job B twice.

With reference to this process, a simulation campaign was carried out to compare the Monte Carlo scheduler with state-of-the-art methods: the centralized deterministic approach of [93] and the multi agent stochastic one proposed in [76], which are representative of the two main classes of approaches in the literature. In order to apply these two approaches the distributions associated to the task durations are assumed to be known. In particular, for [93] the mean time is considered to produce the long time horizon plan, since the approach is deterministic. Notice, conversely, that the approach

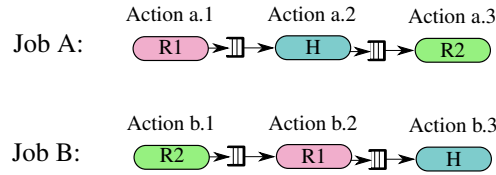


Figure 7.11: Assembly addressed by off line simulations.

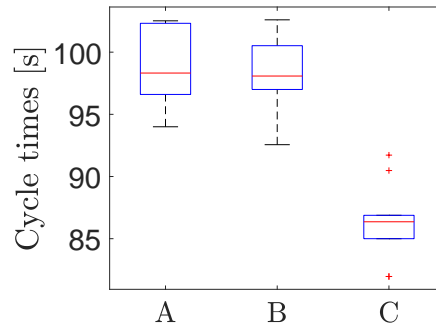


Figure 7.12: Cycle time distributions obtained in the off line comparative simulations: A refers to [93], B to [76] while C to the Monte Carlo scheduler in [19].

proposed in [19] does not require any such information.

20 distinct simulations for each approach were performed. A total amount of 10 cycles of the human’s assembly pattern were simulated for each trial. The distribution of the cycle times obtained in the simulations is reported in Fig. 7.12. Apparently, the Monte Carlo approach outperforms the other two, the main reason being that the other approaches spend a lot of time repairing the plans computed off-line assuming that the human is a controllable agent. Indeed, since the behavior of the human is quite different from the scheduled one, the system frequently ends up in deadlocking situations that must be solved, wasting time.

Notice that the methods of [76] and [93] employ different recovery strategies. In the former case, the plan is repaired using specific rules, whereas in the case of [93] the long time horizon plan is suspended and one of the task enabled for the robots (excluding the controllable wait) is randomly executed until the deadlock is solved. The distributions employed in this analysis were stationary. If, instead, time varying distributions are considered, the performance gap between approach in [19] (which learns online the distributions) and those of [93], [76] could even be larger.

7.4 Fuzzy scheduling

The possibilistic approach here described is an alternative to the probabilistic one of Section 6.2.1. In fact, the approach detailed in this Section exploits the algebraic rules of fuzzy numbers [133] to propagate the uncertainty. In [16], the firing delays d of the uncontrollable transitions are assumed to be triangular fuzzy numbers, see Appendix D. Two main reasons may motivate such a choice. The first one is that the activities durations are known in a very approximate and imprecise way, especially when considering those undertaken by the human operator. Also when considering the approach in the previous Section, the modelling of realistic distribution might require the collection

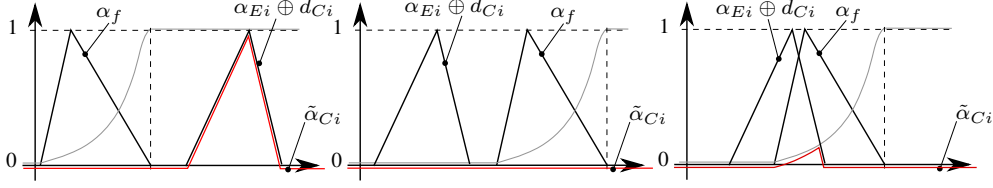


Figure 7.13: Examples of computation of $\tilde{\alpha}$. For the left picture $\max_t (\tilde{\alpha}_{C_i}(t)) = 1$, as the support of the arrival time of the child is completely above the one of the father, while for the picture in the middle $\max_t (\tilde{\alpha}_{C_i}(t)) = 0$ as the opposite situation arises, meaning that for the situation in the middle, that node would have been removed from the children list. The situation depicted on the right is intermediate between the aforementioned cases.

of a big amount of data, before a scheduler exploiting PDFs begins to properly work. On the opposite, the uncertainty in the activity durations could be modelled considering a restricted set of statistics, which is efficient only by making use of fuzzy sets.

The second reason is that the propagation of fuzzy numbers presents some interesting numerical properties, reducing the computational load of the approach. Indeed, when dealing with the propagation of probabilistic quantities, the solution of complex integrals must be often addressed, equation (6.6), which is almost all of the times non tractable in a closed form. Moreover, numerical methods like the Monte Carlo approaches in Section 7.3, are able to obtain only approximate solutions, exploiting algorithms that require hundreds or thousands of iterations. This fact is not true when considering fuzzy theory.

Computation of the reachability tree

The RT is computed in a closed form, propagating fuzzy quantities. The exploration of a certain branch is interrupted when the arrival times in the corresponding leaf starts to be beyond the scheduling horizon T_{sch} . The commands sent to the robots are computed after obtaining the RT, as for the other scheduling approaches.

When assuming the firing delays as fuzzy distributed, the arrival times α (Section 6.2.1) into the nodes of the RT, are also fuzzy numbers. The steps reported in Algorithm 5 summarize all the computations required to build the fuzzy RT and will be discussed in the following.

The routine **Conflict** is in charge of computing the arrival times of the nodes involved in an uncontrollable conflict. Clearly, when not considering the presence of conflict, the unconditioned arrival time $\underline{\alpha}_{C_i}$ is a summation of triangular numbers:

$$\bar{\alpha}_{C_i} = \alpha_{E_i} \oplus d_{C_i} \quad (7.3)$$

where α_{E_i} is the arrival time of $enab(S_{C_i})$, see Section 6.2.1. However, the presence of the uncontrollable conflict must be taken into account. The computation of the conditional fuzzy distribution α_{C_i} is subdivided into two main steps:

- Stage 1: For every transition in the conflict, the arrival time $\tilde{\alpha}_{C_i}$ must be first determined. $\tilde{\alpha}_{C_i}$ accounts for the fact that the arrival time in S_f , *i.e.* the father of S_{C_i} , must be lower than the arrival time in S_{C_i} . $\tilde{\alpha}_{C_i}$ is computable as an intersection of

Algorithm 5

```

1: procedure TREE COMPUTATION( $M_0$ )
2:    $RT = \{S_0\}$ ;
3:    $Open \leftarrow \{S_0\}$ ;
4:   while  $Open \neq \emptyset$  do
5:      $Open2 = Open$ ;
6:      $Open = \emptyset$ ;
7:     for all  $S_{o2} \in Open2$  do
8:        $E \leftarrow$  enabled transitions in  $S_{o2}$ ;
9:        $\bar{E} \leftarrow \{t_e \in E | t_e \text{ is controllable}\}$ ;
10:      if  $\bar{E} \neq \emptyset$  then
11:        for all  $\bar{t}_e \in \bar{E}$  do
12:           $m_{new} = m_{o2} + Post^{\bar{t}_e} - Pre^{\bar{t}_e}$ ;
13:           $S_{new} \leftarrow \langle m_{new}, \alpha_{o2} \rangle$ ;
14:           $RT \leftarrow RT \cup S_{new}$ ;
15:           $Open = Open \cup S_{new}$ ;
16:        else
17:           $T = \{t_{C_i} \in E \setminus \bar{E}\}$ ;
18:           $S_{C_1, \dots, n} \leftarrow \mathbf{Conflict}(T, S_{o2})$ ;
19:          for all  $i = \{1, \dots, n\}$  do
20:            if  $\alpha_{C_i}$  is below  $T_{sch}$  then
21:               $RT \leftarrow RT \cup S_{C_i}$ ;
22:               $Open \leftarrow Open \cup S_{C_i}$ ;
    
```

events (see Appendix D):

$$\tilde{\alpha}_{C_i}(t) = (\alpha_f < t) \cap (\alpha_{E_i} \oplus d_{C_i})(t) = \min_t \left(L(\alpha_f, t), (\alpha_{E_i} \oplus d_{C_i})(t) \right) \quad (7.4)$$

When $\max_t (\tilde{\alpha}_{C_i}(t)) < \varepsilon$, *i.e.* a small constant, the node is considered unreachable and S_{C_i} is removed from the list of transitions constituting the conflict. Figure 7.13 reports some examples of computations of $\tilde{\alpha}$.

The arrival time $\bar{\alpha}$, *i.e.* the one agnostic of the conflict, Section 6.2.1, is computed as follows:

$$\bar{\alpha}_{C_i}(t) = \frac{1}{p} \tilde{\alpha}_{C_i}(t) \quad (7.5)$$

$$p = \max_t (\tilde{\alpha}_{C_i}(t)) \quad (7.6)$$

Notice that $\bar{\alpha}_{C_i}$ is a convex fuzzy set (Appendix D), but is not a triangular fuzzy number. It represents an arrival time compatible with the arrival time of $father(S_{C_i})$ and it is exploited in Stage 2.

- Stage 2: Knowing the set of times $\bar{\alpha}_{C_1, \dots, n}$, the computation of every α_{C_i} can take place. The fuzzy set describing the arrival time after winning a conflict, can be computed, as another intersection of events:

$$\begin{aligned} \hat{\alpha}_{C_1}(t) &= \bar{\alpha}_{C_1}(t) \bigcap_{l=2, \dots, n} \left(\bar{\alpha}_{C_l}(t) > t \right) \\ &= \min \left(\bar{\alpha}_{C_1}(t), L(\bar{\alpha}_{C_2}, t), \dots, L(\bar{\alpha}_{C_n}, t) \right) \end{aligned} \quad (7.7)$$

The possibility to win the conflict P_{fi} is computed as:

$$P_{fi} = \max_t \left(\hat{\alpha}_{C_i}(t) \right) \quad (7.8)$$

α_{C_i} is finally computed by normalizing $\hat{\alpha}_{C_i}$:

$$\alpha_{C_i}(t) = \frac{1}{P_{fi}} \hat{\alpha}_{C_i}(t) \quad (7.9)$$

It is easy to prove that α_{C_i} is a convex fuzzy set having the alpha-cut $\alpha_1(\alpha_{C_i})$ which is a singleton. Therefore, the arrival time α_{C_i} can be well approximated by the following triangular fuzzy number:

$$\begin{aligned} \alpha'_{C_i} &= \langle A_1, A_M, A_2 \rangle \\ A_M &= \alpha_1(\alpha_{C_i}) \\ A_1, A_2 &= \text{extremals of } \alpha_0(\alpha_{C_i}) \end{aligned} \quad (7.10)$$

α'_{C_i} is assumed as arrival time α_{C_i} for node S_{C_i} . The possibility to reach node S_{C_i} , call it F_{C_i} , is computable considering an intersection of events: the winning of all the preceding conflicts in the RT. Therefore, P_{C_i} is computed in this way:

$$F_{C_i} = \min \left(F_{C_i}, F_{\text{father}(S_{C_i})}, F_{\text{father}(\text{father}(S_{C_i}))}, \dots \right) \quad (7.11)$$

The computations expressed by the above equations are an adaptation of those reported in Section 6.2.1, considering the fuzzy propagation rules. As can be seen, there are many analogies. However, integrals are replaced with the *min* or *max* operator, leading to a significant computational advantage.

7.4.1 Selection of the best plan

The set of leaves contained in the RT expresses all the possible evolutions of the system within the scheduling horizon. Scheduling aims to select the optimal evolution to impose. This choice is performed by computing for every node S_i a fitness parameter called J_i . The propagation of the fitness parameter J across the RT is similar to the cost propagation presented in Section 7.3. The following cases have to be considered:

- S_i is a leaf. The path in the tree leading to this leaf has to be considered, which is a possible trajectory of the system. The total idle time W_R spent by the robots when following this trajectory can be easily computed by simply considering the number of times the robots were forced to wait, see Section 6.3.1.

On the other hand, the idle time spent by the human worker is non trivial to evaluate. Indeed, we might think to compute such quantity as the difference between $\alpha_I = \langle I_1 I_M I_2 \rangle$ and the arrival time $\alpha_S = \langle S_1 S_M S_2 \rangle$, which are the arrival times into nodes reached after firing a transition that insert or remove, respectively, a token from the coloured place in Figure 6.11. However, the difference is an operation not always possible when dealing with the fuzzy domain. Therefore, W_H is computed in order to obtain a quantity strictly correlated to the magnitude of the waiting time:

$$W_H = \varphi_1 \cdot \varphi_2 \cdot \varphi_3 \quad (7.12)$$

where coefficients φ are defined in this way:

$$\varphi_1 = 1 - \max_t \left(\min(\alpha_I(t), \alpha_S(t)) \right) \quad (7.13)$$

$$\varphi_2 = 1 + \frac{S_M - I_M}{T_{sch}} \quad (7.14)$$

$$\varphi_3 = \frac{S_1 - I_1}{S_2 - I_1} \quad (7.15)$$

Notice that in the worst condition, *i.e.* $I_2 < S_1$, φ_1 assumes the maximal value. After computing W_H and W_R , the fitness parameter can be computed as:

$$J_i = \frac{1}{w_1 W_{Hi} + w_2 W_{Ri}} \quad (7.16)$$

w_1 and w_2 are tunable parameters that can be set considering the cost per hour of the worker against the one of the robots.

- J_f is a non leaf and a controllable conflict departs from this node: it is possible to decide which action to perform after arriving in this node, *i.e.* the one leading to the child having maximal fitness is the one to impose. Therefore the fitness is assumed equal to (refer to Figure 6.7):

$$J_f = \max_{i=1, \dots, n} (J_{Ci}) \quad (7.17)$$

- J_f is a non leaf and an uncontrollable conflict departs from this node. In this case, J must reflect the fact that the system will evolve from this node in a non deterministic way. Considering the possibilities P_{fi} of the children, the fitness is computed as a combination:

$$J_f = \sum_{i=1, \dots, n} J_{Ci} \cdot P_{fi} \quad (7.18)$$

Considering the rules proposed for computing J , it is clear that the computation process proceeds in a backward way: all the fitness of the leaves are computed at first, then the computations for the ancestors are performed, traversing the levels of the tree, analogously to the scheduler in Section 7.3. After the propagation of every coefficients J from the leaves to the root, the selection of the best commands to send for the robots is trivial: the action leading from the root to the one having the maximal J .

7.5 Validating experiments

7.5.1 Use case a

The scheduling strategy presented in Section 7.2 was applied to the use case here detailed. Figure 7.14, depicts the workspace shared by the human operator and YUMI. In this section the results obtained in two kinds of experiments are reported, in which the human operator continuously performs the same pattern of actions. In the first case (named pattern 1), the human makes one autonomous assembly followed by one collaborative operation, while in the second one (named pattern 2), he/she performs two

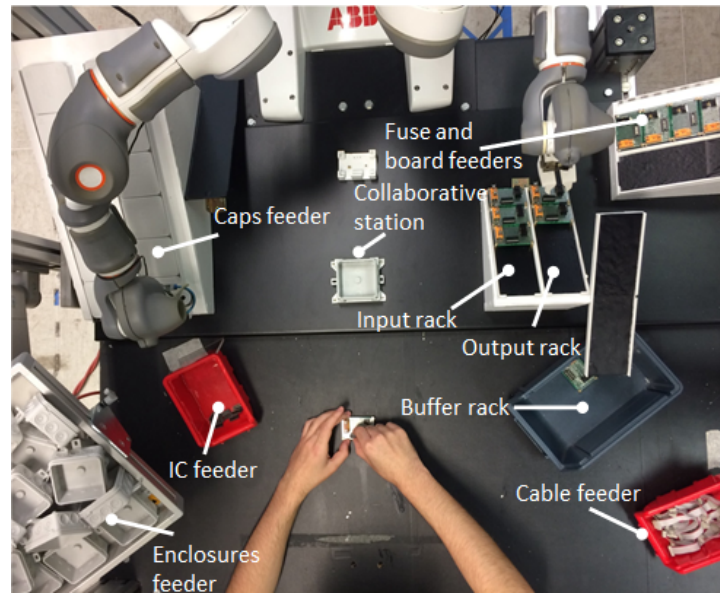


Figure 7.14: Workspace shared by the human operator and YUMI.

autonomous assemblies followed by two collaborative operations. Results regarding pattern 1 are reported in Fig. 7.15(a). Instead, Fig. 7.15(c) reports the results obtained by applying a round robin scheduler, which alternates an autonomous assembly with a quality control.

Then, this sequence of robotic operations is interrupted only when the human begins a new collaboration (as soon as the robotic arms become available). As can be seen, when considering the round robin approach, the human operator spends a relevant time remaining inactive while waiting for the robots to begin a new collaboration. Instead when considering the controller which employs the pcTPN described, the operations are scheduled so as to let the robots be ready to collaborate as soon as the human is ready too. In this way an improvement for the throughput of the system is achieved. Indeed, when considering a round robin approach, one finite product is produced every 35.80 s on average, while with the proposed approach this number is reduced to 27.63 s (with a reduction of cycle time of approximately 23%). Regarding pattern 2, the results³ are reported in Fig. 7.15(b) and 7.15(d). For this case, the sequence followed by the round robin scheduler is made up by two autonomous assemblies followed by two quality control operations. The average throughput in this case is one product every of 33.35 s, with the round robin controller, and one product every 30.10 s when adopting the strategy proposed in this work (with a reduction of cycle time of approximately 10%).

7.5.2 Use case b

The effectiveness of the Monte Carlo approach described in Section 7.3 has been validated with the help of a collaborative workspace consisting of an ABB dual-arm robot YUMI, an ABB IRB 140 robot, two linear positioners (denoted Carts in the following) and a human operator, see Figure 7.17. A MICROSOFT KINECT is used to monitor the positions of the operator and a processing unit runs the scheduling algorithm. The

³Videos are available at <http://www.youtube.com/watch?v=KG7WLNdi8Uw>.

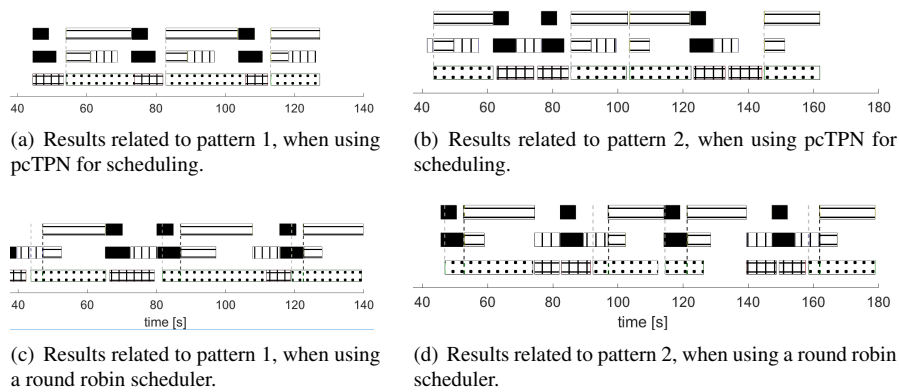


Figure 7.15: In all the above figures the top line reports the actions executed by right arm of YUMI over time, the middle one refers to the actions executed by the left arm, while the bottom line is related to the actions executed by the human operator. The legenda related to the operations represented is depicted in Figure 7.16. Dotted gray lines indicate time instants at which the human starts to be available for a new collaboration, while dotted black line refers to instants at which YUMI begins the collaborative task (those two kinds of lines are coincident when considering the scheduling obtained with the use of pcTPN).

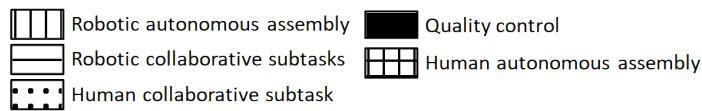


Figure 7.16: Legenda related to Figure 7.15.

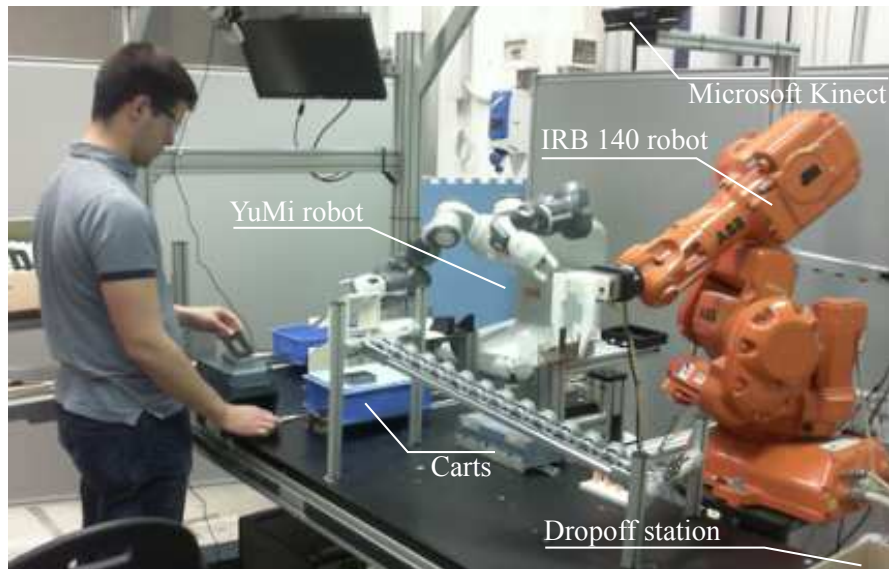


Figure 7.17: The experimental setup with the two robots, the carts and the human operator.

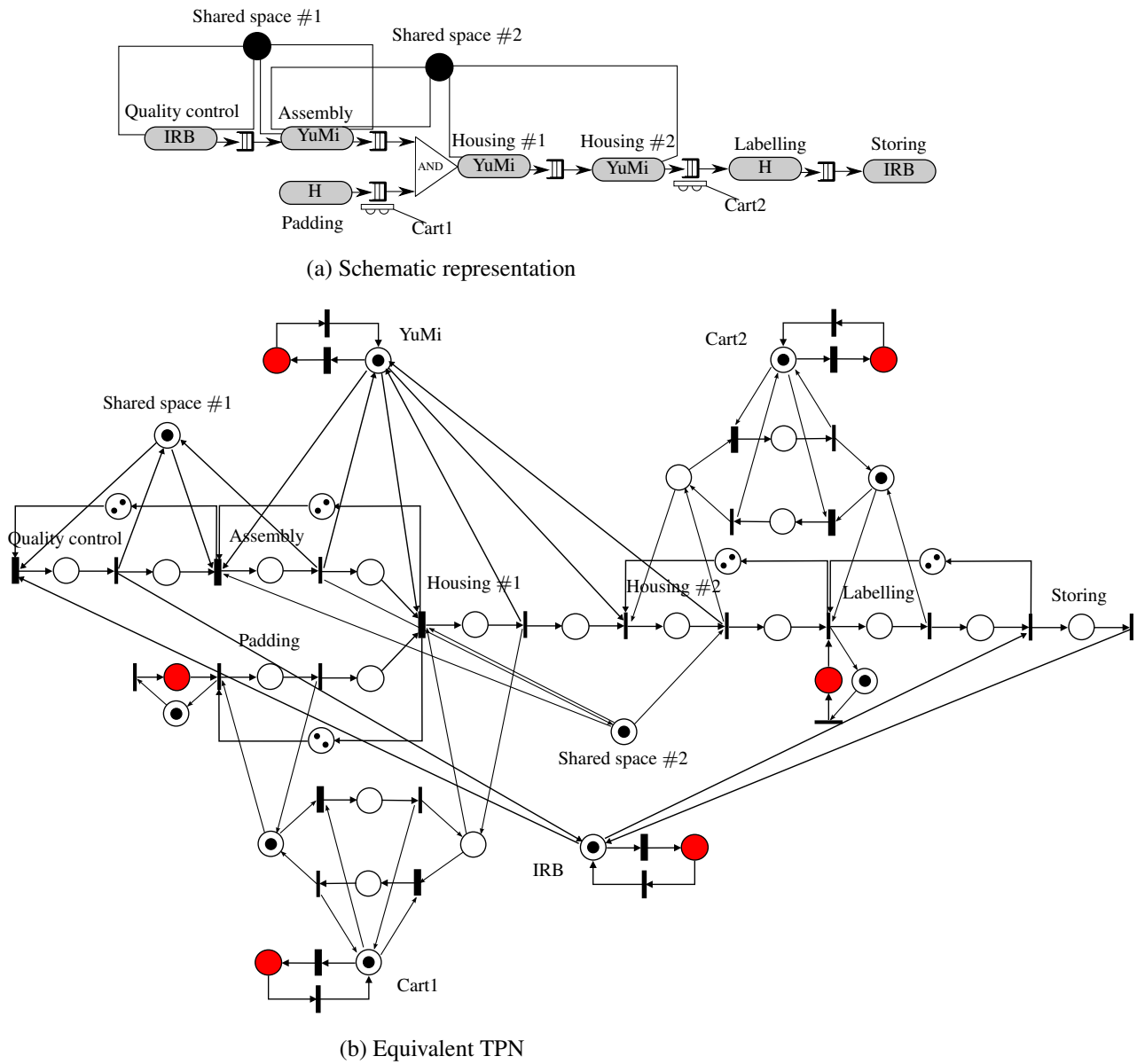


Figure 7.18: Schematic representation and equivalent PN adopted for the use-case.

MICROSOFT KINECT standard APIs are used to retrieve the positions of the operator's wrists over time, interpreting the depth stream. Then, the detection of a human related event is triggered by the entering or exiting of the operator's hand into specific spherical areas, for the purpose of taking or placing components⁴.

The human and the robots cooperate to prepare a kit consisting in the components of a USB/microSD adapter and to house it in a metal box. In particular, the human operator is responsible for inserting the padding into the metallic box and for placing a label on it, after having trimmed it from a sheet. The IRB 140 robot performs a functional test on the USB adapter and stores the completed part into a drop-off station (see again Figure 7.17) for shipment. Finally, the YUMI robot is in charge of performing the assembly of the adapter with its cap, as well as for housing the adapter in the box. The modelling pcTPN is represented in Figure 7.18. Notice that the housing operation has been split into two parts as it is particularly long and requires the two carts to be available at the same time. By dividing it into two parts, it is possible to release the use of CART 1 after the completion of the first part of the activity and to continue with the second one when CART 2 becomes available.

20 volunteers, males and females within MSc and graduate students, were enrolled for the experiments. They were asked to perform the assembly of USB adapters following one of two possible patterns:

- **Pattern 1:** the operators execute a one-piece-flow assembly strategy, cyclically executing the following three actions: padding, trimming a new label, and applying the trimmed label in the final product;
- **Pattern 2:** the operators perform two consecutive padding operations, followed by two consecutive trimmings and placements of the labels.

Moreover, two different scheduling algorithms were adopted to control the robots and the two carts:

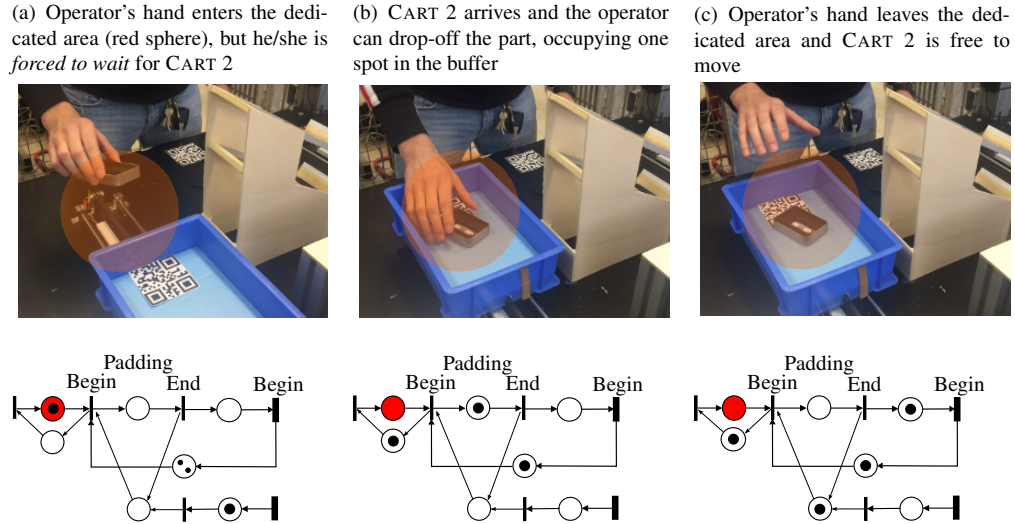
- **Scheduler A:** the Monte Carlo scheduler described in Section 7.3;
- **Scheduler B:** an approach which neglects the real PDFs describing the firing delays, assuming a uniform distribution for all the uncontrollable transitions. The extremals of the support of the modelling uniform distributions are the maximum firing times measured from the system. An optimal plan is computed performing, also for this kind of scheduler, a Monte Carlo simulation, but considering the aforementioned uniform distributions. Indeed, this scheduler is something intermediate between the approach proposed in Section 7.2 and the one proposed in this Section.

Volunteers were divided into 4 groups: 1.A, 1.B, 2.A and 2.B. For each group, a particular assembly pattern was followed, and a particular scheduling approach was considered, with obvious notation⁵. The coefficients c_R and c_H involved for the computation of the cost C were assumed equal to 0.5 and 1.0, respectively. In each experiment, a total of 9 products were assembled. The horizon time considered for the scheduler was

⁴A video to help understanding the assembly cycle and the activities performed by the different agents is available at https://youtu.be/_Jx01mNZ1C8.

⁵For instance, volunteers in Group 2.A were asked to perform Pattern 2, while the robots were controlled with Scheduler A.

Table 7.3: Excerpt of the PN of Figure 7.18 and example of synchronization between the station and the PN model used for scheduling.



set equal to $T_{sch} = 30 s$, while the deterministic delay considered for the controllable agents was set equal to $t_w = 3 s$. The mean computation time for building a new reachability tree (including the propagation of costs, when considering $N = 150$ trials for the Monte Carlo simulation is equal to $286 ms$.

Analysis

Figure 7.19 reports the measured cycle times, computed as the temporal differences between the completion of two successive products (both detected as the firing of an End transition of a Storing, see Figure 7.18).

A clear statistical evidence suggests that the proposed approach (Scheduler A) is able to reduce the cycle time for both the two considered assembly patterns, leading to an increased productivity. This fact is confirmed by the results reported in Figure 7.24, which compare the evolution of the system for an experiment in Group 1.A with another experiment from Group 1.B. As it can be seen, the time wasted by the human is

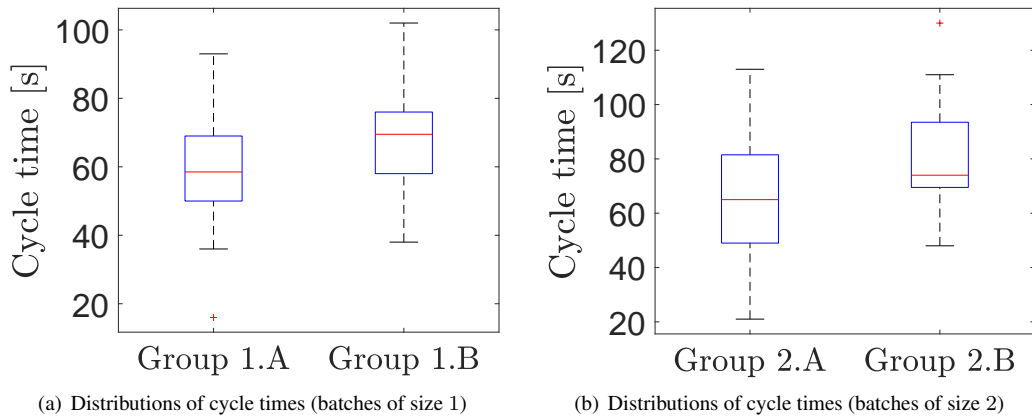


Figure 7.19: Distributions of the measured cycle times, for the two assembly patterns considered.

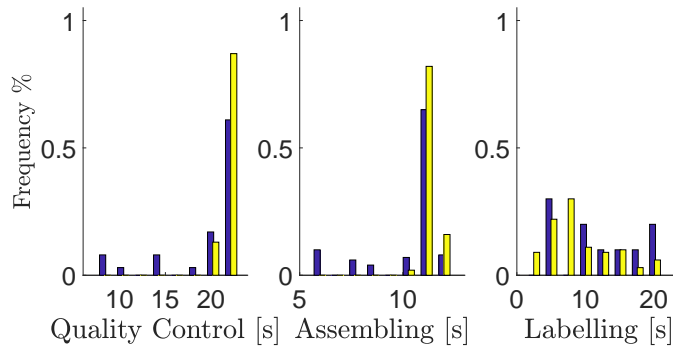


Figure 7.20: Distributions of the measured durations of some uncontrollable transitions of the net reported in Figure 7.18 in the middle (yellow) and at the end (blue) of the experiment.

significantly higher for the trial of Group 1.B, leading to higher cycle times. In particular, when the operator from Group 1.A finished the 9-th products, the operator in Group 1.B was only able to finalize 7 products. Besides the qualitative results contained in Figure 7.24, the median cycle time for Group 1.A turns out to be smaller than the one of Group 1.B ($p = 0.9971$, Wilcoxon test with $\alpha = 0.05$), and similarly the median of Group 2.A is lower than the one of Group 2.B ($p = 0.9926$).

The reduced cycle times experienced when applying the proposed approach are justified by the reduced amount of time wasted in an idle state for both the operator and the robots, as reported in Figures 7.21 and 7.22. The idle times of the robots are computed by summing all the elapsed times between the ending of an action and the beginning of a new one. Conversely, the inactivity times of the operators are computed considering the time spent by the tokens of the PN of Figure 7.18 in the coloured places indicated in the same figure (see also Figure 7.3(b)). As it can be seen, the distributions of the idle times highly correlates with ones reported for the cycle times.

Analysis of the firing distributions

Figure 7.20 reports the distributions of the measured times for some of the transitions of the pcTPN of Figure 7.18. The values retrieved from all the experiments of Group 1.A

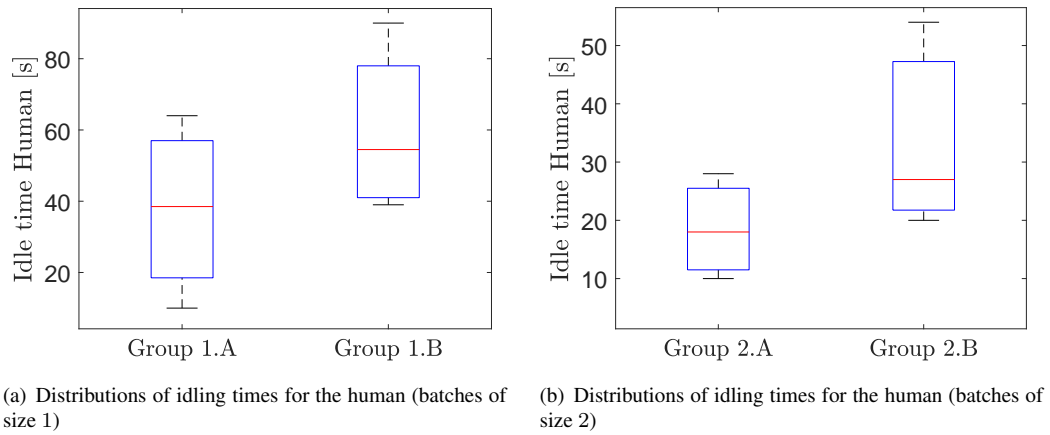


Figure 7.21: Distributions of the human inactivity times, for the two assembly patterns considered.

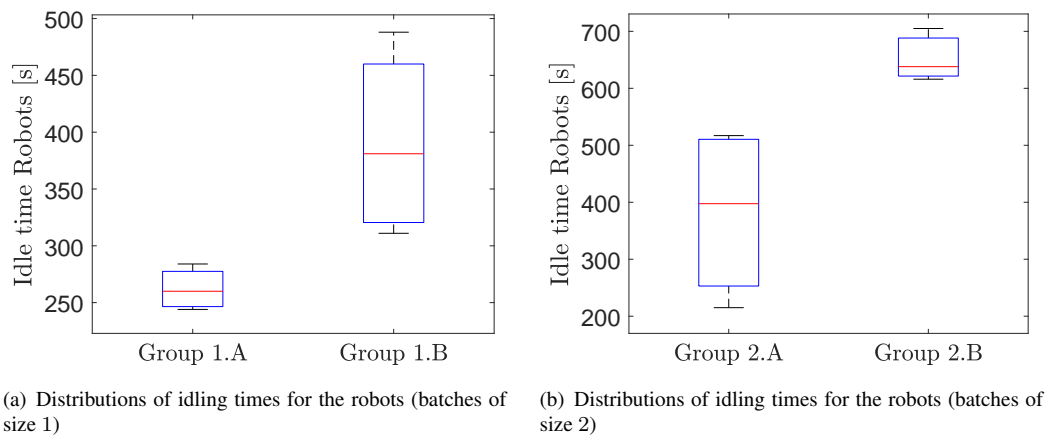


Figure 7.22: Distributions of the agents inactivity times, for the two assembly patterns considered: the summations of all the idling times of the agents in the system is reported.

and 1.B, for a specific transition, are grouped together. Such times are clearly not exponentially distributed, implying that any approach based on SPN would be unsuitable for this application.

Moreover, Figure 7.20 also compares the distributions of three particular activities, approximately after 5 assembly cycles with the same distributions at the end of the same experiment. It is possible to notice that distributions are not stationary and their time dependency has to be accounted for in the scheduling algorithm through learning and adaptation mechanisms.

Learning capabilities

With the aim of investigating the learning capabilities of the scheduling algorithms, Figure 7.23 reports the evolution of the total idle times of agents (YUMI, IRB 140 and the operator) during the experiments from Groups 1.A and 1.B. It is possible to appreciate a monotonic decreasing trend of the idle time in each cycle for all the experiments performed, indicating the learning capability of the two scheduling algorithms. Both scheduling algorithms progressively collect duration measurements from the system (see Section 7.1) and consequently refine the corresponding duration models.

The evolution of the idle times for each of the volunteers was fitted with a linear regression. The slopes corresponding to Group 1.A were steeper than those corresponding to group Group 1.B ($p = 0.7897$, Wilcoxon with $\alpha = 0.05$), indicating a slightly accentuated learning efficiency for the scheduler proposed in this work (Scheduler A).

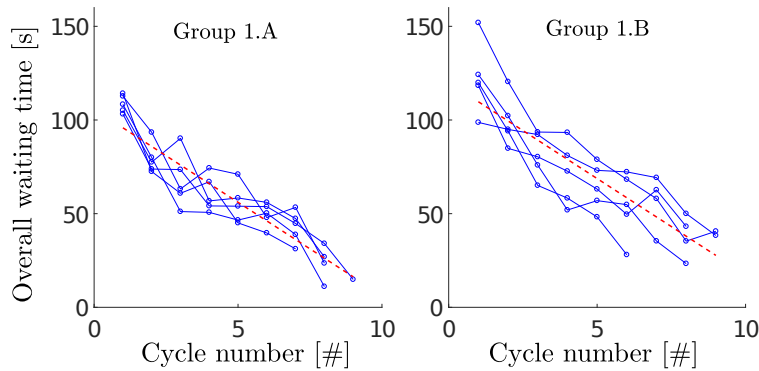


Figure 7.23: The evolution of the overall wait time of the agents, during the experiments of Group 1.A and Group 1.B. The reported values, refer to the summation of the idle times of YUMI, IRB 140 and the human operator. The dashed red curves are the regressed lines interpolating all the data.

7.5. Validating experiments

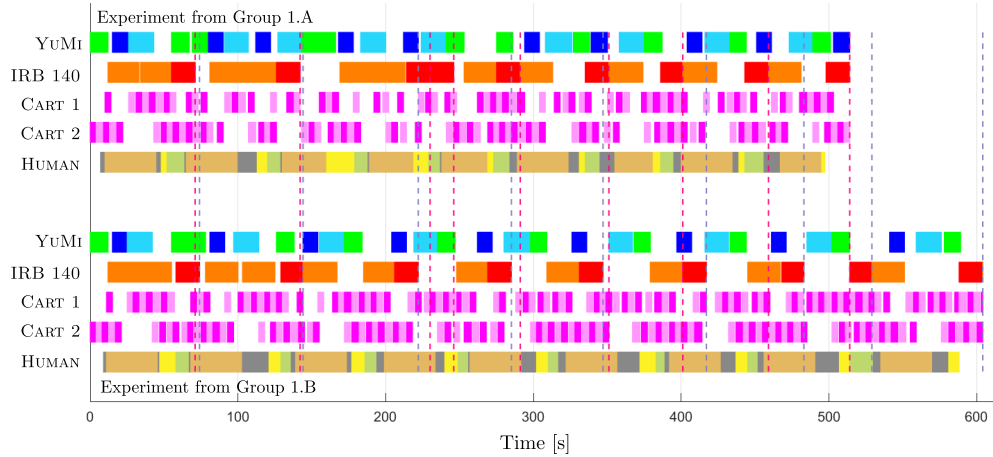


Figure 7.24: Sequence of events occurred for two particular experiments: one from Group 1.A (top) and one from Group 1.B (bottom), see Figure 7.5 for the legend. The temporal duration of the activities performed by the agents is proportional to the length of the corresponding coloured bar (waiting activities are not reported). The dashed vertical lines refer to time instants when a new finite product is available (i.e. the end of the storing operation done by the IRB 140): the red ones referring to the experiment of Group 1.A while the blue ones to that of Group 1.B.

Table 7.4: Composition of the times spent by agents performing the assigned tasks, see Figure 7.5 for the legend. For each task, the overall time spent doing that action is considered for creating the proposed histogram charts, summing the values of all the experiments in a specific group.

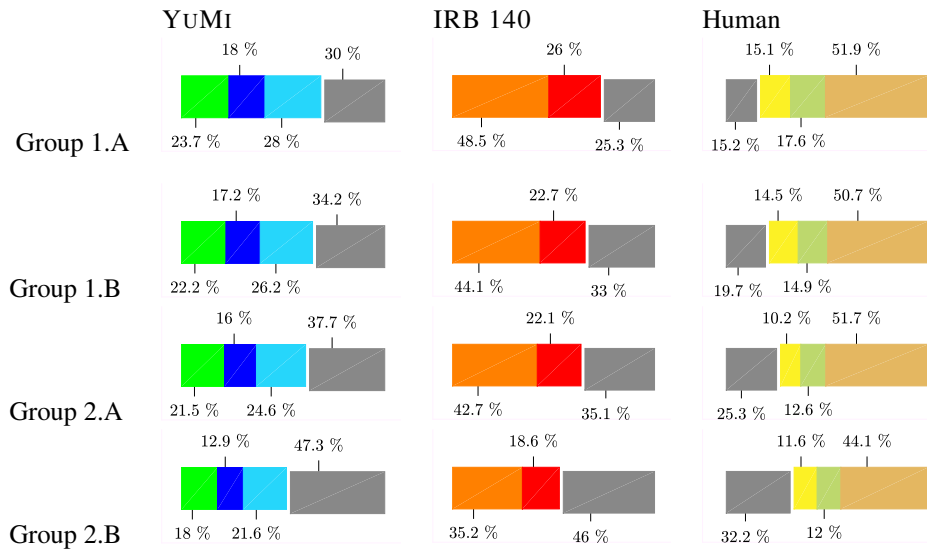


Table 7.5: Color legend for Figures 7.4 and 7.24.

YuMi		IRB 140			
	Assembly		Housing #2		Quality Control
	Housing #1		Idle		Storing
			Idle		
Human		CART 1 & 2			
	Wait for a Cart		Padding		Move towards YuMi
	Labelling		Trim label		Move towards Human

7.5.3 Use case c

The effectiveness of the fuzzy scheduling approach in Section 7.4 was tested in the collaborative assembly of a torch and a clock, see Figure 7.26⁶. The experimental set-up is reported in Figure 7.25. The dual arm YUMI is the robotic co-worker of the human. The operations reported in Figure 7.26 will be now detailed:

- assembly of a torch:
 - Action R1: Assemble the frontal part of the torch with the light inside.
 - Action H1: Screw the frontal part with the body of the torch.
 - Action L1: Move the body of the torch to assembly station A.
 - Action H2: Assemble the batteries of the torch.
 - Action R2: Insert the batteries into the body of the torch.
 - Action H3: Finalize the torch screwing the bottom part and archive the finite product.
- assembly of a clock:
 - Action R3: Assemble the clock dial with the engine.
 - Action H4: Insert the hands.
 - Action L2: Insert the glass into the frontal frame.
 - Action L3: Add to the frontal frame assembly, the dial with the engine and the bottom part of the clock.
 - Action R4: Move the assembled clock into the delivery buffer.
 - Action H5: Finalize the clock by screwing all the angles.

Actions R1,2,3,4 are assigned to the right arm of *YuMi*, while L1,2,3 to the left one. All the other actions are assigned to the human. Work in progress are stored in the buffers located in the green areas of Figure 7.25(c). The QR codes that can be seen in the same Figure, are placed for making the robot aware of which bins are occupied (the robot's hands are embedded with an integrated vision system).

⁶A video showing all these phases is available at <https://www.youtube.com/watch?v=5zhIp51Ndfk&feature=youtu.be>.

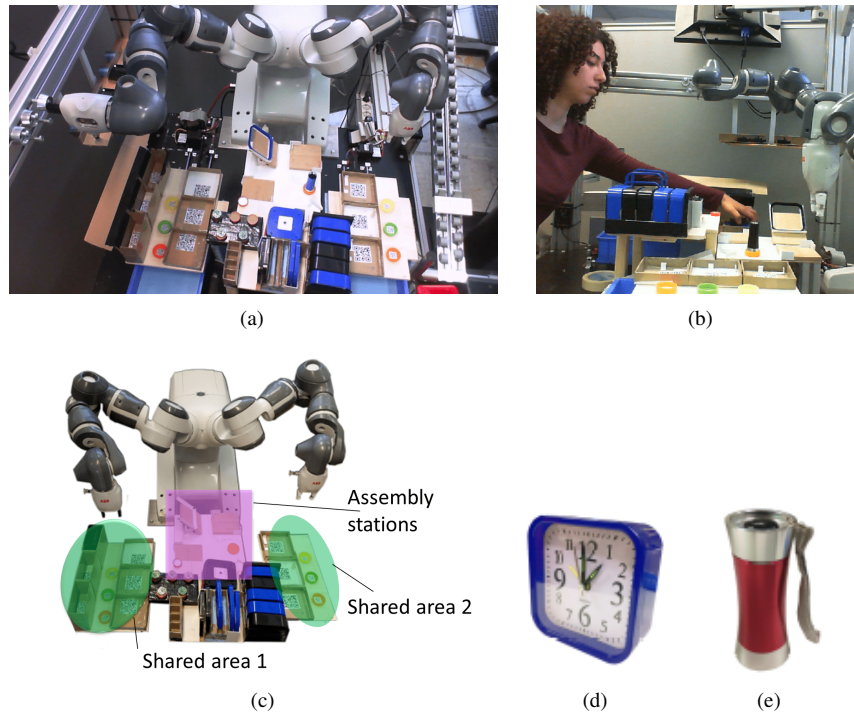


Figure 7.25: *On the top, the layout adopted for the experiments. On the bottom left part, a detailed view of the layout: the violet area contains the stations used by the robots to perform the assigned intermediate assemblies, while the green ones are the buffers through which the human and the robots exchange components. The pictures on the bottom right part depicts the two products to be assembled.*

Three main sources of fuzziness affect the system. The first one is clearly related to the time duration required by the human to perform his or her assigned actions. The second one is the presence of a shared area for the two robots (violet zone of Figure 7.25(c)). This area contains some assembly stations and can be accessed by robots only one at a time: when an arm is already inside, the other one is forced to wait till the exit of the first. Therefore, the time required to perform an action for which the shared area has to be used, is uncertain.⁷

The last source of uncertainty is the motion control imposed for robots. Indeed, for safety reasons, the robots are slowed down along the assigned paths when traversing the green areas of Figure 7.25(c), in case of the simultaneous presence of the operator's hands (a MICROSOFT KINECT camera is employed for tracking the positions of the operator's hands). Therefore, the sharing of a common workspace for humans and robots, increase the non determinism related to completion times of robotic tasks, making the fuzzy approach particularly suited⁸.

10 volunteers were enrolled. Each volunteer was asked to perform two kinds of experiments:

1. Experiment A: the operator was asked to persistently perform the assembly of

⁷In principle the shared area can be modelled as an additional resource of the modelling Petri Net, adding some preconditions to those actions that use this area, as similarly done for the use case in Section 7.5.2. This approach was not followed only to increase the uncertainty of the system, in order to show the capabilities of the scheduler.

⁸Notice that the human and the robots in the use case of Section 7.5.2 did not actively share the space.

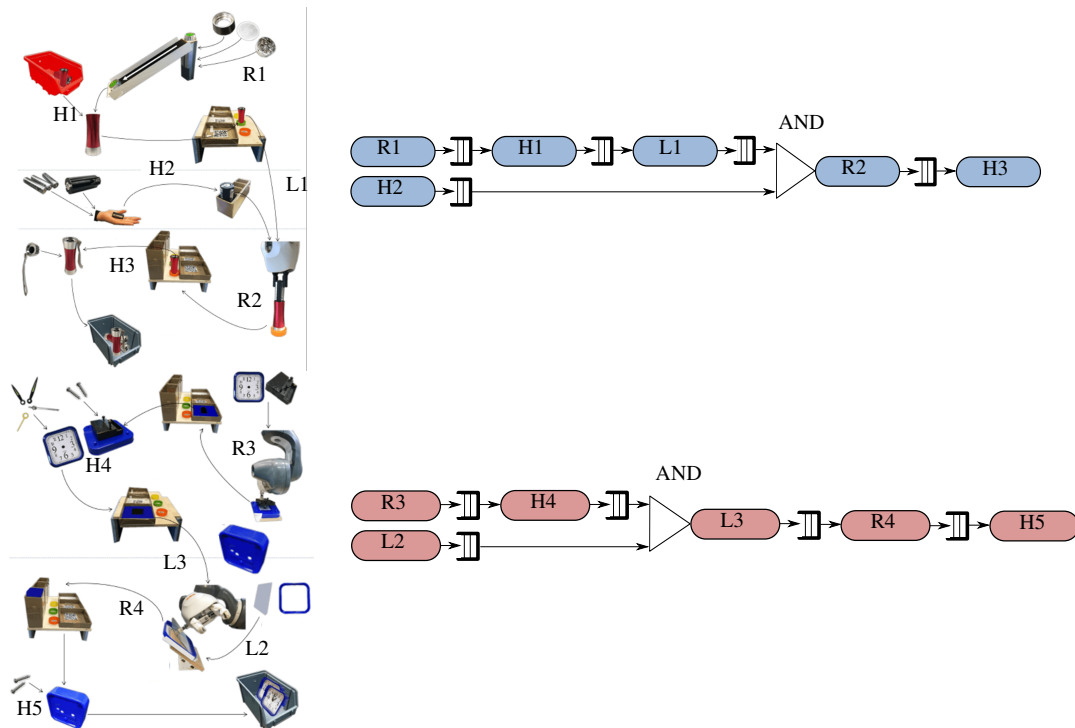


Figure 7.26: On the top the actions required for the assembly of a single torch and the corresponding assembly flow, on the bottom the same for a clock.

torches only, executing in a cyclic manner actions H1, H2 and H3.

2. Experiment B: in this case the operator was asked to repeat the following pattern of actions: H1, H4, H2, H5 and H3, *i.e.* alternating the assembly of a new torch with the one of a new clock.

In each trial of the Experiment A, the involved operator assembled a total number of 5 torches; while in the case of Experiment B, 5 torches and 5 clocks were produced. The operator was forced to repeat the same pattern of the two kind of experiments. Therefore, he or she was forced to wait till all the preconditions of the following action to undertake were not met.

Each volunteer performed Experiment A and B while the robots were controlled with one of following scheduling approaches ⁹

1. The fuzzy scheduler reported in Section 7.4.
2. Uniform scheduler: a scheduler similar to [14], which addresses the uncertainties related to the firing times of transitions in a more naive way, assuming the firing delays of the transitions as uniformly distributed. This scheduler represents an attempt to adapt [14] to the approach of Section 7.4. A brief description of the uniform scheduler is provided in Section 7.5.3.

For the experiments, T_{sch} was set equal to 50 s, in order to have at an average 4 events happening within the scheduling horizon. A crisp value of 5s was assumed for t_w (see Section 6.3.1).

⁹5 volunteers performed the experiments with a scheduler while the remaining with the other one.

Fuzzyfication of the activity durations

The distributions d , modelling the firing delays are updated according to new samples collected during time (see Section 7.1). Such samples have to undergo a fuzzyfication process. The problem amounts essentially to find the best triangular fuzzy set describing a quantity for which a restricted set of samples $\{D_1, \dots, D_M\}$ are available. Many methods have been proposed in literature to address this problem. There is no objective way to evaluate the goodness or correctness of such methods and mostly the choice of the method depends on the kind of problem to handle and on the type of data available. The method proposed in [2] was chosen. The best asymmetrical triangular set $\langle T_1, T_M, T_2 \rangle$ approximating samples $D_{1, \dots, M}$ is computed as follows:

$$\begin{aligned} T_M &= \frac{1}{M} \sum_{j=1}^M D_j \\ T_1 &= \max(0, T_M - \sigma_L) \\ T_2 &= T_M + \sigma_R \end{aligned} \quad (7.19)$$

where σ_L and σ_R of the above equations are computed as follows:

$$\begin{aligned} S &= \frac{\int_{-\infty}^{T_M} f(s) ds}{\int_{T_M}^{+\infty} f(s) ds} \\ \sigma_L &= \sigma \frac{S}{1 + S} \\ \sigma_R &= \sigma \frac{1}{1 + S} \end{aligned} \quad (7.20)$$

$f(d)$ is the empirical probability distribution of samples in D , while σ is their standard deviation.

Results

The idle times measured during the experiments are reported in Figure 7.27. By a qualitative analysis, the performance obtained when applying the fuzzy scheduler seems to be better. This is confirmed by a single-tailed Wilcoxon rank sum returning an $r = 0.0023$ when comparing Experiments A with the two scheduling approaches tested, while returns an $r = 0.0251$ when comparing Experiments B. The aforementioned tests were performed with an $\alpha = 5\%$.

The productivity obtained during the experiments was also estimated, by considering the time elapsing from the exiting of a finite product and the subsequent one. Such cycle times are reported in Figure 7.28. Also in this case, a clear statistical evidence confirms that the fuzzy approaches perform better: Wilcoxon test returns $r = 0.065$ when considering Experiments A, while $r = 0.122$ comparing Experiments B.

Not surprisingly, the fuzzy approach overcame the other one, which seems to be not suited for properly addressing the variability of the activity durations (since simple uniform distributions are assumed).

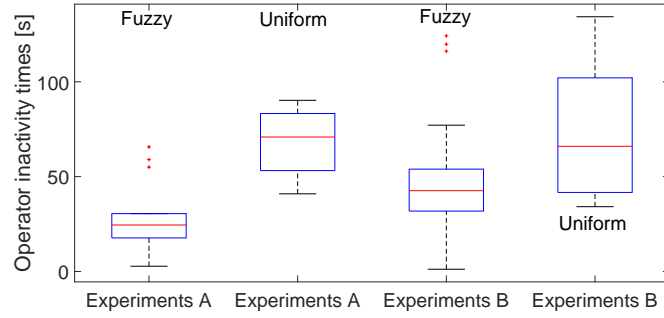


Figure 7.27: *Idling times of the operator. Every sample of the reported distributions refers to the idling measured within the assembly of a single product.*

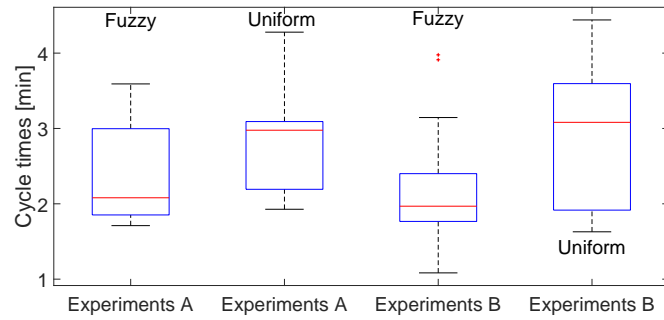


Figure 7.28: *Measured cycle times.*

Uniform scheduler

The uniform scheduler replaces all the triangular firing delays d with trapezoidal fuzzy numbers d' having $\alpha_0(d') = [\underline{t}, \bar{t}]$ and $\alpha_1(d') = [\underline{t} + \varepsilon, \bar{t} - \varepsilon]$, with ε that is a positive sufficient small constant. d'_j is defined such that:

$$\alpha_0(d') = [\underline{t}, \bar{t}] = \alpha_1(d) \quad (7.21)$$

The construction of the RT as well as the way the best plan is selected (Section 7.4 and 7.4.1) are identical to the fuzzy scheduler described so far, with the only two following differences. The computation of the generic arrival time α_{C_i} is altered, assuming α_{C_i} as that trapezoidal number for which:

$$\alpha_0(\alpha_{C_i}) = [\underline{t}, \bar{t}] = \alpha_0(\alpha'_{C_i}) \quad (7.22)$$

while W_H is computed as follows:

$$W_H = \varphi_1 \cdot \varphi_3 \quad (7.23)$$



Part III

Motion control of cobots

CHAPTER 8

Safe control of cobots

All the scheduling approaches discussed in Chapter 7 were based on a prediction of the human activities. Indeed, after predicting the operator's future behaviour, the proper assistive actions were scheduled. In a certain sense, the plan imposed to the system, is induced by the human (see the pipeline at the bottom of Figure 6.1). Indeed, the produced plan is tailored for a specific human since the system is indirectly adapting to the behaviour of that particular operator.

It seems natural to endow also the motion controllers with similar adaptation capabilities. In this way, observing the human during time, it is possible to forecast his or her future motion and then adopt the proper corrective actions for the robots motion. The approaches proposed in this Section have exactly this aim. In particular, the approach of Section 8.1 is a reactive one, while the one proposed in Section 8.2 is proactive.

The majority of the approaches developed in the past are classifiable as reactive: the robot is slowed down along its nominal path or it is forced to undertake local dodging manoeuvres, in case imminent collisions with the human mate are detected. The motion of the robot is typically a result of a pure closed loop control scheme. One example is the strategy described in [36], where a repulsive field deforms the trajectory of the robot, in order to let it accomplish its task, but in a safe way for the human. A similar approach was adopted in [107], where the accelerations of the robot are modulated according to a scaling factor, computed by solving a constrained optimization problem. The aim is to modify as little as possible the initial path, to guarantee the satisfaction of a safety constraint. In [18], the previous approach was extended, to tackle the problem with a model predictive control perspective, optimizing the accelerations on a longer time horizon.

The aim of a proactive approach is instead to compute some trajectories for the robot, designed to reduce in advance the risk of collision with humans, without waiting for

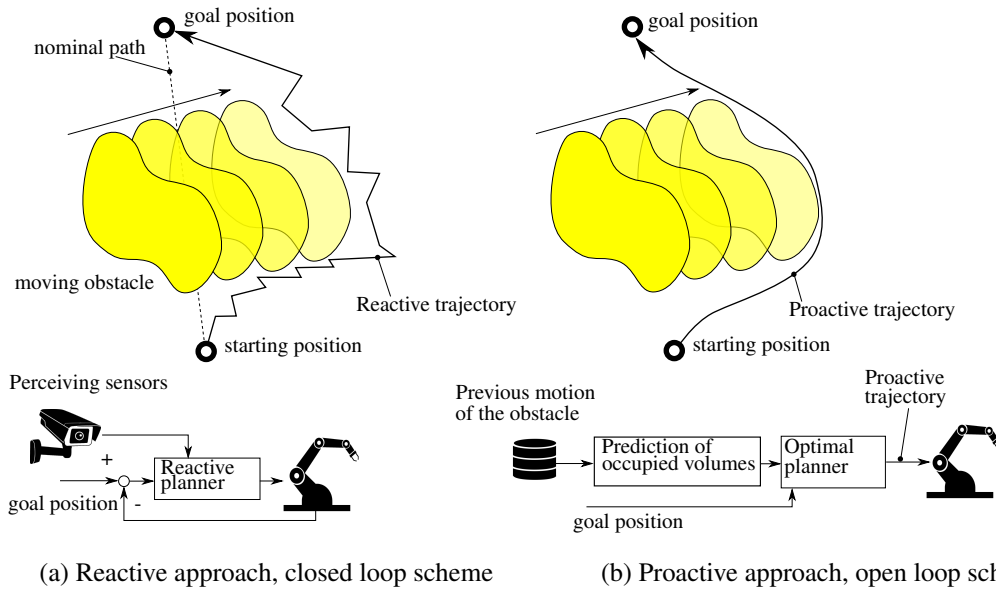


Figure 8.1: *Difference between a reactive approach and a proactive one. The motion resulting from a reactive approach guarantees the absence of collisions with a moving obstacle, but it's not globally optimized, since with a certain frequency is constantly recomputed considering the current position of the obstacle. On the opposite, the proactive path is computed from its start to its end once; by taking into account a prediction of the entire motion of the yellow obstacle. A combination of the two approaches is also possible.*

the situation to become critical. Figure 8.1 resumes the differences between reactive and proactive approaches.

Few proactive path planners were proposed in the past. Examples are [58], [121] and [66]. [58] addressed the problem of co-planning, computing the trajectories for both a human and a mobile robot. The proposed approach is however able to plan only 2D planar paths. [121] is another example of a 2D planner, where optimal paths for a mobile cobot are computed considering not only the safety of the human mate but also the acceptance of the resulting motion, trying to avoid trajectories for which the robot goes out from the visibility cone of the human. [66] tackled the proactive planning problem for an articulated robot, even though the developed method relies on an off-line computation of the paths.

It is important to remark that the computation of proactive trajectories requires to estimate which human activities are likely to be executed simultaneously with a specific robotic one, *i.e.* predict the behaviour of the human in the near-far future. However, this allows a proactive planner to achieve a global optimality for the produced motion, since the whole path is optimized, while reactive approaches are typically locally optimal, since corrective actions are imposed to avoid in the near future the collisions.

Both the reactive and the proactive approaches here proposed, need to find a way to adequately model the human motion, which is however one of the most challenging problems in the context of HRC. It might in fact be desirable to fit a model suitable to make predictions, using the smallest possible training set. The last aspect is crucial if we consider that each human worker is physically different. Therefore, it is necessary to use a model that evolves during time, without the need to undergo a long, off-line, training phase. This leads to a short, almost zero, set-up time when there is a switch

between the people who interacts with the robot. The desire to use a small amount of training data strikes with the fact that the human motion models are often highly non linear and highly dimensional.

Possible choices are, for instance, linear dynamical systems with Gaussian process noise, first- and second-order Markov models, or auto-regressive (AR) models [94]. Switching Linear Dynamic System and hybrid dynamics can provide much richer classes of temporal behaviours [96], [53]. Nevertheless, they are computationally challenging to learn, and require large amounts of training data. A possible way to cope with high-dimensional data, but considering a reduced training sets, is to make use of Gaussian Processes (GPs). This is actually the approach followed by both the work detailed in Section 8.1 and 8.2. To be more precise, 8.2 made use of an evolution of the standard GP formulation, considering a latent variable model called Gaussian Process Dynamical Model (GPDM) [127].

The reactive and the proactive approaches can be also combined: a proactive path can be computed minimizing the probability of collisions with the human, but then its execution can be managed by a reactive motion controller, enforcing some additional safety constraints for handling an unexpected behaviour of the operator. This is similar to the combination of open loop controls with the closed loop ones: an open loop trajectory guarantee the global optimality, while the feedback scheme provide robustness.

8.1 The reactive approach

The reactive approach proposed in [13] modulates the robot speed along an assigned path, according to a prediction of the volume that will be occupied by the human in the imminent future. It is based upon the definition of Swept Volumes: portions of the environment, entirely containing the human predicted motion. Swept Volumes are treated like obstacles to avoid. Predicting in a reliable way the human motion is therefore critical for building such containing volumes.

[13] was inspired by [107], which actually introduced the concept of Swept Volumes. However, in [107] the motion of the human was predicted with a simple Kalman filter, offline tuned. The resulting Swept Volumes were very conservative, assuming always the worst case scenario. On the other hand, in [13] adaptive Gaussian Processes (GPs) (see Appendix E) were adopted for predicting the human motion. The model was adaptive in the sense that the parameters characterizing the model were constantly recomputed, according to recent samples for the human motion. In this way, less conservative Swept Volumes were produced.

8.1.1 Background about Swept Volumes generation

With the aim of computing the Swept Volumes, the human is treated like a manipulator, whose motion is described by a 12 d.o.f. kinematic model. $q = [q_{base} \ q_{armR} \ q_{armL}]$ is the vector describing the human posture. $q_{base} = [q_1 \ q_2 \ q_3 \ q_4]$ contains the position and orientation of the torso, while $q_{armR} = [q_5 \ q_6 \ q_7 \ q_8]$ describes the posture of the right arm of the operator, similarly $q_{armL} = [q_9 \ q_{10} \ q_{11} \ q_{12}]$ for the left arm. Refer to the left pictures of Figure 8.2.

Commercially available depth cameras are able to provide on-line the position of some skeletal points of interest (see Section 9.0.1 and Figure 9.1), which are used together with an inverse kinematic function to compute q , see [134] for further details. By considering some kinematic limitations for the human motion (maximal and minimal joint velocities and accelerations) it is possible to compute the reachable set of every joint, represented by $\Delta^+ = [\Delta_1^+ \ \cdots \ \Delta_{12}^+]$ and $\Delta^- = [\Delta_1^- \ \cdots \ \Delta_{12}^-]$. The latter quantities are, respectively, the maximal and minimal possible excursions that human articulations can undergo within a prediction time T starting from the current one. More formally, for the generic j^{th} joint:

$$\begin{cases} \Delta_j^+ = \min \Delta \quad s.t. \\ \Delta \geq 0 \quad \wedge \quad q_j(t + \delta t) < q_j(t) + \Delta \quad \forall \delta t \in [0, T] \\ \Delta_j^- = \min \Delta \quad s.t. \\ \Delta \geq 0 \quad \wedge \quad q_j(t + \delta t) > q_j(t) - \Delta \quad \forall \delta t \in [0, T] \end{cases} \quad (8.1)$$

Δ^+ and Δ^- are considered to compute the swept volumes, by following the method adopted in [123]. Such shapes are treated like obstacles that need to be avoided, thus ensuring the human safety.

The method proposed in [107] is effective in guaranteeing the safety, but is at the same time very conservative when computing Δ^+ and Δ^- , since the kinematic limitations considered are not tailored to specific subjects. The aim of [13] was instead to exploit

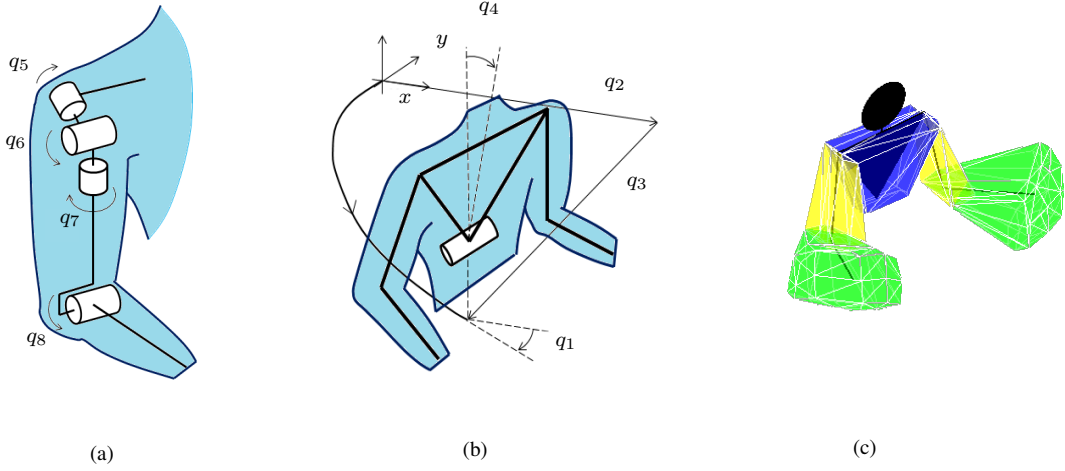


Figure 8.2: On the left, (a) and (b), the kinematic model adopted to describe the human posture. For the base of the human, a unicycle model is considered, while for the arms a spherical joint is centred at the shoulder, with an additional joint located at the elbow. On the right, (c), an example of swept volumes bounding the future occupancy of the anatomical parts of the human. In blue the volume swept by the torso, in green those for the forearms, while in yellow those for the upper arms.

Gaussian Processes to forecast the evolution of q in order to compute Δ^+ and Δ^- in a less conservative way.

8.1.2 Gaussian Processes for Swept Volumes generation

Gaussian Processes can be exploited to compute customized models of human motion, tailored to specific subjects. Every operator present in a cell is assumed to be monitored by a surveillance camera, able to provide an estimate of the operator's posture q at a certain sampling frequency $1/\Delta t$. Let ϕ_k denote the state of the operator's articulations at step k :

$$\phi_k = [\ddot{q}_k^T \quad \dot{q}_k^T \quad q_k^T]^T \quad (8.2)$$

ϕ_k can be estimated applying a Kalman filter on a linear kinematic model made of a chain of three integrators:

$$\begin{aligned} \phi_{k+1} &= A \phi_k + w \\ z_{k+1} &= q_{k+1} = [0 \quad 0 \quad I] \phi_k + e \\ A &= \begin{bmatrix} I & 0 & 0 \\ \Delta t I & I & 0 \\ 0.5 \Delta t^2 I & \Delta t I & I \end{bmatrix} \end{aligned} \quad (8.3)$$

where w and e are the process and the measurement noise respectively, which are assumed to be Gaussian. The classical forward formulation of the Kalman filter is able to provide recursively $\phi_{k|k}$, *i.e.* the mean of the posterior distribution of the state ϕ_k , conditioned to all the measures seen in the past z_1, \dots, z_k . In [107], the estimate $\phi_{k|k}$, is taken into account for computing the reachable sets of every q_i , by assuming the accelerations \ddot{q} and the velocities \dot{q} are bounded. The aforementioned sets are then considered

for the swept volumes computation. However, the kinematic limitations adopted are not tailored to specific subjects, but worst case values are used.

In collaborative contexts we can assume the human motion to have some kind of periodicity, since the human and the robot have to alternatively accomplish a certain number of operations (refer to assembly flows in Section 7.5.1, 7.5.2 and 7.5.3). In similar scenarios the worst case approach of [107] produces unnecessary conservative predictions. On the opposite, the aim of [13] was to model the aforementioned periodicity, to provide more accurate medium-term predictions. Such predictions cannot be extracted from the purely linear model expressed in equation (8.3), which can be only adopted for filtering the observations. However, the estimates $\phi_{k|k}$ provided by the Kalman filter are exploited to refine a Gaussian Process able to efficiently describe the human motion. An adaptive approach is used, starting from an initial model and continuously refining it at a certain frequency. Let h be the function governing the periodicity of the human motion. The definition of h is:

$$\phi_k = h(\phi_{k-1}, \dots, \phi_{k-P}) \quad (8.4)$$

where P is the order of the model. h is clearly unknown, but can be approximated by making use of a Gaussian Process h_{GP} (see equation (E.7) and (E.18)):

$$h(\phi_{k-1}, \dots, \phi_{k-P}) \doteq h_{GP}(\phi_{k-1}, \dots, \phi_{k-P}) \quad (8.5)$$

Three different possible ways to build h_{GP} will be proposed in the following. As discussed in Appendix E, Gaussian Processes rely on samples. In this case we can use as samples for h_{GP} , not the on-line computed estimates $\phi_{k|k}$, but $\phi_{k|K}$, *i.e.* the mean of the posterior distribution of ϕ_k conditioned to all measures in a window: $z_{1, \dots, ki, \dots, kf}$.

Therefore, the proposed approach periodically updates h_{GP} , reasoning on the most recent available window of observations. The frequency of the model update is clearly lower than $1/\Delta t$, *i.e.* the sampling time adopted for updating the human state, which is also the sampling time adopted for updating the swept volumes Δ^+ and Δ^- . At every step k , Δ_k^+ and Δ_k^- are recomputed on the basis of the most recent learnt h_{GP} .

The computation of $\phi_{k|K}$ is made applying the backward formulation of the Kalman filter (see [91] at Section 3.6.1), on the window ki, \dots, kf , leading to:

$$\begin{aligned} \phi_{k+1|k} &= A\phi_{k|k}; & V_{k+1|k} &= AV_{k|k}A^T + W \\ J_k &= V_{k|k}A^T V_{k+1|k}^{-1} \\ \phi_{k|K} &= \phi_{k|k} + J_k(\phi_{k+1|K} - \phi_{k+1|k}) \end{aligned} \quad (8.6)$$

where W is the covariance of w , while $V_{k|k}$ is the covariance of the posterior distribution of ϕ_k , conditioned to the measures $z_{1, \dots, k}$. Both $\phi_{k|k}$ and $V_{k|k}$ are computed on-line during the canonical application of the Kalman filter, therefore those quantities are already known when updating the model. Expressions in equation (8.6) are applied recursively backwards in time. The estimates $\phi_{ki|K, \dots, kf|K}$ are manipulated in a proper way, to obtain new samples for enriching the training set of h_{GP} .

For all the approaches the following kernel function (see Appendix E) is assumed:

$$\begin{aligned}
 K &= \begin{bmatrix} k_{11} & \cdots & k_{1N} \\ \vdots & \ddots & \vdots \\ k_{N1} & \cdots & k_{NN} \end{bmatrix} \\
 k_{ij} &= \theta_1 X^{iT} X^j + \theta_3 \exp\left(-\frac{\theta_2}{2} \|X^i - X^j\|_2^2\right) + \theta_4 \delta_{ij} \quad (8.7)
 \end{aligned}$$

where δ_{ij} is the Kronecker delta.

The way h_{GP} can be adopted for computing human swept volumes is now detailed. The computation of Δ_k^+ is made considering a prediction horizon of a certain length M , the computation of Δ_k^- is similar. Since GPs are stochastic models, the vector Δ_k^+ is actually a collection of probability distribution functions. Clearly, characterizing in a closed form such distributions is almost impossible. On the opposite, that distributions can be approximated by empirical ones, taking into account a set of samples $\{\tilde{\Delta}_1^+, \dots, \tilde{\Delta}_T^+\}$. These samples are obtained by simulating possible future human motions. More formally, a Monte Carlo simulation consisting of T trial can be deployed. Every trial starts from $\phi_{k|k}$ and propagates it M times by using h_{GP} , obtaining a sampled trajectory Q_t . For the t^{th} trial, $\tilde{\Delta}_t^+$ is computed considering the maximal excursion reached by every joint in that trial. More formally, the required steps are done in this way:

$$\begin{aligned}
 Q_t &= [q_{k|k} \quad \tilde{q}_{k+1} \quad \tilde{q}_{k+2} \quad \cdots \quad \tilde{q}_{k+M}] \\
 \tilde{\phi}_{k+l} &\doteq h_{GP}(\tilde{\phi}_{k+l-1}, \dots, \tilde{\phi}_{k+l-P}) \quad \forall l = 1, \dots, M \\
 \tilde{\Delta}_{j,t}^+ &= \max\left(0, \max(q_{j,k}, \tilde{q}_{j,k+1}, \dots, \tilde{q}_{j,k+M}) - q_{j,k}\right) \\
 \tilde{\Delta}_t^+ &= [\tilde{\Delta}_{1,t}^+ \quad \cdots \quad \tilde{\Delta}_{12,t}^+]^T \quad (8.8)
 \end{aligned}$$

In the above equation every $\tilde{\phi}_{k+p}$ (with p negative or null) is assumed to be equal to $\phi_{k+p|k+p}$. Similar considerations hold for computing the distributions describing Δ_k^- . Swept volumes are built considering certain percentiles (one for every joint in the kinematic model) for Δ_k^+ and Δ_k^- . Here, a compromise between the risk to have a collision and the productivity of the cobot to control must be done. Indeed, selecting high values for the percentiles leads to the computation of conservative bounding volumes, containing with a very high probability the future motion of the human. However, accepting a certain low risk of collision, i.e. the 3%, could significantly reduce the sizes of the volume describing the human occupancy, allowing to move the robot along its path with an increased speed. Figure 8.3 summarizes the above considerations.

Characterizing the exact relationship between the risk of a collision and the percentiles of Δ_k^+ and Δ_k^- can be a matter for future studies, while in [13], the mean values of the

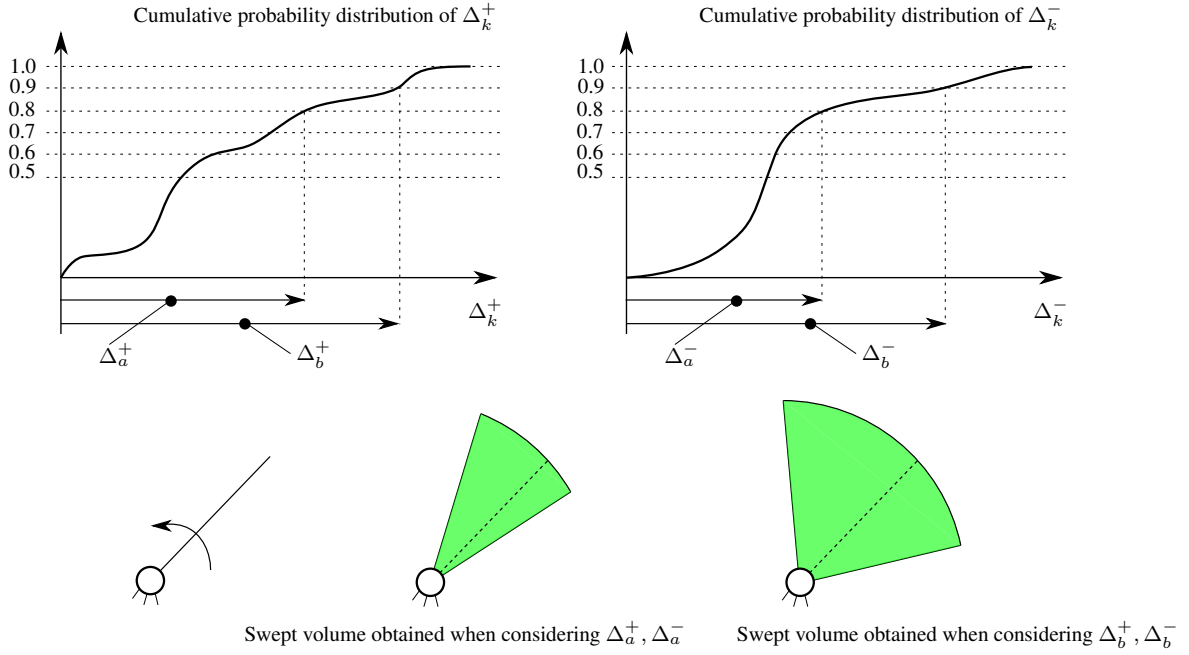


Figure 8.3: On the top, examples of cumulative distributions for Δ_k^+ and Δ_k^- (the kinematic chain is assumed to be made of a single joint), while on the bottom examples of swept volumes for a single joint mechanism, depicted on the left bottom corner, when assuming different percentiles for Δ_k^+ and Δ_k^-

empirical distributions were assumed for the swept volume generation:

$$\Delta_k^+ = \frac{1}{T} \sum_{t=1}^T \tilde{\Delta}_t^+ \quad (8.9)$$

$$\Delta_k^- = \frac{1}{T} \sum_{t=1}^T \tilde{\Delta}_t^- \quad (8.10)$$

The entire pipeline of the approach is reported in Fig. 8.4.

Model A, decentralized approach

The first possible model makes 12 independent predictions, one for each human articulation. Indeed, for every joint only the previous accelerations, velocities and positions are considered to make a prediction about the future trajectory. This is equivalent to have 12 independent GPs assembled together to form h_{GP} . For notational purposes, let φ^j denote the state of the j^{th} joint:

$$\varphi^j = [\ddot{q}_j \quad \dot{q}_j \quad q_j]^T \quad (8.11)$$

The GP modelling the evolution in time of φ^j is a function:

$$\varphi_k^j \doteq h_{GP}^j(\varphi_{k-1}^j, \dots, \varphi_{k-P}^j) \quad (8.12)$$

The comprehensive prediction about ϕ is obtained by assembling in the proper way every predicted φ^j .

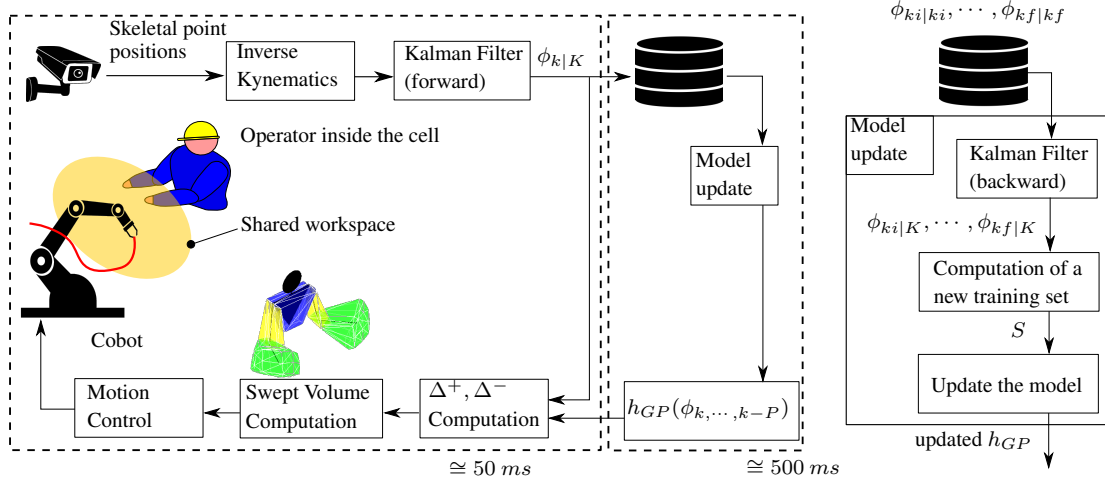


Figure 8.4: Pipeline describing the developed approach. The picture on the right details the steps involved for updating h_{GP} , i.e. the model describing human motion.

Every h_{GP}^j is updated separately, adding to the training set, a set S of new samples defined as follows (see Appendix E):

$$\left\langle \dots, \left[\begin{array}{c} X^l = \begin{bmatrix} \varphi_{k-1|K}^j \\ \vdots \\ \varphi_{k-P|K}^j \end{bmatrix} \\ \hline Y^l = \varphi_{k|K}^j \end{array} \right], \dots \right\rangle \quad (8.13)$$

Model B, centralized approach

The decentralized model introduced before is able to express the periodicity of human motion. However, such motion usually evolves in a coordinated way among the articulations. For this reason, it seems more natural to consider a model constituted by a single GP, that takes into account the past evolution of the entire state ϕ when making a prediction for every joint.

$$\phi_k \doteq h_{GP}(\phi_{k-1}, \dots, \phi_{k-P}) \quad (8.14)$$

Notice that in this case the computational effort required to update the model and compute Δ^+ , Δ^- is significantly higher than a purely decentralized approach, since the complexity of GP scales quadratically with the cardinality of the domain of the function to approximate.

Model C, hybrid approach

The centralized model introduced before is meant to represent the coordination existing between the articulations. However, it is also computationally heavy. Moreover, the actual coordination governing human motion involves groups of joints. For example, when the operator moves his/her arms, joints of the left arm are moved independently from the ones of the right. For this reason, we can think of a third kind of model that

represents separately three main groups of articulations: the moving base, the right arm and the left one. α , β and γ denotes the state of the three groups of joints:

$$\alpha = \begin{bmatrix} \ddot{q}_{base\ k} \\ \dot{q}_{base\ k} \\ q_{base\ k} \end{bmatrix}; \quad \beta = \begin{bmatrix} \ddot{q}_{armR\ k} \\ \dot{q}_{armR\ k} \\ q_{armR\ k} \end{bmatrix}; \quad \gamma = \begin{bmatrix} \ddot{q}_{armL\ k} \\ \dot{q}_{armL\ k} \\ q_{armL\ k} \end{bmatrix} \quad (8.15)$$

The prediction about ϕ is the assembly of three independent predictions:

$$\alpha_k \doteq h_{GP}^\alpha(\alpha_{k-1}, \dots, \alpha_{k-P}) \quad (8.16)$$

$$\beta_k \doteq h_{GP}^\beta(\beta_{k-1}, \dots, \beta_{k-P}) \quad (8.17)$$

$$\gamma_k \doteq h_{GP}^\gamma(\gamma_{k-1}, \dots, \gamma_{k-P}) \quad (8.18)$$

h_{GP}^α , h_{GP}^β and h_{GP}^γ are updated considering three independent newer training sets.

8.1.3 Experiments

The experimental set-up adopted is reported in Figure 8.5. The MICROSOFT KINECT reported in the same Figure is exploited to perceive the posture of the human during the time. This information is sent to a CPU, which is also connected to YUMI. The robot speed is modulated as done in [134]. At every steps, the convex shapes representing the swept volumes are recomputed by the CPU connected to the robot. Such volumes are exploited for defining a constrained optimization problem whose solution is used for modulating the robot speed along its current assigned path. The aim is to compute the maximal speed to set for avoiding collisions with the aforementioned swept volumes (in case of critical circumstances, the robot motion can be also interrupted). Clearly, a less conservative computation of human swept volumes leads to reduced size obstacles to avoid, allowing a greater cruise speed.

A human operator and YUMI are required to simultaneously assemble two distinct products: the box with a USB pen described in Section 3.1.3 and the PCB board described in Section 7.2.1, respectively. The robot's arms perform the picking of the box and the caps of the PCB board simultaneously. They then assemble it by making use of the assembly station indicated by the picture in the middle of Figure 8.5. After the assembly of the box is completed, the newer piece is taken by the left arm of YUMI and put in a specific storage area near the right part of Figure 8.5 (the storage area is not visible in the same Figure).

The human autonomously performs all the assembly steps described in Section 3.1.3, taking the necessary parts from the buffer reported in Figure 8.5. The operator's trajectory are approximately reported in the right part of Figure 8.5.

As can be seen, both the agents have to cross a shared area. The locations of buffers were not selected in an ergonomic way, but the experimental set-up was designed mainly to show the validity of the developed approach, simulating a situation of co-existence for the human and the robot.

Six volunteers were enrolled for the experiments and were divided into two groups I and II. For those in the group I, the approach presented in [107] was applied ¹ for

¹Only for the way swept volumes are generated, but not the way the robot's motion is controlled, since the approach of [134] is adopted as clarified at the beginning of this Section.

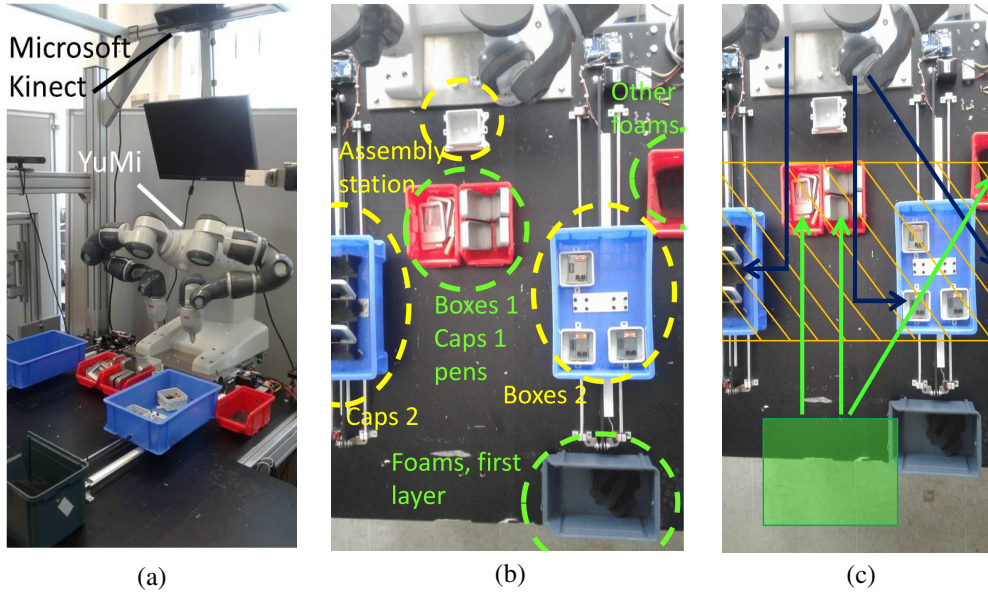


Figure 8.5: *Experimental set-up considered for the experiments. (a): YUMi of ABB, is visible in the centre; with a MICROSOFT KINECT vision system on top. (b): Locations of buffers containing the items required to assemble the box with the USB and the PCB board. (c): Some approximative trajectories followed by both the human (in clear green) and the robot (dark blue). The dashed orange area is the portion of space shared by the human and the robot, while the green shaded one is the portion of the space occupied by the operator's torso during the experiments.*

the computation of swept volumes adopting a basic linear kinematic model; while for those in group II, the hybrid approach described in Section 8.1.2 was considered (with $P = 4$). Section 8.1.3 will compare performance obtained by the two groups, while Section 8.1.3 presents the results of some off-line simulations, comparing the performance achieved by all methods proposed for modelling the human motion, on the data (position in time of the operators) acquired on-line during the real experiments.

Real experiments

The sample time of the depth camera was set to 50 ms. When applying the hybrid method, a newer Gaussian Process model for the three groups of joints is trained every 500 ms. Therefore, for every new training of the models, 10 newer samples are available. The training set has a fixed size equal to 100 samples. Newer samples replace 10 older ones every time an update of the model is invoked. Samples to replace are selected randomly with a uniform distribution in order to have a novel set with samples of both the recent and the remote past². Each of the volunteers in the groups I and II was required to accomplish 5 assemblies, while the robot simultaneously accomplished the assigned operations. The mean cycle time of the robot for the experiments of group II was about $8.59 \pm 1.3\%$ lower than the one related to experiments for group I, showing the beneficial effects of having a less conservative prediction of human motion. The mean time required to train a new model was 309.86 ms, with a standard deviation

²In principle, it could be possible to consider the trace of $V_{k|K}$ and sample position to replace accordingly, *i.e.* trying to remove samples with a higher uncertainty. However, after a transient phase, Kalman Filter reaches a steady state for which the trace of the estimation covariance stops to changing significantly. Therefore, that approach would end up to be equivalent to sampling with a uniform distribution.

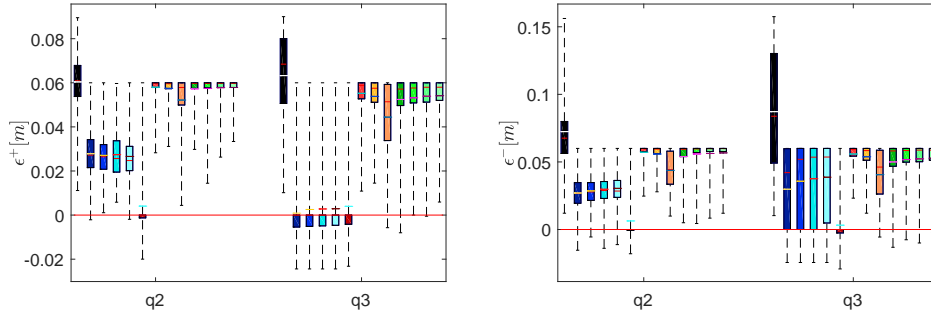


Figure 8.6: Distributions of ϵ^+ and ϵ^- , for joint q_2 and q_3 of the kinematic model reported in Figure 8.2. The legend of Figure 8.7 applies. The red horizontal line, divide conservative predictions from the ones for which a violation happens. Results for similar joints of both arms are condensed in a single boxplot.

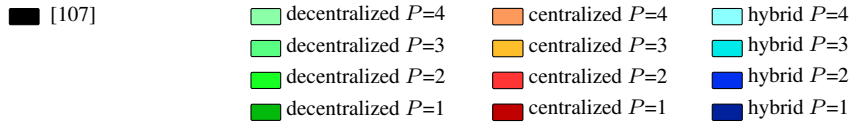


Figure 8.7: Legend to consider for Figure 8.6 and 8.8

equal to 153.07 ms, which is compatible with the frequency selected for updating h_{GP} .

Off-line simulations

For every sample of the data logged on-line during the experiments, it is possible to compute $\epsilon^+ = \Delta_{pred}^+ - \Delta_{real}^+$ and $\epsilon^- = \Delta_{pred}^- - \Delta_{real}^-$. $\Delta_{pred}^{+,-}$ are the excursions predicted for every human articulation using h_{GP} , with the different methods described so far; while $\Delta_{real}^{+,-}$ is the real excursion followed. The excursions $\Delta_{real}^{+,-}$ are computed according to the estimates $q_{1|1}, \dots, q_{N|N}$ retrieved on-line by the Kalman filter during the experiments and saved for this off-line processing phase. Notice that it is a sufficient and necessary condition to require both $\epsilon^+ \geq 0$ and $\epsilon^- \geq 0$ for guaranteeing that the predicted excursion entirely contains the real followed trajectory. Moreover, the more ϵ^+ is positive and large, the more conservative the prediction was (the same applies for ϵ^-). Figures 8.6 and 8.8, report some statistics about ϵ^+ and ϵ^- , on the data of the experiments. As can be seen, the approach in [107] produces predictive excursions that almost always contain the real excursions of joints, even though the approach turns out to be very conservative, since $\epsilon^{+,-}$ are positive and big. On the opposite, the methods exploiting any kind of Gaussian Process as model for the human motion, result to produce lower $\epsilon^{+,-}$ which can be also negative sometimes. In particular, the hybrid approach of Section 8.1.2 seems to be the one producing predictions with the best compromise between conservativeness and risk of violation. Surprisingly, in both in Figure 8.6 and 8.8 no significant differences can be detected when considering a specific method with different values for the order P . For the hybrid approach of Section 8.1.2, Figure 8.10 reports also the distance of the real positions of wrists, elbows and shoulders to the predicted swept volumes, considering only those samples for which $\epsilon^+ \leq 0$ or $\epsilon^- \leq 0$.

8.1. The reactive approach

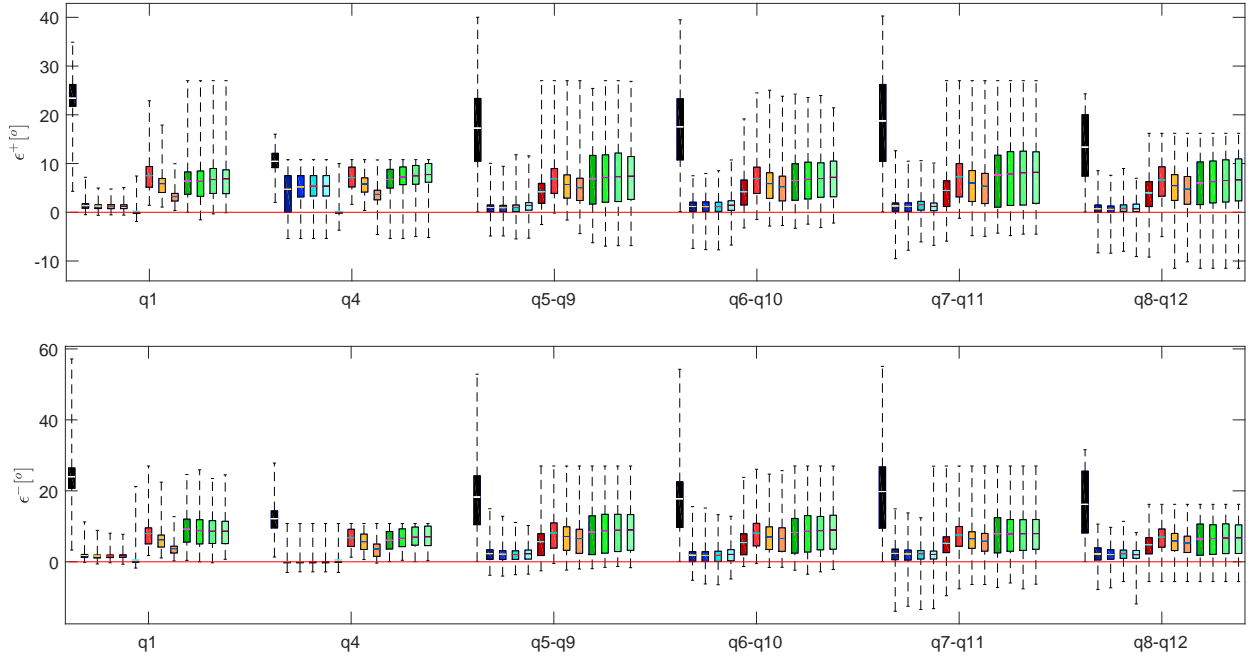


Figure 8.8: Distributions of ϵ^+ and ϵ^- , for joints $q_1, q_4, q_5, q_6, q_7, q_8, q_9, q_{10}, q_{11}, q_{12}$ (i.e. the rotating ones) of the kinematic model reported in Figure 8.2. Results for similar joints are condensed in a single boxplot. The same legend in Figure 8.7 applies.

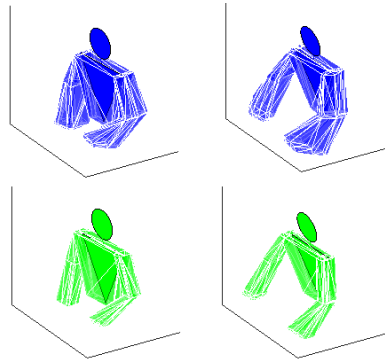


Figure 8.9: Comparisons between the swept volumes obtained considering [107] (the blue ones on the first row) against those obtained applying the hybrid approach with $P = 4$ (the green ones on the second row), for two sampled instants of the logged data.

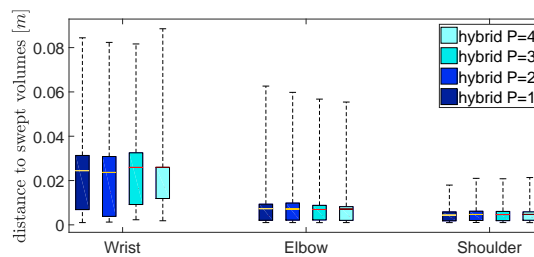


Figure 8.10: Statistics about the distances between skeletal points and predicted swept volumes, in case of violation of the predicted joint excursion.

As can be seen from the analysis of Figure 8.10, the violation of the predicted swept volumes by the real trajectory followed by skeletal points is in the order of few centimeters. Figure 8.9 reports also a visual comparison of some swept volumes computed with [107], w. r. t. the ones computed by the hybrid approach with a $P = 4$.

8.2 The proactive approach

The computation of proactive paths require to model the interaction between the operator and each robot.

8.2.1 Modelling the human-robot collaboration

Consider the co-assemblies described in Section 7.5.1, 7.5.2 or 7.5.3: a certain number of operations are assigned to the robots and others are undertaken by the human. Without loss of generality, from now on the workspace is assumed as populated by a single cobot and a single human mate.

The set $\mathcal{A}_S = \{a_{S1}, \dots, a_{Sn}\} \subseteq \mathcal{A}$ (see the introductory Section of I) contains those actions requiring the operator to traverse the space shared with the robot. Considering the robotic agent, set $\mathcal{R} = \{r_1, \dots, r_m\}$ contains those actions requiring the cobot to traverse the shared area.

For every $r_j \in \mathcal{R}$, a corresponding nominal path Q_j is preassigned. Q_j ensures the completion of r_j , but ignoring the motion that the human mate will simultaneously perform. It is described by a series of intermediate waypoints q_{jk} 's. In terms of joint configurations:

$$Q_j = [q_{j0} \quad q_{j1} \quad \dots \quad q_{jN-1} \quad q_{jN}] \quad (8.19)$$

The objective of the proactive approach in [12] was to compute for every r_j , a proactive path Qp_j , modifying the waypoints of Q_j , in order to minimize a certain global cost $\mathcal{J}(Qp_j)$ (see also Section 8.2.3). The shape of \mathcal{J} is designed to obtain a Qp_j as much as possible similar to Q_j , but for which the probability of a collision with the human is minimized (see again Section 8.2.3). On the opposite, a pure reactive approach would be only able to locally deform a trajectory, executing on-line some near range corrective actions, obtaining most of the time a suboptimal solution, see the initial part of this Chapter. To be more precise, an adaptive approach was adopted, periodically recomputing proactive paths, according to the most recent data describing the motion of the operator.

To obtain such a description, the trajectories performed by the operator while executing actions in \mathcal{A}_S (which are basically sequence of postures), are retrieved from a depth camera and segmented on-line (possibly using the approach proposed in Chapter 4). Each acquired realization of a certain a_{Si} is saved as an additional sample in a specific database (see Figure 8.11). These samples are then exploited to obtain a probabilistic description of the space occupied by the operator, which is in turn used by the proactive planner (see Section 8.2.2 and Section 8.2.3).

\mathcal{P}_{ji} will be used to indicate the conditional probability that the human is doing a certain action a_{Si} while the robot simultaneously executes a particular action r_j . $A(t)$ will describe the actions performed during time by the human. $A(t)$ takes values in a discrete set $\{0, S1, \dots, Sn\}$, according to what action in \mathcal{A}_S the human is doing for time t . A

value equal to 0, indicates that the human is not doing any of the actions in \mathcal{A}_S , because he is doing either nothing or an action in $\mathcal{A} \setminus \mathcal{A}_S$. $R(t)$ is a similar function defined considering actions in \mathcal{R} , *i.e.* those of the robot. \mathcal{P}_{ji} is a time varying quantity, defined as follows:

$$\mathcal{P}_{ji}(t) = \mathbb{P}(A(t) = i | R(t) = j) \quad (8.20)$$

The computation of $\mathcal{P}_{ji}(t)$ relies on $I_r(t)$ and $I_h(t)$, which are two indicator functions defined as follows:

$$\begin{aligned} I_r(t) &= (I_{r1}(t) \ \cdots \ I_{rm}(t))^T \\ \text{such that } I_{rj}(t) &= \begin{cases} 1, & \text{if } R(t) = j \\ 0, & \text{otherwise} \end{cases} \end{aligned} \quad (8.21)$$

Such a function indicates whether the robot, at time t , is executing r_j . $I_h(t)$ is similarly defined, considering $A(t)$, hence the human. Every \mathcal{P}_{ji} can be computed considering a sliding temporal window of length T :

$$\begin{aligned} P_{RH}(t) &= \begin{bmatrix} \mathcal{P}_{11}(t) & \cdots & \mathcal{P}_{1n}(t) \\ \vdots & \ddots & \vdots \\ \mathcal{P}_{m1}(t) & \cdots & \mathcal{P}_{mn}(t) \end{bmatrix} = \\ &= M \cdot \int_{t-T}^t I_r(\tau) \cdot I_h(\tau)^T d\tau \end{aligned} \quad (8.22)$$

$$M = \begin{bmatrix} \frac{1}{\int_{t-T}^t I_{r1}(t) dt} & 0 & \cdots & 0 \\ 0 & \frac{1}{\int_{t-T}^t I_{r2}(t) dt} & 0 & \vdots \\ & & \ddots & \\ 0 & \cdots & 0 & \frac{1}{\int_{t-T}^t I_{rm}(t) dt} \end{bmatrix} \quad (8.23)$$

Even when considering a big set \mathcal{A}_S , distributions in P_{RH} will have a low entropy, if we assume that the human actions follow a certain pattern (which for the cases addressed in this work is reasonable, see Chapter 5). Indeed, few human actions would be likely to be executed simultaneously to a specific robotic one.

8.2.2 Probabilistic description of the human motion

The starting time of a specific action a_{Sj} is a random variable having high variability [98]. For this reason, it is better to adopt a probabilistic time-independent description. At a first stage, a trajectory Φ_j is obtained exploiting the samples acquired for a_{Sj} . Φ_j is then employed to compute a probabilistic point cloud C_j describing the volume occupied by the human while executing a_{Sj} (see the left part of Figure 8.11). The computation of a single Φ is detailed in Section 8.2.2, while the steps involved in the determination of the corresponding C are reported in 8.2.2.

Modelling the human trajectories

GPDM [127] can be exploited for learning the human motion from samples. In particular, GPDM is a latent variable model with a non-linear probabilistic mapping from a

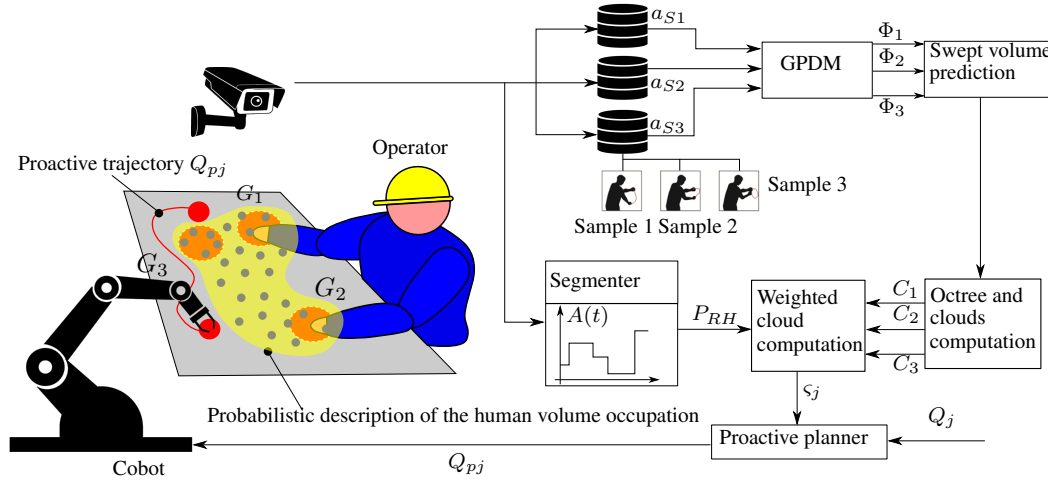


Figure 8.11: The entire pipeline of the approach. A depth camera records the motion of the human, whose trajectories are then segmented and stored in different databases, one for each human action. Such samples are considered by GPDM to compute regressed trajectories for the human motion, which in turn are exploited to compute the corresponding probability clouds and the proactive paths.

latent space \mathcal{X} to an observation one \mathcal{Y} and a non-linear dynamics evolving in the latent space. Every element \mathcal{Y} is a collection of information describing the human posture, as for example the position in the space of some points of interest like the elbows, the wrists, etc. The training set consists of a collection of acquired trajectories:

$$\left\langle \begin{bmatrix} \Phi_1^1 \\ \vdots \\ \Phi_{L1}^1 \end{bmatrix}, \dots, \begin{bmatrix} \Phi_1^N \\ \vdots \\ \Phi_{LN}^N \end{bmatrix} \right\rangle \quad (8.24)$$

for which the corresponding latent space sequences:

$$\left\langle \begin{bmatrix} X_1^1 \\ \vdots \\ X_{L1}^1 \end{bmatrix}, \dots, \begin{bmatrix} X_1^N \\ \vdots \\ X_{LN}^N \end{bmatrix} \right\rangle \quad (8.25)$$

are not known, and are therefore treated like additional hyperparameters (see Section appendice GP) to be learnt by the training process. The problem amounts essentially to learn two Gaussian Processes simultaneously: the first one approximating a function $g_{XX}(X)$ which describes the latent dynamics, and the second one approximating $g_{XY}(X)$, *i.e.* the mapping between \mathcal{X} and \mathcal{Y} .

The details about the learning steps are omitted, since are extensively discussed in [127] and they are not much different from the ones described in Appendix E.0.1, with the only difference that the unknown values assumed by the latent variables for the samples in the training set, must be computed, considering the gradient of the likelihood function w.r.t. to them.

The obtained GPDM, can be used to compute the regressed trajectory $\Phi = \langle \Phi_1, \dots, \Phi_L \rangle$, which models the human motion for a specific action. Indeed, an initial X_1 (which can be assumed as the mean of every starting X of the trajectories in the training set) is

propagated by making use of the learnt functions g_{XX} and g_{XY} , taking for every step their expectations, leading to:

$$\begin{aligned}\Phi_k &= \mathbb{E}[g_{XY}(X_k)] \\ X_{k+1} &= \mathbb{E}[g_{XX}(X_k)]\end{aligned}\tag{8.26}$$

Computing the probability cloud

The computation of Φ described in the previous Section is the starting point for the determination of the corresponding probability cloud C . C consists in a set of g points in the space with an associated probability of occupation:

$$\begin{aligned}C &= \left\{ \begin{bmatrix} p_{C1} \\ \gamma_1 \end{bmatrix}, \dots, \begin{bmatrix} p_{Cg} \\ \gamma_g \end{bmatrix} \right\} \\ p_{C1, \dots, g} &\in \mathbb{R}^3, \quad \gamma_{1, \dots, g} \in [0, 1]\end{aligned}\tag{8.27}$$

C is computed according to the volume swept by the operator when executing Φ . We are not interested in describing the volumes swept by the entire body of the operator, but we can limit our analysis to the motion of the forearms and arms, since those parts are typically the ones that effectively share the space with the robot. Moreover, without loss of generality, it is assumed that every action a_{Sj} is performed by moving one single arm.

The exact computation of the swept volumes of an articulated mechanism is a computationally demanding problem. However, an approximate approach is adopted, considering the trajectories of the skeletal points (Figure 9.1) between two subsequent poses Φ_k, Φ_{k+1} as approximately linear.

This approximation is acceptable when considering a trajectory Φ for which two consecutive poses are very close, which is true if the samples in the training set are acquired with a sufficiently high frequency (commercially available depth cameras are able to provide samples at about 30 Hz). The volume swept between two intermediate poses Φ_k, Φ_{k+1} for the forearm or the arm, is assumed to be the Minkowski sum of a sphere³ and a polytope whose vertices are the positions of the involved skeletal points (wrist and elbow or elbow and shoulder) for k and $k+1$. Therefore, such volumes have convex shapes, that can be approximated by using OctTrees [82] (see Figure 8.12). The latter approximation is useful for computing in a faster way the probabilistic cloud C . Indeed, the computation of C is made considering a discretized 3D grid of g equally spaced elements. Such set contains the nodes of the grid obtained by partitioning into G parts every dimension of the smallest oriented bounding box entirely containing the set of volumes swept when performing Φ (refer to Figure 8.12). Clearly, it holds that $G = g^3$.

For every pair Φ_k, Φ_{k+1} , a vector c_k of g elements must be computed. Every element in c_k can be equal to 1 or 0, depending on the circumstance that the corresponding point in the grid is contained or not by the volumes swept from k to $k+1$. The computations are speeded up considering the approximating OctTree: the points in the grid contained in every leaf of the tree can be easily extracted and the relative value in the binarized vector c_k is set equal to 1. The occupancy probabilities γ are computed as the mean

³The radius of the sphere is chosen by considering the typical anatomical size of human arms.

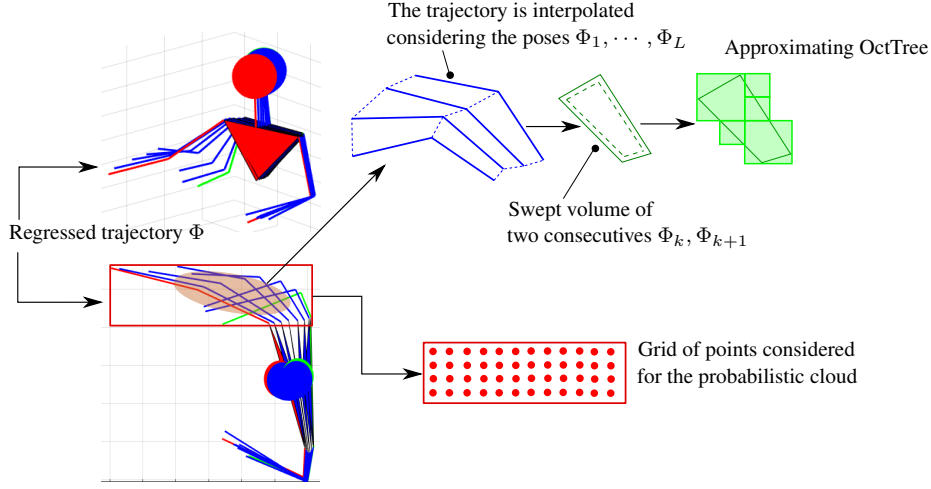


Figure 8.12: Pictures on the left depict an example of regressed trajectory taken from two distinct views. The bounding box containing entirely the trajectory (red box in the figure) is considered for the definition of the g points contained in C . The volume swept by the arm of the operator, between two consecutive poses, is approximated by the dark green convex set in the middle, which in turn is approximated by an OctTree, indicated in the right with light green (only the OctTree of the upper part of the arm is reported in the figure).

value of c along the trajectory Φ :

$$\gamma = \begin{pmatrix} \gamma_1 \\ \vdots \\ \gamma_g \end{pmatrix} = \frac{1}{L-1} \sum_{k=1}^{L-1} c_k \quad (8.28)$$

where L is the number of poses contained in Φ . Those points for which the corresponding γ would result to be 0 or below a certain threshold, can be discarded from the clouds.

8.2.3 Proactive path planning

For every action $a_{Sj} \in \mathcal{H}$, a probabilistic cloud C_i is computed by following the steps detailed in the previous Section. When computing the proactive path associated to the generic robot action r_j , a corresponding cloud ς_j must be considered. ς_j is a weighted union of all the clouds $C_{1,\dots,n}$ describing the volume occupancy for every human actions:

$$\varsigma_j = \{\{\mathcal{P}_{j1} \odot C_1\} \cup \dots \cup \{\mathcal{P}_{jn} \odot C_n\}\} \quad (8.29)$$

where the operator \odot applies as follows:

$$w \odot C = \left\{ \begin{bmatrix} p_{C1} \\ \gamma_1 \cdot w \end{bmatrix}, \dots, \begin{bmatrix} p_{Cg} \\ \gamma_g \cdot w \end{bmatrix} \right\} \quad (8.30)$$

This is done for considering not only the volume occupied by the human when executing specific actions, but also the probability that he/she is doing those actions simultaneously to r_j . Once ς_j is available, the proactive path Q_{pj} is computed as the

result of an optimization problem, whose cost function \mathcal{J} balances the risk of collisions with the human and the need to alter as little as possible Q_j . Figure 8.11 summarizes the entire pipeline of the approach.

The points in the resulting cloud ς_j , induce a radial repulsive field, for which the potential can be evaluated for a generic point in the space. The value of the potential of ς_j , as explained below, is taken into account by the cost function. \mathcal{J} is a summation of terms, one for every waypoint q_1, \dots, q_{N-1} , characterizing the path⁴:

$$\mathcal{J} = \sum_{k=1}^{N-1} J(q_k) \quad (8.31)$$

For the aim of computing proactive paths, $J(q_k)$ can be obtained as a summation of three terms:

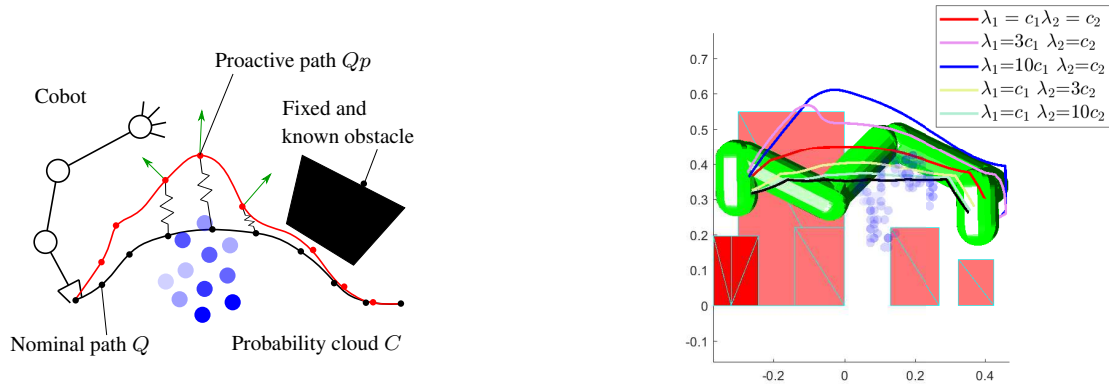
$$\begin{aligned} J(q_k) &= J_{Ob}(q_k) + \lambda_1 \cdot J_{Rep}(q_k) + \lambda_2 \cdot J_{Diff}(q_k) \\ J_{Rep}(q_k) &= \sum_{t \in T} \sum_{p_{Cj} \in \varsigma_j} \gamma_i \exp(-\alpha \|p_{Cj} - t\|_2) \\ J_{Diff}(q_k) &= \|q_{jk} - q_k\|_2 \end{aligned} \quad (8.32)$$

where:

- $J_{Ob}(q_k)$ is the collision cost. It is equal to the summation of the penetration depths of every robotic links in every fixed obstacles present. In case q_k is a collision free pose, $J_{Ob}(q_k)$ is equal to 0. For convex shapes, $J_{Ob}(q_k)$ can be computed by making use of the GJK algorithm in combination with EPA, refer to [126].
- J_{Diff} discourages big deformations of the path w.r.t. the nominal one, acting like an attractive field that counterbalances the repulsion induced by the probability cloud.
- J_{Rep} is a repulsive term. It is the value assumed by the potential of a field induced by ς_j , evaluated in a series of points $T = \{t_1, t_2, \dots\}$ along the kinematic chain of the manipulator (for instance the end effector and the elbow).
- $\lambda_{1,2}$ are tunable parameters balancing the importance of every term in J . Notice that both $\lambda_{1,2}$ must be sufficiently small, in order to avoid to compute poses that result in collision with the fixed obstacles. The exact values to assume are problem-specific (depending on the number of joints for the robot, the set of obstacles in the environment, etc...) and can be tuned through a set of few offline simulations.

Figure 8.13 summarizes the above considerations. The minimum for \mathcal{J} is found by applying the STOMP algorithm [57]. STOMP is essentially an iterative stochastic algorithm which progressively updates an initial path to improve its associated cost, deforming the position of the intermediate waypoints of the path. A certain number of samples (hypothesis of alternative poses) are generated at every iteration, locally exploring the configurational space. The poses for the path considered at the subsequent iteration, are obtained as weighted sums of the aforementioned samples. Weights

⁴ q_0 and q_N are not considered here because the starting and the ending poses are fixed.



(a) The initial path of the manipulator (black) and the corresponding proactive one (red). Blue points in the middle represent a probability cloud, which induces the repulsive field depicted with green arrows. The effect produced by the black springs in the figure is associated to the term J_{Diff} in the cost function. The black shape on the right upper corner is a fixed known obstacle.

(b) Different obtained results of STOMP, varying the proportion between the coefficients $\lambda_{1,2}$, on an example of point cloud taken from the real experiments.

Figure 8.13: Proactive planning.

are set proportionally to the value assumed by function J for the corresponding sample. This results in a kind of gradient descend applied to path planning. The main advantage with respect to similar strategies, is that STOMP does not require to know in a closed form the gradient $\partial_Q \mathcal{J}$, allowing large flexibility when designing the cost function. The right picture of Figure 8.13 reports different results obtained by STOMP, varying the parameters $\lambda_{1,2}$, when considering the nominal path assigned to YUMI for the experiments (see Section 8.2.4) and the probability cloud reported in the right pictures of Figure 8.20.

8.2.4 Experiments

The proactive approach has been tested adopting the experimental set up of Figure 8.14. The left arm of YUMI is equipped with a USB camera and is the only arm involved by the proposed experiments. Both the MICROSOFT KINECT and YUMI are connected to a CPU, which collects the information from the camera and computes the proactive paths for the robot. Proactive paths are then sent to YUMI, which executes them by making use of its native motion planning utilities.

The goal is to assemble a box containing the USB pen drive described in Section 3.1.3. The set of required operations is:

- action 1: take one newer box
- action 2: insert in the box one layer of foam and the USB pen drive
- action 3: bring the filled box to the first quality check station
- action 4: execute a first quality check
- action 5: take the filled box and put a cover

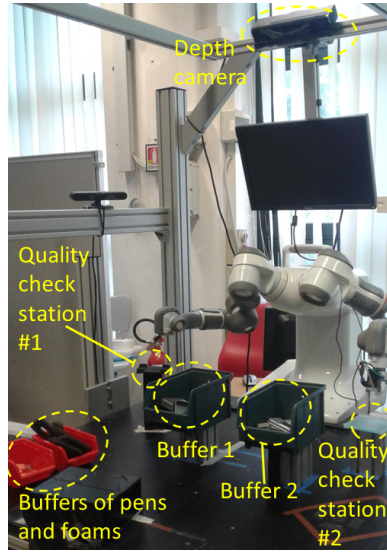


Figure 8.14: The experimental setup. The left arm of YUMI is involved in the collaborative assembly.

- action 6: put the assembled box on a second quality check station
- action 7: perform the final quality check

Actions 1,2,3,5 and 6 are executed by the human, while 4 and 7 are assigned to YUMI. Items are stored in buffers, which are located as indicated in Figure 8.14. Buffers 1 and 2 contain both a certain amount of boxes and covers. Therefore, it's up to the operator to choose buffer 1 or 2 when executing action 1 or 5. After the human completes action 3, YUMI receives the command to start action 4, after which it moves to the second quality check station waiting for the human to execute action 6. \mathcal{R} contains a single action r_1 , representing the motion of the robot between the two quality control stations (quality checks don't require the robot to move). After the second quality check, the robot goes to a position far from the shared space with the human, waiting for a new command. \mathcal{A}_S is composed of a_{S1} and a_{S2} . a_{S1} consists in the picking from buffer 1 (a box or a cover), while a_{S2} is a similar action for buffer 2. Two different databases store samples for a_{S1} and a_{S2} . Once a newer sample is available, the oldest one is deleted and the corresponding database is enriched with the newer one (FIFO logic). Every time a new proactive path is computed, the regressed trajectories Φ_1, Φ_2 are recomputed, considering the samples contained at that time in the buffers. The capacity of the databases was set equal to 3 samples for the proposed experiments. Notice that actions 2, 3, 6 are not contained in \mathcal{A}_S : action 2 because it doesn't require the human to cross the shared workspace with the robot, while actions 3 and 6 are executed while YUMI is in an idle state. The aim of the experiment is to compute proactive paths for r_1 . This choice was made to represent the human motion with a restricted number of recent samples, in order to acquire a certain level of adaptation.

14 participants were recruited for our experiments, which were divided into two groups of 7 each. The first group was asked to perform the collaborative assembly with the robot performing proactive paths, while the others performed the same assembly with the robot persistently executing its nominal path. For both the groups, the robot's motion was controlled with the strategy described in [134], modulating its speed along a

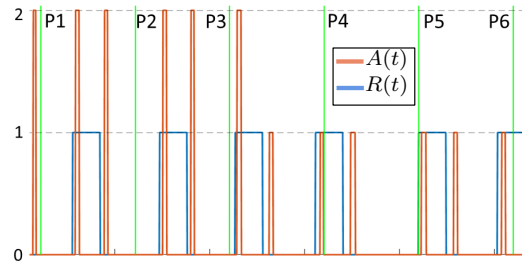


Figure 8.15: The activities executed during time by both the human and the robot. Green vertical line indicates instants at which a proactive path is recomputed. S_k indicates the k -th invocation of the proactive planning algorithm. When $A(t) = 1$ the human was performing a_{S1} , while $A(t) = 2$ refers to a_{S2} .

fixed path (respectively a proactive one and a path agnostic of the human presence).

When considering the experiments for the first group, at the beginning no prior knowledge about the operator’s movement is available, and the robot simply executes its assigned nominal path. After filling with samples the databases containing samples for a_{S1} and a_{S2} , the proactive computation loop starts, recomputing \mathcal{P}_{11} , \mathcal{P}_{12} and a newer proactive path every 20 s. This cycle time was selected in order to have, when updating the robotic path, approximately two new recorded samples for the human trajectories.

The evolution in time of $A(t)$, $R(t)$ for one of the experiments ⁵ of the first group is reported in Figure 8.15, while Figure 8.20 reports some significant proactive paths for the same experiment. Figure 8.20 reports some significant proactive paths computed for one of the experiments of the first group. In that experiment, the operator took all the parts from buffer 2 for the initial cycles, switching to buffer 1 for the final ones (this explains the values for the probabilities \mathcal{P}_{11} and \mathcal{P}_{12} reported in Figure 8.20). The total number of points considered for the binarization process (see Section 8.2.2) was 15625 (every dimension of the bounding boxes of Figure 8.12 was split into 25 uniform bins). As can be seen in Figure 8.20, when \mathcal{P}_{11} is greater than \mathcal{P}_{12} , the path is more deformed in its initial part (the one for which the robot passes close to the human when this latter is executing action a_{S1}), see the picture on the right of Figure 8.20. On the opposite, when \mathcal{P}_{11} is lower than \mathcal{P}_{12} , the path is more deformed toward the end, see the picture on the left of Figure 8.20.

Results

The performance obtained with the two groups of participants are now discussed, from both an objective and a subjective point of view. In all the experiments, a total amount of 4 boxes were assembled. The picture at the top of Figure 8.17 shows the distribution of the distance between the human and the robot during the experiments. There is a clear statistical evidence that it increases when applying proactive paths (single-tailed

⁵ A record of this experiment is also available at <https://www.youtube.com/watch?v=JExTJakGpZY>.

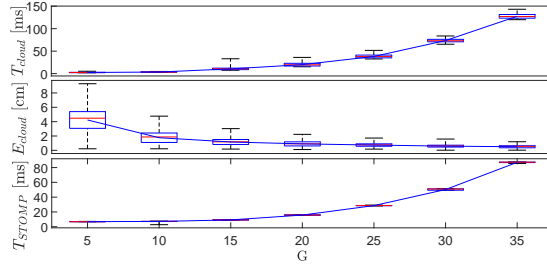


Figure 8.16: The pictures report a scalability analysis, varying the grid resolution G (see Section 8.2.2). The pictures on the top and on the bottom report the computational times for obtaining a single probability cloud and a single proactive path respectively. The picture in the middle reports the approximation error introduced when describing the human trajectories with different resolutions for the discrete grid. As can be seen, the computational times grows faster than the E_{cloud} decrease. Therefore, reducing G leads to minor computation times, without severely compromise the way the human motion is approximated.

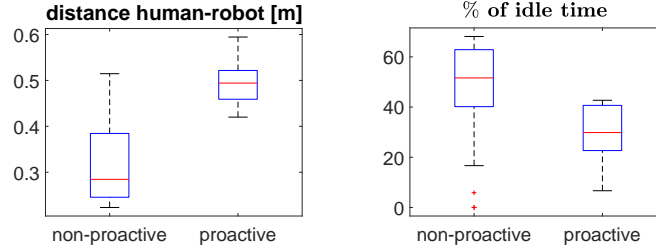


Figure 8.17: On the top the distribution of the distance between the human and the robot during the experiments; while on the bottom, the distribution of the percentage of idle time.

Wilcoxon rank sum test with confidence $\alpha = 0.05$ returns $r = 1.0 - 10^{-7}$. The bottom picture of Figure 8.17 reports the percentage of robot inactivity time (*i.e.* the amount of time for which the robot speed goes below the 1.0% of its nominal value). Also in this case a clear statistical evidence indicates that the latter quantity is reduced when considering pro active paths (single-tailed Wilcoxon rank sum test with confidence $\alpha = 0.05$ returns $r = 0.9984$). Apart from the quantitative metrics previously reported, the subjects were also asked to fill in a survey, whose results are reported in Figure 8.19. It is interesting to note that subjects that worked with the robot performing pro-active paths, seem to indicate that the perceived fluency and safety of the interaction have increased. The population of regressed human trajectories τ retrieved from the experiments was also exploited to conduct a scalability analysis, whose results are visible in Figure 8.16. For the general case of updating m robotic paths, taking into account n human trajectories, the total computation time T required for proactive planning would be equal to:

$$T = n \cdot T_{cloud}(G) + \sum_{j=1}^m \cdot N_j \cdot T_{STOMP}(G) \quad (8.33)$$

where in the above equation:

- T_{cloud} is the time required for computing a single probability cloud C , which is a function of G , *i.e.* the resolution adopted (see Section 8.2.2).

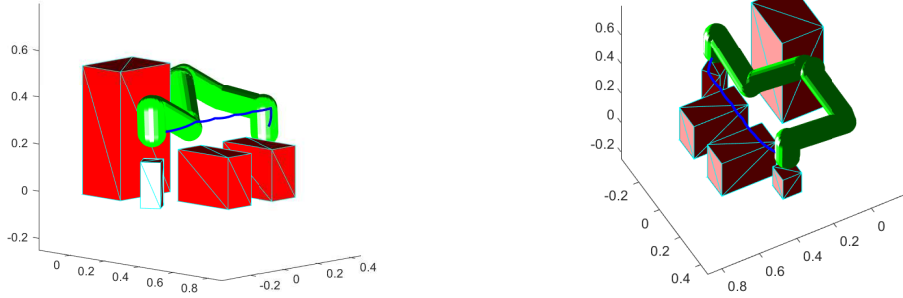


Figure 8.18: The nominal path realizing r_1 from two different views. The blue curve refers to the trajectory of the end effector. Red shapes represent the fixed obstacles considered by STOMP. Green capsules are adopted to depict the robot links for the initial and the final pose of the path.

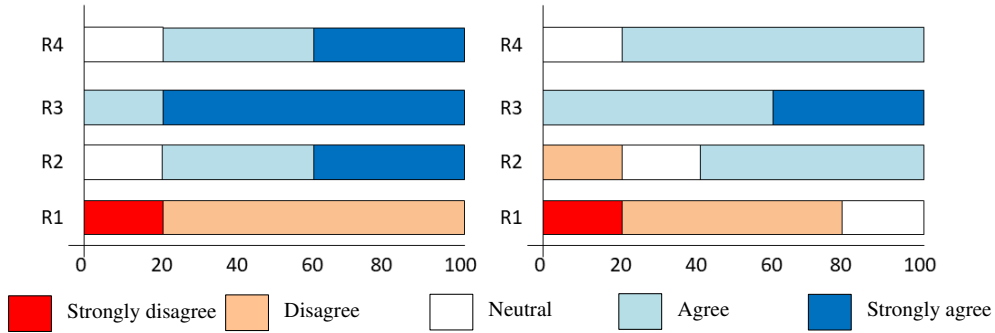


Figure 8.19: Subjects rate in percentage to the following quotes. R1: "The robot movements were unnatural and strange"; R2: "The interaction with the robot was fluent"; R3: "The interaction with the robot was safe"; R4: "The robot was able to efficiently forecast the human trajectories". Picture on the left refers to the group of participants for which pro active paths were executed. Picture on the right, to the group for which nominal paths were executed.

- T_{STOMP} is the time required to compute a single path, normalized w.r.t its number of waypoints.
- N_j is the number of waypoints of the j^{th} path.

The error E_{cloud} reported in Figure 8.16 refers, for a single frame Φ_k (in Figure 8.16 all errors for the frames from all the regressed trajectories computed are gathered), to the distance of a single skeletal point (the shoulder, the elbow or the wrist) to the nearest vertex in the grid depicted in Figure 8.12. The higher E_{cloud} is, the worse the resulting cloud describes the human volume occupation. The resolution G can be adapted in order to balance the precision of the clouds with the maximal allowed computation time. For the considered experiments, $m = 1$ $n = 2$ and G was set equal to 25. The path of the robot is made of 43 waypoints, leading to a mean computation time T equal to 1.23 s. Considering a more demanding scenario for which we suppose to have $m = 3$ $n = 5$, setting $G = 20$, the mean T would be about 1.9920 s.

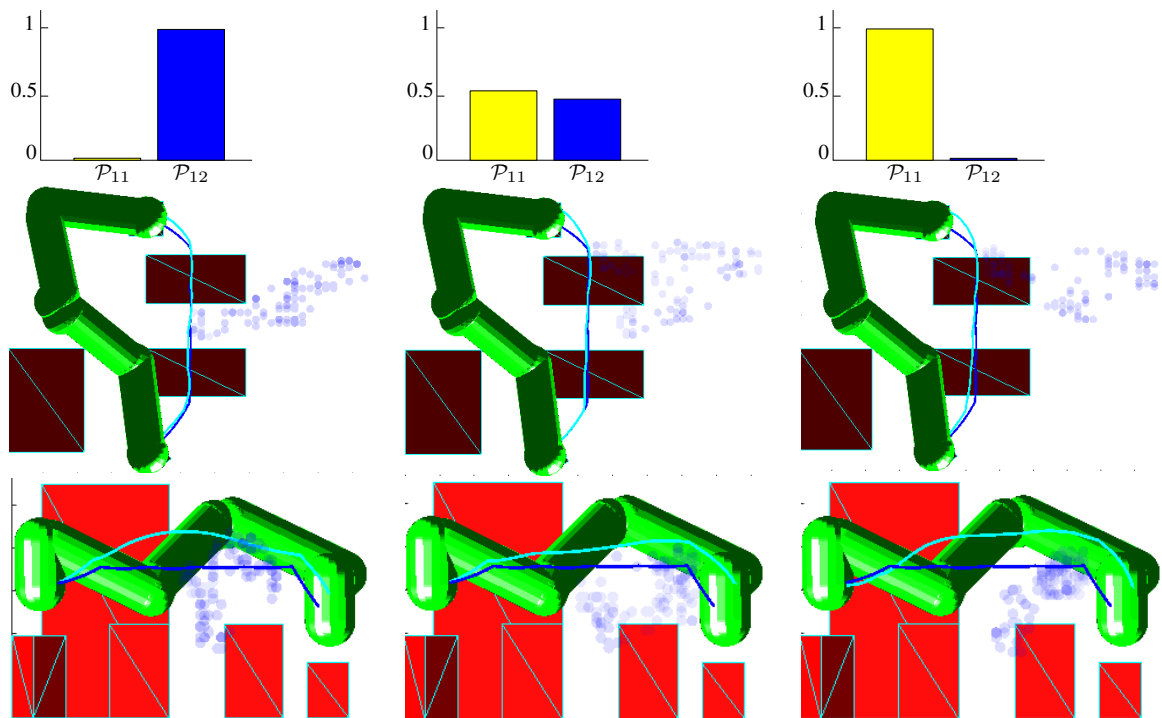


Figure 8.20: Examples of proactive paths obtained for some invocations of STOMP (Red shapes are the fixed obstacles populating the scene) during the experiments. For every column: on the top, the probabilities considered for the computation of the probabilistic cloud; in the middle and in the bottom, two distinct views of the proactive paths computed (cyan), compared with the initial nominal one (blue). The probability clouds considered for planning are depicted as a series of blue points whose intensity is proportional to the probabilities contained in γ .

CHAPTER 9

Occlusions handling

As clarified in the previous parts of this document, tracking the human motion is of paramount importance for understanding his or her behaviour. Indeed, the analysis of the operators motion is one of the few indirect available observations, exploited by the inferring algorithms described in Chapter 3. Also when assuming simple spherical areas for detecting the beginning or the ending of a human task, as done for instance in 7.5.2, at least the position during time of the operator's hands must be estimated. Moreover, when dealing with motion control of cobots, the perception of the human pose is crucial for imposing to the robots the proper trajectory to follow, as was extensively discussed in Chapter 8.

However despite many years of research, human pose estimation still remains a tough task due to: variability of human visual appearance in images, variability in lighting conditions, variability in human physique, partial occlusions due to self articulation and layering of objects in the scene, complexity of human skeletal structure, high dimensionality of the pose, and the loss of 3D information that results from observing the pose from 2D planar image projections [119]. Moreover, when considering cluttered cells the probability to incur in an occlusion of some anatomical parts is high. In such cases, the information about the position of some human anatomical parts is no longer available and it can result in unnecessary limitations on the generation of trajectories that the robot is allowed to follow, for safety reasons. Therefore, occlusions must be managed to gain both safety and productivity.

One possible approach could be to adopt multiple sensors, positioned so as to avoid as much as possible occlusions [138], [128]. In such context a strategy for sensors fusion must be developed. Other common approaches adopt non-parametric Bayesian filters such as the particle filter [43], which are characterized by representing the posterior distribution with samples of the state. [55] addresses the full-body articulated human

motion tracking from multi-view video sequences, making use of a particle swarm optimization (PSO). The joints in the kinematic tree are optimised in a sequence, starting with the torso and proceeding towards the arms, since motion of joints at higher levels of the kinematic tree constrains that of joints appearing at lower levels. A hierarchical approach to estimate the pose has been used also in [80], where the hand tracking problem is tackled using the Hierarchical Model Fusion framework (HMF), first proposed by Bray et al. [9]. It is a particle filter variant that decomposes the initial problem into smaller and simpler problems and efficiently addresses the implications of the high dimensionality.

In [38] the intuition that people tend to follow efficient trajectories rather than random paths is exploited. The proposed strategy learns common destinations within the environment by clustering training examples of trajectories, then a path planner procedure to predict future human motion helps the update stage of the underlying particle filter algorithm. Another machine learning method is proposed in [74], where Gaussian process models (see Appendix E) are used to compensate for self-occlusion. Their method relies on the evaluation of the similarity between the input posture and a large posture database, which clearly must be learnt.

Finally, [51] introduced the constrained motion proposal (COMP) algorithm, that uses multi-target proposal densities and motion models incorporating kinematic constraint information into a particle filter. Similar constraints will be exploited in this work too.

In [15], a novel approach to address the problem of human pose estimation in presence of occlusions was proposed and will be detailed in the following. When occlusions occur, the uncertainty relative to the pose is limited in areas of space consistent with the shape of the occlusions. The problem is cast in a constrained estimation of the state of a dynamical system. The developed method does not require any knowledge about the possible goals that a human operator intends to reach, or learning complex kinematic models from huge amount of training samples. Moreover it is not computationally heavy, allowing its implementation for on-line applications.

9.0.1 Representation of the human pose

By human pose, we refer to the configuration of the human body in space, which can be described in different ways, making use of several distinct kinematic models. The model introduced in [107] considers only the upper human body kinematic resulting in a 3-dofs base moving on the ground plane, one lumped 1-dof (flexion/extension) torso, a head (fixed) and two 4-dofs arms (shoulders are treated like spherical joints). The entire posture is described using a 12-components vector q , see also Figure 8.2.

An analogous representation can be given in terms of a set of 3D points composing a rough scheme of the human skeleton. The points of interest are: thorax (T), head (H), left shoulder (LS), right shoulder (RS), left elbow (LE), right elbow (RE), left wrist (LW), and right wrist (RW), see Figure 9.1. The position in the space of these points can be grouped in a single vector c . The skeletal representation is particularly useful because there are several algorithms in the literature (for instance [131]) that can be used to extract the skeletal points positions from a depth image. In order to move from the skeletal representation to the corresponding kinematic one, *i.e.* obtaining q from c , an inverse kinematics procedure can be implemented.

The trajectory followed by the human during time is a function $q(t)$, or equivalently

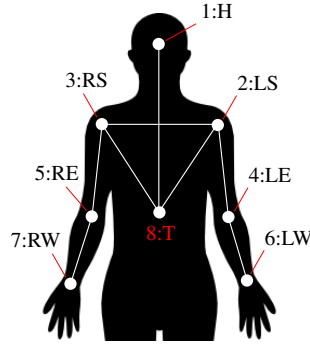


Figure 9.1: Skeletal approximation of the human silhouette.

a function $c(t)$. The pose estimation problem can then be cast in the state estimation of a dynamical system, having certain motion and measurement equations. Then, occlusions introduce additional constraints that must be taken into account, as will be discussed. Indeed, in case of occlusion, portions of the human silhouette cannot be seen and therefore the position of some skeletal points will not be available. In such cases, information retrieved from the depth map (which are always available) can be used to partially compensate.

9.1 Dealing with occlusions in the human silhouette

The problem of tracking the human posture in presence of occlusion was solved in [15] by making use of particle filtering (PF) [43]. The explanation of the tracking algorithm developed will be here reported, starting from a simpler example.

9.1.1 Single point tracking

Consider tackling the problem of tracking a simple point object in the 3D space. In this case the pose is completely defined by just knowing the values of the Cartesian coordinates (x, y, z) of the point, which can be grouped in a single vector $u = [x \ y \ z]^T$. The evolution in time of the pose can be described by the dynamics of three discrete integrators¹, leading to the following process equation:

$$U_{k+1} = AU_k + w_k \quad (9.1)$$

$$U_k = \begin{bmatrix} u_k \\ \dot{u}_k \\ \ddot{u}_k \\ \dddot{u}_k \end{bmatrix} \quad A = \begin{bmatrix} I & \Delta T I & \frac{\Delta T^2}{2} I & \frac{\Delta T^3}{6} I \\ 0 & I & \Delta T I & \frac{\Delta T^2}{2} I \\ 0 & 0 & I & \Delta T I \\ 0 & 0 & 0 & I \end{bmatrix} \quad (9.2)$$

w_k is the process noise, which is supposed to be Gaussian: $w_k \sim \mathcal{N}(0, W)$. W is a diagonal matrix obtained by considering for each block of the state vector the

¹It is easy to extend the state till higher order derivatives of u .

corresponding first truncated element of the Taylor expansion:

$$W = \begin{bmatrix} \sigma_u^2 \frac{\Delta T^4}{24} I & 0 & 0 & 0 \\ 0 & \sigma_{\dot{u}}^2 \frac{\Delta T^3}{6} I & 0 & 0 \\ 0 & 0 & \sigma_{\ddot{u}}^2 \frac{\Delta T^2}{2} I & 0 \\ 0 & 0 & 0 & \sigma_{\ddot{u}}^2 \Delta I \end{bmatrix} \quad (9.3)$$

Standard deviations $\sigma_u, \sigma_{\dot{u}}, \sigma_{\ddot{u}}, \sigma_{\ddot{u}}$ are tunable parameters. We can assume to have an automatic object detection procedure implemented in a depth camera monitoring the environment and providing an estimate for u_k in the camera frame (see Figure 9.2). The measurement equation became:

$$y_{k+1} = C s_{k+1} + e_k \quad (9.4)$$

$$C = [I_{3 \times 3} \quad O_{3 \times 9}] \quad (9.5)$$

$e_k \sim \mathcal{N}(0, E)$ represents the output noise, which is assumed to have a normal distribution too. On the system described by the equations (9.1) and (9.4) it is possible to apply the PF technique and retrieve the state estimate U_k . It is worth to point out that for the linearity characterizing the system, it holds that:

$$U_{k+1} \sim \mathcal{N}(AU_k, W) \quad y_{k+1} \sim \mathcal{N}(CU_{k+1}, E) \quad (9.6)$$

PF basically tries to approximate the posterior distribution of the estimated state directly by a set of finite samples called particles [43], which represent hypothesis about the real value of the system state U and are propagated during time. We denote the set of particles at time step k by $\mathcal{U} = \{U_k^{(1)}, \dots, U_k^{(N)}\}$ where N is the total number of particles adopted. At every step, new measurements y_{k+1} are available and the following stages are followed to update the particle set [43]:

1. *Motion update*: starting from the set \mathcal{U}_k a new set $\bar{\mathcal{U}}_{k+1}$ is obtained: each $\bar{U}_{k+1}^{(i)}$ is drawn from a proposal distribution $f_q(\bar{U}_{k+1}^{(i)} | U_{0, \dots, k}^{(i)}, y_{0, \dots, k})$. A common choice for f_q is to impose: $f_q(\bar{U}_{k+1}^{(i)} | U_{0, \dots, k}^{(i)}, y_{0, \dots, k}) = p(\bar{U}_{k+1}^{(i)} | U_k^{(i)})$. p takes into account only the process equation (equation 9.1), *i.e.* $p(\bar{U}_k^{(i)}) = f_{Gauss}(\bar{U}_k^{(i)} | W, AU_k^{(i)})$ (see Section 3.1.2 for the meaning of f_{Gauss}).
2. *Measurement update*: for each particle $U_{k+1}^{(i)}$ a weight $\omega_{k+1}^{(i)}$ must be computed. This quantity is proportional to the probability to retrieve y_{k+1} (which is the output measured by the sensor), when assuming the state at step $k+1$ equal to $\bar{U}_{k+1}^{(i)}$:

$$\omega_{k+1}^i \propto \omega_k^i h(y_{k+1} | \bar{U}_{k+1}^{(i)}) \frac{p(\bar{U}_{k+1}^{(i)} | U_k^{(i)})}{f_q(\bar{U}_{k+1}^{(i)} | U_{0, \dots, k}^{(i)}, y_{0, \dots, k})} \quad (9.7)$$

h is a likelihood function whose definition is made according to the measurement equation:

$$h(y_{k+1}) = \frac{\exp(-\frac{1}{2}(y_{k+1} - C\bar{U}_{k+1}^{(i)})^T R^{-1}(y_{k+1} - C\bar{U}_{k+1}^{(i)}))}{\sqrt{|2\pi R|}} \quad (9.8)$$

i.e. is the value assumed by a Gaussian distribution having as mean $C\bar{U}_{k+1}^{(i)}$ and covariance R . Since we have assumed f_q equal to p , we can rewrite equation (9.7) as follows:

$$\omega_{k+1}^i = \omega_k^i h(y_{k+1} | \bar{U}_{k+1}^{(i)}) \quad (9.9)$$

leading to a recursive update of weights.

3. *Resampling*: a new set of N particles \mathcal{U}_{k+1} is drawn from the posterior belief, which is described by considering values assumed by weights. Weights of particles in set \mathcal{U}_{k+1} (the one obtained after resample), will have all the same weights.

Concerning the resampling phase, this is particularly important in order to avoid degeneration problems [43]. In literature, different kinds of resampling techniques are available like the Multinomial Resampling (MR), the Stratified Resampling (SR), the Residual Resampling (RR), etc.. [102]. [15] made use of the Bootstrap Resampling (BR), which approximates the posterior belief as a discrete distribution with values consistent with weights.

Since the system expressed by equation (9.1) and (9.4) is linear, there would be no reason to adopt the PF formulation, since in this case the Kalman filter is proved to be the best filtering approach. However, the sample-based nature of the PF algorithm facilitates the process of including constraints in the state estimation problem, defining a second likelihood function [117]:

$$L_c(\bar{U}_{k+1}^{(i)}) = \begin{cases} 1, & \text{if } \bar{U}_{k+1}^{(i)} \in \mathcal{S}_{k+1} \\ 0, & \text{if } \bar{U}_{k+1}^{(i)} \notin \mathcal{S}_{k+1} \end{cases} \quad (9.10)$$

where \mathcal{S}_{k+1} represents the admissible region allowed by some constraints at time $k+1$. Then, the complete expression of function h can be assumed to be:

$$h = L_c(\bar{U}_{k+1}^{(i)}) h(y_{k+1} | \bar{U}_{k+1}^{(i)}) \quad (9.11)$$

This modification enables the algorithm to discard all particles violating constraints. Indeed, weights of these particles will be set to 0 and they will not survive to the subsequent resampling phase. Notice that nearly no extra computation is required to manage constraints. On the contrary, parametric filtering techniques like for instance the Unscented Kalman Filter, require the solution of some additional optimization problems to give a constrained estimation of the state.

When the object we want to keep track of goes under an occlusion, y_{k+1} is not available, but the information coming from the depth map is exploited translating it in a constraint to be enforced. Indeed, those particles for which the pose of the object would not result in occlusion (see Figure 9.2), have a $L_c(\bar{U}_{k+1}^{(i)}) = 0$, which implies that their weights will be set to 0 too. On the contrary, particles consistent with the depth map will have all the same weight, since no measurements are available. Similar considerations hold for the human silhouette and are the basis for the pose estimation algorithm as will be explained in the next Section.

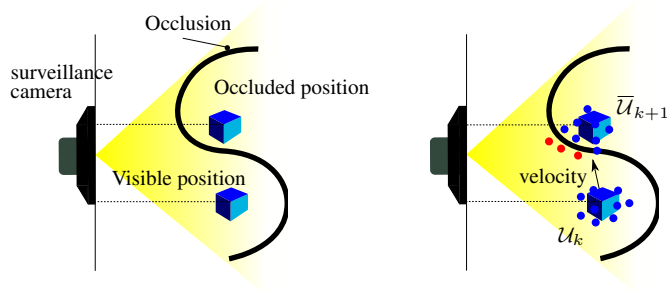


Figure 9.2: On the left, an example of occlusion, blue dotted lines represent information gained from the depth map. On the right, estimation of the pose using PF, blue dots are the positions of particles in set S , i.e. the one computed propagating set S of the previous step using the process equation. The object at step k is visible, then for the subsequent step goes under an occlusion. Red dots are used to indicate those particles not consistent with the depth map, which will not survive after the resampling step of PF.

9.1.2 Human pose tracking

In the previous Section an example of object position tracking, when moving in an environment for which occlusions occur, was shown. As discussed, additional constraints for the particles propagation have to be imposed. When considering tracking of the human silhouette, the mechanism is similar but additional constraints must be enforced. For each skeletal point (also indicated as joint) returned by a surveillance camera, a tracking state number is returned too, indicating whether the position of this point is certain or not (as in case of occlusion). Two distinct approaches can be used to describe the human posture: the one using a vector of joint values q and the one using the position in the Cartesian space of skeletal points c . When managing occlusions, these two approaches present pros and cons.

Representation of the pose

Consider the pose estimation using q . Since depth cameras provides the skeletal data in the camera space, to retrieve the measurements we need to perform a kinematic inversion, whose formulation has to be changed according to which skeletal point goes under occlusion. However, in the joint space it is very easy to impose constraints in order to satisfy physical limits on joint variables. Anyway, the description of the occlusion shape in the configurational space (C-space) is totally impractical for a space made of 12 dimensions. The alternative is to estimate the human pose through skeletal points, adopting as pose the vector c . In this way, no kinematic inversion is needed to run the PF because we can operate directly in the Cartesian space. On the other hand, imposing constraints on the feasible pose is very complex because constraints are naturally expressed as limits on the joint variables. Another disadvantage is related to the skeletal distances (which are constant during time) that have to be imposed as an additional constraint on the propagation of particles, as it will be detailed later. On the other hand, the constraints coming from the occlusions are very trivial to be handled because also the depth map is given in the Cartesian space. To ensure real-time computation, the approach which considers c to represent the human pose is suited: joint limitations are neglected for the pose estimation process.

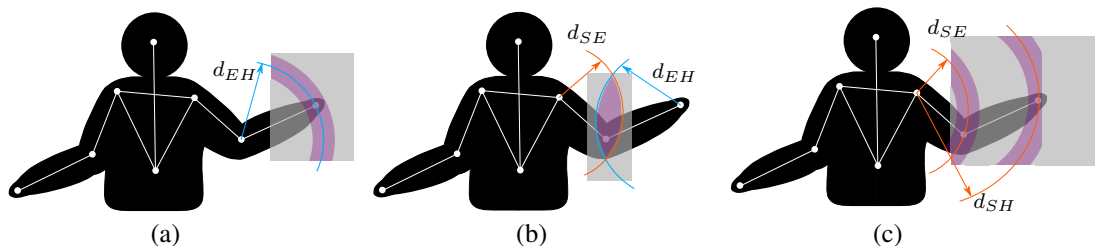


Figure 9.3: Possible occlusion cases considered for left arm. The shapes of the occlusions are depicted as a gray area, while the light purple area delimits the area admitted by the skeletal distances constraints.

Dealing with occlusions

Skeletal points are independently tracked. This strategy is adopted, instead of tracking at the same time all the 8 skeletal points, since PF suffers from dimensionality problem. This is a well-known issue in the literature ([43], Section I) that can be mitigated making use of hierarchical approaches [80]. Instead, [15] reduced the problem of pose estimation to a problem of tracking multiple point objects in the Cartesian space. Therefore, for every skeletal point, we adopt the process and measure equation expressed in equation (9.1) and (9.4) respectively.

\mathcal{U}_k^h refers to the set of particles related to the h^{th} skeletal point. The propagation step of PF, is always performed as described in Section 9.1.1, independently for every joint. Then in case the skeletal point considered is not in occlusion, the subsequent canonical steps of PF are performed. Instead, in case of occlusion the procedure is different and varies according to which points are under occlusion. In such cases, constraints coming from the knowledge of the depth map and the estimated values for the anatomical lengths must be taken into account. The distances between the skeletal points can be measured in an initialization phase, which starts when a new operator enters the scene. The management of occlusions relies on the definition of a hierarchical structure for the skeletal points. Here we limit the possible occlusions to those of the arms (nevertheless the analysis of all the 2^8 possible cases can be done). However, this simplifying hypothesis is reasonable if we consider typical human-robot collaboration contexts, like collaborative assemblies, for which the body parts having a high probability to be occluded are the arms.

Occlusion cases

Under the previous assumption, the possible situation of occlusions are the following ones:

- Case 1: occlusion of the hand
- Case 2: occlusion of the elbow
- Case 3: occlusion of both the hand and the elbow

Skeletal points are numbered as in Figure 9.1. The derivation of the hierarchical structure will be made according to this numbering. For the sake of simplicity the analysis of the possible occlusions will be performed only for the skeletal points associated to the left arm, then similar arguments can be used for the right one. Set \mathcal{U}^6 contains particles

$\{U_k^{6(0)}, \dots, U_k^{6(N)}\}$ describing possible states for the sixth skeletal point, which is the hand of the left arm. The first three values of vector $U_k^{6(i)}$ refer to the positional part of the state and can be denoted also as $H^{(i)}$. The estimate of the position of the left hand, can be computed as the mean of particles $H^{(i)}$, after resampling. Similar notations can be adopted for the other skeletal points.

Case 1: we can use the available information of the elbow to limit the possible position in space of the hand. In fact, the hand must be ideally at a distance d_{EH} , the Euclidean distance between the elbow and the hand computed during the initialization phase (refer to the left picture of Figure 9.3), from the estimated position of the elbow \bar{E} . Anyway this can result in a too restrictive constraint, that cannot be met by any particle $U_k^{6(i)}$. For this reason, the admissible region considered in this work will be a spherical crown with a mean radius equal to d_{EH} and a certain amplitude ϑ , rather than a simple spherical surface with radius equal to d_{EH} . To take into account both the constraint about the depth map and the skeletal distance, the Likelihood function will have the following expression:

$$L_c(\bar{U}_k^{6(i)}) = L_{c1}L_{c2} \quad (9.12)$$

where L_{c1} is an indicator function which is 1 if the particle is consistent with the depth map and 0 otherwise, while L_{c2} is defined in the following way ²:

$$L_{c2}(\bar{U}_k^{6(i)}) = \begin{cases} 1, & \text{if } -\vartheta \leq \|E - \bar{U}_k^{6(i)}\| - d_{EH} \leq \vartheta \\ 0, & \text{otherwise} \end{cases} \quad (9.13)$$

When considering Case 1, the skeletal point number 4 is defined as "father" of the skeletal point 6 and we denote this like $4 \succ 6$.

Case 2: here we can say that $2 \succ 4$ and also $6 \succ 4$. In such a way the particles associated to the elbow can propagate in the intersection of the two spherical crowns having a mean radius d_{SE} and d_{EH} , centered in the left shoulder and the left elbow respectively with the obvious meaning of notation (see the picture at the middle of Figure 9.3). In this case, function L_{c2} is computed as follows (function L_{c1} has the same definition):

$$L_{c2}(\bar{U}_k^{4(i)}) = \begin{cases} 1, & \text{if } -\vartheta \leq \|S - \bar{U}_k^{4(i)}\| - d_{SE} \leq \vartheta \wedge \\ & -\vartheta \leq \|H - \bar{U}_k^{4(i)}\| - d_{EH} \leq \vartheta \\ 0, & \text{otherwise} \end{cases} \quad (9.14)$$

Case 3: particles associated to the elbow can propagate in a spherical crown centered in the estimated position of the shoulder with mean radius d_{SE} , while the ones related to the hand can propagate inside the sphere of radius $d_{SH} = d_{SE} + d_{EH}$, once again centered in the shoulder. According to our formalism we have in this case $2 \succ 4$ and

²An alternative approach could be to consider a soft constraints formulation. The value assumed for function L_{c2} can be proportional to the degree of violation of constraints, *i.e.* distance of particles from a single spherical surface

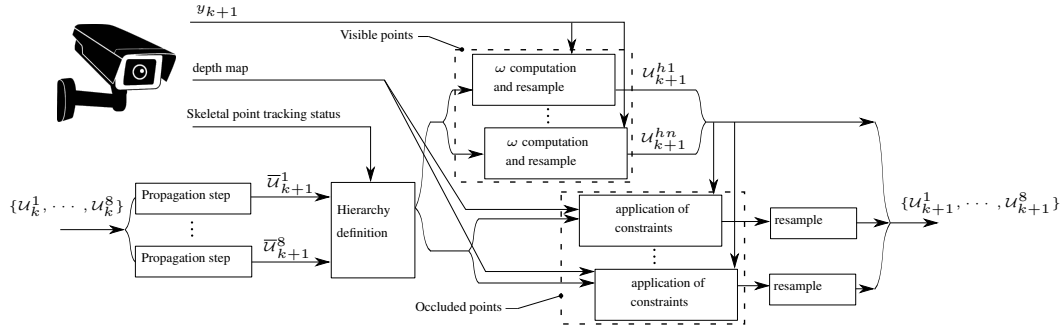


Figure 9.4: Pipeline of the proposed approach. y_{k+1} contains the estimated position of skeletal points, acquired from a depth camera sensor.

2 \succ 6 (refer to the right picture of Figure 9.3). The likelihood functions considered in this case for points 4 and 6 are analogous to those expressed by equations (9.13) and (9.14).

We are now in position to present the general structure of the tracking algorithm managing occlusions. Figure 9.4 reports a pipeline of the proposed strategy.

After the initialization phase, the system has to keep track of the pose of a human operator. At each iteration, the depth map and the position of the skeletal points (with the corresponding status, *i.e.* tracked or in occlusion) are acquired. Propagation of particles for every joint is performed and then according to which skeletal point is under occlusion, the aforementioned hierarchy is defined. Those sets of particles pertaining to points not under an occlusion, the canonical formulation of PF, equation (9.9), applies for the weight computation. On the opposite, those points under an occlusion, must consider the likelihood function in equation 9.11, for the computation of weights. Then, the resampling step is then performed for every set.

9.1.3 Experiments

The pose estimation algorithm described so far was applied in a realistic co-assembly of some electronic components over a printed circuit board. The robotic cell reproduced was populated with some obstacles, which force the human to go under an occlusion during the execution of the task. The layout of the cell is visible in Figure 9.5. A volume delimiting the robot workspace was defined: if a significant percentage (10 %) of the particles describing the position of a skeletal point is contained in this volume, the robot is forced to slow down on the assigned path to avoid a collision with the human.

The operations performed by the robot and the human are the following ones.

Robot operations (numbers refer to places indicated in Figure 9.5):

1. the left arm picks a board from the slider (1);
2. it carries the board in position (2);
3. when the left arm leaves the board the right arm goes in (2) and assembles a component on the board;
4. the left arm goes back in (2) and picks up the board;

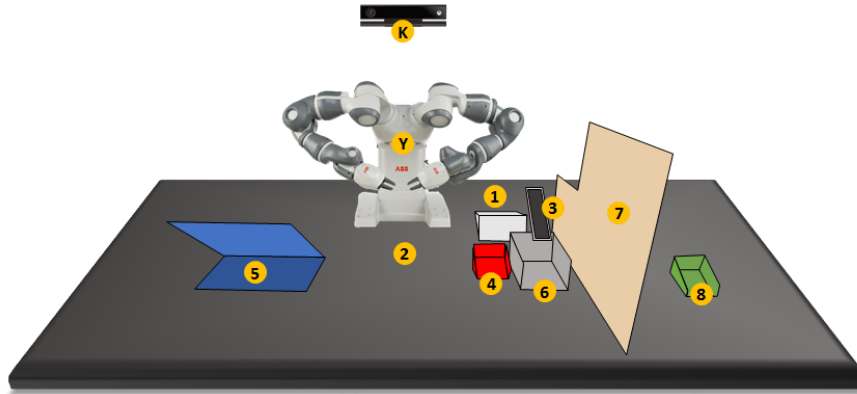


Figure 9.5: *The robotic cell considered for the experiment*

5. the board is brought to the slide (3);

Human operations (numbers refer to places indicated in Figure 9.5):

1. picks a board from the box (4);
2. takes a fuse from (5) and assembles it on the board;
3. places the board on the slide (1);
4. takes the worked board from the gray box (6) and leaves it in the green one (8).

The aim was to prove that managing occlusions can improve the productivity of the task, while at the same time preserving safety. To this purpose, the results obtained applying a Kalman filter formulation (essentially, the forward formulation detailed in Section 8.1.2) were compared to the PF one. In particular, Kalman filter was applied independently for every skeletal point (on the measure and process equation expressed by equation (9.4) and (9.1) respectively). When a point goes under an occlusion, the open loop Kalman filter formulation is applied for that point. In the latter case the covariance related to estimate starts growing (no measurements are available). If we consider a confidence of 90 % about the estimate given by Kalman filter, we get an ellipsoid that bounds the possible position of the skeletal point considered. When a significant portion of this ellipsoid intersects the volume defining the workspace of the robot, the robot is forced to slow down. Figure 9.6 compares the two approaches. For the experiment, 500 particles were assumed for every skeletal point when considering the PF formulation. The cycle times were logged during the experiments. One cycle is defined as the sequence of operations from a pickup board from slide (1) action to the next one. The results are reported in Figure 9.7. The reported statistics refer to 30 robot cycles per each technique considered³. By the analysis of Figure 9.7 we can state that managing occlusion is crucial for productivity. Indeed, cycle times are significantly reduced with the less conservative approach exploiting particle filtering.

³Records of the experiments are available at <https://www.youtube.com/watch?v=6ZFsinmKqjo>

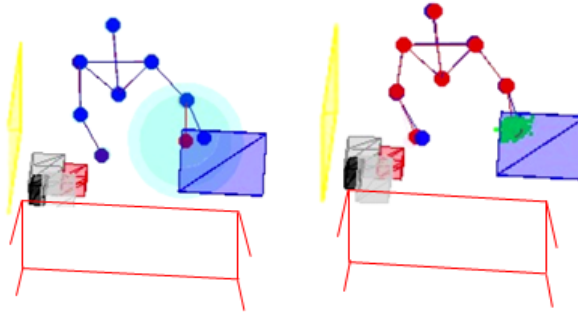


Figure 9.6: *The approaches considered to tackle occlusions. On the left, the Kalman filter formulation (KF), on the right the PF formulation. Red edges are those delimiting the workspace of the robot. In red are indicated the skeletal points estimated using KF or PF, while in blue are depicted the measurements retrieved from the MICROSOFT KINECT (values are always returned, even though their status indicate that is an occlusion present). KF goes in open loop when occlusion occur. This reflects in the growing of the uncertainty covariance ellipsoid, which can occupies regions inconsistent with the occlusion.*

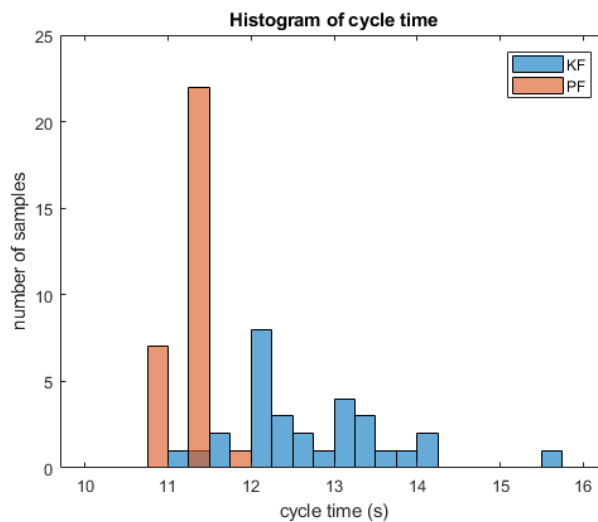


Figure 9.7: *Histograms relative to the robot cycle time for the approaches considered*

CHAPTER 10

Conclusions

The aim of this work was to study how to allow humans and robots to actively collaborate for performing a common goal. Particular attention was posed to industrial contexts, where typical applications are represented by collaborative assemblies. Indeed, there is no need to fully automatize the production processes: repetitive or dangerous actions are executed by the robots, while the ones requiring medium-high cognitive capabilities are performed by humans. For this reason, the problem becomes to let these two agents interact efficiently.

In particular, the humans were not conceived as additional controllable agents which are instructed to perform actions scheduled by a centralized planner. On the opposite, the decision-making capabilities of the operators are exploited, since they are allowed to drive the evolution of the plant. In this context, the robotic mates must be endowed with the cognitive capabilities required for both: understand and interpret the human behaviour (the current one as well as the past one) and predict the actions that a human will perform in the medium-term future. Both these tasks were tackled adopting a data-driven perspective: a machine learning model is typically off-line trained, representing an initial generic knowledge describing a generic human operator. Then, the different kind of data retrieved from the sensors connected to the robotic devices are exploited to adapt the model describing the human actions, obtaining something that is tailored for the specific operator that is interacting with the plant.

Predicting the human actions was proved to be crucial when scheduling the robotic activities. Indeed, when such predictions are properly exploited, the robots can synchronize with humans, performing in advance those operations that will enable the predicted human ones. Since humans have an intrinsically non-deterministic behaviour, a stochastic perspective was adopted: robust plans are computed for the plant to min-

imize the expected inactivity times of the humans and the robots. The developed assistive scheduling framework was proved to significantly improve the productivity of a plant, efficiently combining the flexibility of humans with the efficiency of robots. In particular, an approach that considers the human activity as uniformly distributed within two finite bounds, Section 7.2, can be efficiently adopted when the human is assigned simple and short actions, whose durations are low and exhibits a low (however non-negligible) variability. Instead, when more complex actions are done by humans, the Monte Carlo approach in Section 7.3 is more suited since it is able to handle a moderated stochasticity in the action durations. Finally, when dealing with workstations for which the robots and the humans actively share the space, not only the actions done by the human have a stochastic durations, but also the ones of the robots, since their motions must be controlled and regulated to avoid collisions. In such contexts, it is difficult to address the stochasticity by adopting a Monte Carlo approach, which would require too many samples to perform moderately well. In such circumstances, the fuzzy approach presented in Section 7.4 is efficiently able to produce robust plans for controlling the robot actions.

The assistive scheduling framework proposed is based on adapting the robots to the humans. A similar adaptation was also exploited for enhancing the motion control of cobots. Indeed, a machine learning perspective can be exploited for predicting the human motion. Then, the motion of robots can be optimized by considering such a prediction, as similarly done when scheduling robotic tasks according to the predicted future human actions. Both reactive and proactive strategies were proved to benefit from this approach.

The ideal cobots should immediately understand the human behaviour, aligning their behaviours very fast with the one of the human mate, even when tasks never seen before arise. The methodologies proposed in this work are an attempt to provide cobots such an ideal behaviour. However, many challenges remain.

For example, the adaptive learning approaches described in Sections 5.1, 5.2 and 8.2, requires at least one or two cycles, i.e. repetitions of the same assembly sequence, to start working properly. Therefore, to deal with much more agile and reconfigurable contexts, for which new tasks persistently arise, much efforts should be spent to reduce the learning time.

Moreover, in several parts of this work, it was assumed to deal with structured collaborations, for which the human actions are finite and known. This is reasonable in most of the actual industrial contexts. However, one key concept of the industry 4.0 framework is to design fast reconfigurable manufacturing plants. In such scenarios, the assumptions posed to the approach modelling the human behaviour can be difficult to be met.

For all these reasons, future works should address the adoption of frameworks performing well also when dealing with general evolving and non-structured layouts. Here, concepts and algorithms typical of the artificial intelligence domain could be exploited much more, providing methods for efficiently represent and interpret the workspace surrounding a cobot.

Finally, not only the robot adaptation is important, but also the human acceptance. Indeed, in all the developed strategies, the human indirectly influence the robot behaviour, which tries to adapt to the human needs. However, much more sophisticated strategies could be adopted to estimate or measure (by making use of additional sensors) the stress level of an operator. In this way, the goodness of a certain adaptation or another can be compared according to the measured effects produced on the human mate. This could be applied both at a task level (scheduling) as well as at a motion control level: do the human prefer to improve the productivity, by accepting more aggressive corrective maneuvers for the robot motion or does he/she prefer the safety, preferring to stay away from the cobot body? In this perspective, reinforcement learning approaches could be beneficial, allowing to obtain an optimal policy by considering as reward the stress perceived by humans.



Appendices

APPENDIX \mathcal{A}

Generalities about learning

Learning is usually done to determine the parameters $\theta = \{\theta_1, \theta_2, \dots\}$ of a probabilistic model describing the distribution of a variable X . This problem is addressed by considering a training set $\langle X^1, X^2, \dots \rangle$ made of independent samples of the distribution whose parameters have to be learnt. Function $L(\theta|X^{1,\dots})$, *i.e.* the likelihood of the parameters w.r.t. the training set, is the one to maximize for performing learning. $L(\theta|X^{1,\dots})$ is defined as follows:

$$L(\theta|X^{1,\dots}) \propto L(X^{1,\dots}|\theta) \cdot \mathbb{P}(\theta)_{prior} \quad (\text{A.1})$$

Priors for θ could be considered for taking into account an a priori knowledge about the real value of θ . Actually, the logarithmic likelihood $\log(L(\theta|X^{1,\dots}))$ is often considered and the elements in the training are assumed to be independent, leading to:

$$\log(L(\theta|X^{1,\dots})) = \sum_i \log(L(X^i|\theta)) + \log(\mathbb{P}(\theta)_{prior}) \quad (\text{A.2})$$

θ is computed as the value maximising $\log(L(\theta|X^{1,\dots}))$. In case a prior knowledge about the parameters is not available, $\mathbb{P}(\theta)_{prior} = 1$, leading to:

$$L(\theta|X^{1,\dots}) \propto L(X^{1,\dots}|\theta) \quad (\text{A.3})$$

Therefore, in such cases, maximising $L(\theta|X^{1,\dots})$ or $L(X^{1,\dots}|\theta)$ has the same effect.

Expectation Maximization

The Expectation-Maximization algorithm aims at learning the optimal parameter θ (see Appendix A) of a model having some observed variables X and also some latent ones Z . Z are variables whose values are hidden, but are linked in a probabilistic way to those observed, *i.e.* X . Learning is done according to a training set $\langle X^1, \dots, X^M \rangle$, made of realizations of X : the corresponding values for $Z^{1, \dots, M}$ are not known. EM is an iterative algorithm, which starts from an initial guess θ_0 and iteratively improves it. At every iteration, an Expectation and a Maximization are performed, explaining the name of the algorithm. The Expectation step is performed for taking into account the expectation of the likelihood w.r.t. Z , in order to maximise the likelihood of the training set, no matter the values for Z , which are, in a certain sense, eliminated.

As usually done (Appendix A), learning aims at maximizing a likelihood function involving the training set. In this case, we would like to find those θ maximising $L(X|\theta)$ ¹. $L(X|\theta)$ can be computed considering how the joint conditioned distribution of X, Z is factorizable:

$$\begin{aligned}\mathbb{P}(X, Z|\theta) &= \mathbb{P}(Z|X, \theta)\mathbb{P}(X|\theta) \\ \mathbb{P}(X|\theta) &= \frac{\mathbb{P}(X, Z|\theta)}{\mathbb{P}(Z|X, \theta)}\end{aligned}\tag{B.1}$$

Passing to the logarithms we obtain:

$$\log(\mathbb{P}(X|\theta)) = \log(\mathbb{P}(X, Z|\theta)) - \log(\mathbb{P}(Z|X, \theta))\tag{B.2}$$

We are now in position to describe the Expectation step of EM algorithm. Right hand side of equation (B.2) is a function of Z , which is unfortunately unknown. For

¹For the moment assume to have a single sample X in the training set

Appendix B. Expectation Maximization

this reason, we want to marginalize Z , by passing to the expectations w.r.t to density $\mathbb{P}(Z|X, \theta_k)$, where θ_k are the values of the parameter at step k :

$$\begin{aligned} \sum_Z \mathbb{P}(Z|X, \theta_k) \log(\mathbb{P}(X|\theta)) &= \sum_Z \mathbb{P}(Z|X, \theta_k) \log(\mathbb{P}(X, Z|\theta)) + \dots \\ &- \sum_Z \mathbb{P}(Z|X, \theta_k) \log(\mathbb{P}(Z|X, \theta)) \\ \log(\mathbb{P}(X|\theta)) \sum_Z \mathbb{P}(Z|X, \theta_k) &= \sum_Z \mathbb{P}(Z|X, \theta_k) \log(\mathbb{P}(X, Z|\theta)) + \dots \\ &- \sum_Z \mathbb{P}(Z|X, \theta_k) \log(\mathbb{P}(Z|X, \theta)) \end{aligned} \quad (\text{B.3})$$

Setting:

$$\begin{aligned} Q(\theta|\theta_k) &= \sum_Z \mathbb{P}(Z|X, \theta_k) \log(\mathbb{P}(X, Z|\theta)) \\ H(\theta_k|\theta_k) &= - \sum_Z \mathbb{P}(Z|X, \theta_k) \log(\mathbb{P}(Z|X, \theta)) \end{aligned} \quad (\text{B.4})$$

leads to ²

$$\log(\mathbb{P}(X|\theta)) = Q(\theta|\theta_k) + H(\theta|\theta_k) \quad (\text{B.5})$$

Considering the difference $\log(\mathbb{P}(X|\theta)) - \log(\mathbb{P}(X|\theta_k))$ and equation (B.5) leads to:

$$\log(\mathbb{P}(X|\theta)) - \log(\mathbb{P}(X|\theta_k)) = Q(\theta|\theta_k) - Q(\theta_k|\theta_k) + H(\theta|\theta_k) - H(\theta_k|\theta_k) \quad (\text{B.6})$$

At this point we can apply the Gibbs inequality, prescribing that in case of two distributions $f_{1,2}$ defined over the same domain applies what follows:

$$- \sum_x f_1(x) \log(f_1(x)) \leq - \sum_x f_1(x) \log(f_2(x)) \quad (\text{B.7})$$

Setting $f_1 = \mathbb{P}(Z|X, \theta_k)$ and $f_2 = \mathbb{P}(Z|X, \theta)$ the inequalities in equation (B.7) allows us to state that:

$$H(\theta|\theta_k) - H(\theta_k|\theta_k) \geq 0 \quad (\text{B.8})$$

and consequently that:

$$\log(\mathbb{P}(X|\theta)) - \log(\mathbb{P}(X|\theta_k)) \geq Q(\theta|\theta_k) - Q(\theta_k|\theta_k) \quad (\text{B.9})$$

For this reason, the Maximization step of the algorithm computes θ_{k+1} in order to increase Q , which leads indirectly (equation (B.9)) to an increase of the quantity of interest, *i.e.* $\log(\mathbb{P}(X|\theta))$. To be more precise, θ_{k+1} is computed as follows:

$$\theta_{k+1} = \operatorname{argmax}_\theta Q(\theta|\theta_k) \quad (\text{B.10})$$

It is not difficult to prove that function Q , when considering a training set made of a certain number of independent samples, is a summation of terms:

$$Q(\theta|\theta_k) = \sum_i Q_i(\theta|\theta_k) = \sum_i \sum_Z \mathbb{P}(Z|X^i, \theta_k) \log(\mathbb{P}(X^i, Z|\theta)) \quad (\text{B.11})$$

²Considering that $\sum_Z \mathbb{P}(Z|X, \theta_k) = 1$.

B.0.1 Learning of Gaussian Mixture Models

In case of GMM, the latent variables Z are the labels specifying which cluster produced every sample X^i . Let be $\gamma_j^i = \mathbb{P}(X^i \in Cluster_j)$ (see Section 3.1.2). γ_j^i is a function of the model parameters and therefore varies along the iterations of the EM algorithm, i.e. γ_{jk}^i . Let n_{jk} be the sum of γ over samples in the training set, i.e. $n_{jk} = \sum_i \gamma_{jk}^i$. When considering GMM, it is true what follows ³:

$$\mathbb{P}(Z = j|X^i, \theta_k) = \gamma_{jk}^i \quad (\text{B.12})$$

$$\begin{aligned} \mathbb{P}(X^i, Z = j|\theta) &= \mathbb{P}(X^i, |Z = j, \theta)\mathbb{P}(Z = j|\theta) \\ &= \lambda_j \left(\sqrt{2\pi |\Sigma_j|^n} \right)^{-1} \exp\left(-0.5(X^i - \mu_j)^T \Sigma_j^{-1} (X^i - \mu_j) \right) \end{aligned} \quad (\text{B.13})$$

In the second equation we exploited the fact that in a mixture model, weights λ expresses the a priori probability of a sample being generated by the corresponding cluster. Considering equations (B.12) (B.13), function Q in case of GMM is computable as follows:

$$\begin{aligned} Q(\theta|\theta_k) &= \sum_i \sum_j \gamma_{jk}^i \left(\log(\lambda_j) - 0.5 \log(|\Sigma_j|) - 0.5(X^i - \mu_j)^T \Sigma_j^{-1} (X^i - \mu_j) \right) \\ &= \sum_j n_{jk} \left(\log(\lambda_j) - 0.5 \log(|\Sigma_j|) \right) + \dots \\ &\dots + \sum_i \sum_j -0.5(X^i - \mu_j)^T \Sigma_j^{-1} (X^i - \mu_j) \end{aligned} \quad (\text{B.14})$$

The maximization step described before, has to solve a constrained maximization problem, considering Q as objective function and $\sum_j \lambda_j = 1$ as a constraint, since GMM are mixture models (equation (3.6)). Since we deal with an equality constraints, we consider the Lagrangian function Q' :

$$Q'(\theta) = Q(\theta|\theta_k) + \xi \left(\sum_j \lambda_j - 1 \right) \quad (\text{B.15})$$

where ξ is the lagrangian multiplier. The maximum of Q' is obtained by finding those combinations of values for which the gradient is null.

Imposing the gradient of Q' w.r.t. the generic weight λ_j equal to 0 leads to:

$$\begin{aligned} \frac{\partial}{\partial \lambda_j} &= \frac{n_{jk}}{\lambda_j} + \xi = 0 \\ \lambda_j &= -\frac{n_{jk}}{\xi} \end{aligned} \quad (\text{B.16})$$

³The same notation of Section 3.1.2 was assumed

Appendix B. Expectation Maximization

In order to let the constraint $\sum_j \lambda_j = 1$ be satisfied, we have to prescribe that:

$$\sum_j \lambda_j = \sum_j \frac{n_{jk}}{\xi} \Rightarrow \xi = - \sum_j n_{jk} \quad (\text{B.17})$$

Therefore, substituting into equation (B.16) leads to:

$$\lambda_{j \ k+1} = \frac{n_{jk}}{\sum_{j=1}^N n_{jk}} \quad (\text{B.18})$$

The gradient of Q' w.r.t. the generic weight μ_j is equal to:

$$\begin{aligned} \frac{\partial}{\partial \mu_j} &= \sum_i \gamma_{jk}^i \frac{\partial}{\partial \mu_j} \left(-0.5(X^i - \mu_j)^T \Sigma_j^{-1} (X^i - \mu_j) \right) \\ &= \sum_i \gamma_{jk}^i \Sigma_j^{-1} (\mu_j - X^i) \\ &= \Sigma_j^{-1} \left(n_{jk} \mu_j - \sum_i \gamma_{jk}^i X^i \right) \end{aligned} \quad (\text{B.19})$$

Imposing $n_{jk} \mu_j - \sum_i \gamma_{jk}^i X^i = 0$, ensures the entire gradient is null. Therefore, it applies what follows:

$$\mu_{j \ k+1} = \frac{\sum_i \gamma_{jk}^i X^i}{n_{jk}} \quad (\text{B.20})$$

Finally, the gradient w.r.t. to Σ_k is equal to (here the properties reported in [99] are exploited):

$$\begin{aligned} \frac{\partial}{\partial \Sigma_j} &= -0.5 n_{jk} \Sigma_j^{-1} + 0.5 \sum_i \gamma_{jk}^i \Sigma_j^{-1} (X^i - \mu_j)^T (X^i - \mu_j) \Sigma_j^{-1} \\ &= 0.5 \Sigma_j^{-1} \left(-n_{jk} I + \left(\sum_i \gamma_{jk}^i (X^i - \mu_j)^T (X^i - \mu_j) \right) \Sigma_j^{-1} \right) \end{aligned} \quad (\text{B.21})$$

Imposing $-n_{jk} I + \left(\sum_i \gamma_{jk}^i (X^i - \mu_j)^T (X^i - \mu_j) \right) \Sigma_j^{-1} = 0$ leads to:

$$\Sigma_{j \ k+1} = \frac{\sum_i \gamma_{jk}^i (X^i - \mu_j) (X^i - \mu_j)^T}{n_{jk}} \quad (\text{B.22})$$

At every step k of EM, every γ_{jk}^i is recomputed and equations (B.18), (B.20) and (B.22) are applied for updating the parameters of the mixture. After all the simplifications it turns out that the updating equations of EM, in case of training a GMM, have an heuristic interpretation. Indeed, equation (B.18), the new value of the weight of a cluster simply consider the importance if that cluster w.r.t. to all the others, *i.e.* the

summation of the probabilities that samples in the training set belongs to that cluster. The new means of the clusters are computed as a weighted mean, equation (B.20), which gives more importance to those samples having an high probability to belongs to the cluster for which the mean is re evaluated. A similar consideration holds for the covariance of clusters, equation (B.22).

APPENDIX C

Factor graphs

Factor graphs [39] are graphical models able to represent the probabilistic relationships existing among a network of stochastic variables. In such frameworks, the joint probability of the variables in the model is computed as a product of a certain number of factors. Each factor involves a sub portion of the entire population of variables. When dealing with directed graphical model, every factor must be a conditional probability distribution (these kind of models are also called Bayesian Networks), while in the case of undirected graphs, factors must simply be non negative functions describing the correlation existing among the variables involved. Such models are built essentially with the aim of propagating the belief across the network: when discovering the realization of certain variables in the net, the probabilistic relationships are taken into account for performing inference about all the other ones in the model. In this dissertation we will discuss only undirected graphical models, for which the variables involved are all categorical.

The generic categorical variable V involved in a factor graph has a discrete domain $Dom(V) = \{v_0, \dots, v_n\}$ containing all the possible realizations of V . The above notation will be adopted for the rest of this Section: capital letters will refer to variable names, while non capital refer to their possible realizations. Group of categorical variables can be in turn considered categorical variables, having a domain that is the Cartesian product of the domains of the variables constituting the group. Suppose variable $X = V_1 \cup V_2 \cup V_3 \cup V_4$, then:

$$Dom(X) = Dom(V_1) \times Dom(V_2) \times Dom(V_3) \times Dom(V_4) \quad (C.1)$$

The generic realization x of X is a set of realizations of the variables $V_{1,2,3,4}$, i.e. $x = \{v_1, v_2, v_3, v_4\}$.

The entire population of variables contained in a model is a set denoted as $\mathcal{V} = \{V_1, \dots, V_m\}$.

Appendix C. Factor graphs

When dealing with factor graphs, the probability of $\bigcup_{V_i \in \mathcal{V}} V_i$ ¹ is computed as the product of a certain number of components called factors.

Every single factor, sometimes also called a potential, is a positive real function describing the correlation existing among a subset of variables $\mathcal{D}^i \subset \mathcal{V}$. Suppose factor Φ_i involves $\{X, Y, Z\}$, i.e. $\mathcal{D}^i = \{X, Y, Z\}$. Then, $\Phi_i(X, Y, Z)$ is a non negative function defined over $D^i = X \cup Y \cup Z$. More formally:

$$\Phi_i(D^i) = \Phi_i(X, Y, Z) : Dom(X) \times Dom(Y) \times Dom(Z) \longrightarrow \mathbb{R}^+ \quad (\text{C.2})$$

The aim of Φ_i is to assume 'high' values for those combinations $d^i = \{x, y, z\}$ that are probable and low values (at least a null) for those being improbable. The entire population of factors $\{\Phi_1, \dots, \Phi_p\}$ is considered for computing the joint probability distribution of the variables in the model, i.e. $\mathbb{P}(V_{1, \dots, m})$. At this point the energy function E must be introduced:

$$E(V_{1, \dots, m}) = \Phi_1(D^1) \cdot \dots \cdot \Phi_p(D^p) = \prod_{i=1}^p \Phi_i(D^i) \quad (\text{C.3})$$

It is exploited for defining the joint probability distribution as follows:

$$\mathbb{P}(V_{1, \dots, m}) = \frac{E(V_{1, \dots, m})}{Z} \quad (\text{C.4})$$

Z is a normalization coefficient defined as follows:

$$Z = \sum_{\tilde{V}_{1, \dots, m} \in Dom(V_1 \cup \dots \cup V_m)} E(\tilde{V}_{1, \dots, m}) \quad (\text{C.5})$$

Knowing the joint probability of $V_{1, \dots, m}$, the probability distribution of a subset $\mathcal{S} \subset \{V_1, \dots, V_m\}$ can be in general (not only for graphical models) obtained through marginalization. Indeed, assuming \mathcal{C} as the complement of \mathcal{S} , then it follows that:

$$\mathbb{P}(S = s) = \sum_{\hat{c} \in Dom(C)} \mathbb{P}(S = s, C = \hat{c}) \quad (\text{C.6})$$

In the above equation, variables in \mathcal{C} were marginalized, i.e. they were in a certain sense eliminated since the probability of the sub set \mathcal{S} was of interest, regardless the possible realizations the variables in \mathcal{C} .

Although the general theory behind graphical models supports the existence of generic multivaried factors, here we will consider only pairwise (also called binary) or unary potentials. Both unary and binary potentials, can be of two possible classes:

- Simple: when the potential is simply described by a set of values characterizing the image of the factor.
- Exponential: they will be indicated with Ψ_i and their image set is defined as follows:

$$\Psi_i(X) = \exp(w \cdot \Phi_i(X)) \quad (\text{C.7})$$

where Φ_i is a simple factor. The weight w is tuned through learning (see Section C.0.2).

¹Which is the joint probability distribution of all the variables in a model

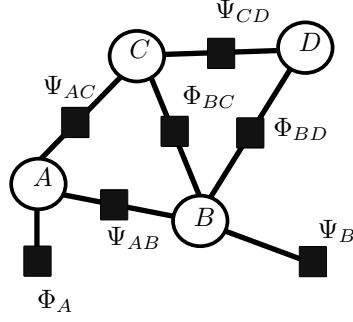


Figure C.1: Example of graph made of 4 variables: A, B, C and D . α, β, γ and δ are assumed as weights for the exponential potentials $\Psi_{AC}, \Psi_{AB}, \Psi_{CD}$ and Ψ_B respectively.

Figure C.1 reports an example of undirected graph: filled squares indicate the factors involved, while circles are used for describing the variables². For the reported graph \mathcal{V} is made of 4 variables: A, B, C, D and there are 5 binary potentials and 2 unary ones. Weights α, β, γ and δ are assumed for respectively $\Psi_{AC}, \Psi_{AB}, \Psi_{CD}, \Psi_B$. For the sake of clarity, the joint probability of the variables in Fig. C.1 is computable as follows:

$$\begin{aligned} \mathbb{P}(A, B, C, D) &= \frac{E(A, B, C, D)}{Z(\alpha, \beta, \gamma, \delta)} = \frac{E(A, B, C, D)}{\sum_{\tilde{A}, \tilde{B}, \tilde{C}, \tilde{D}} E(\tilde{A}, \tilde{B}, \tilde{C}, \tilde{D})} \\ E(A, B, C, D) &= \Phi_A(A) \cdot \exp(\alpha \Phi_{AC}(A, C)) \cdot \exp(\beta \Phi_{AB}(A, B)) \cdots \\ &\cdots \Phi_{BC}(B, C) \cdot \exp(\gamma \Phi_{CD}(C, D)) \cdot \Phi_{BD}(B, D) \cdot \exp(\delta \Phi_B(B)) \end{aligned} \quad (\text{C.8})$$

Graphical models are mainly used for performing belief propagation. To this aim, the subset $\mathcal{O} = \{O_1, \dots, O_f\} \subset \mathcal{V}$ must be defined for denoting the set of evidences: those variables whose value become known for some reason. Conversely, the hidden variables are contained in the complementary set $\mathcal{H} = \{H_1, \dots, H_t\}$. Clearly $\mathcal{O} \cup \mathcal{H} = \mathcal{V}$ and $\mathcal{O} \cap \mathcal{H} = \emptyset$. H will be used for referring to the union of all the variables in the hidden set:

$$H = \bigcup_{i=1}^t H_i \quad (\text{C.9})$$

while O is used for indicating the evidences:

$$O = \bigcup_{i=1}^f O_i \quad (\text{C.10})$$

The conditional distribution of H w.r.t. O can be determined in this way:

$$\begin{aligned} \mathbb{P}(H = h | O = o) &= \frac{\mathbb{P}(H = h, O = o)}{\sum_{\hat{h} \in \text{Dom}(H)} \mathbb{P}(H = \hat{h}, O = o)} \\ &= \frac{E(h, o)}{\sum_{\hat{h} \in \text{Dom}(H)} E(\hat{h}, o)} = \frac{E(h, o)}{Z(o)} \end{aligned} \quad (\text{C.11})$$

²The same graphical notation will be adopted also for the rest of this Chapter

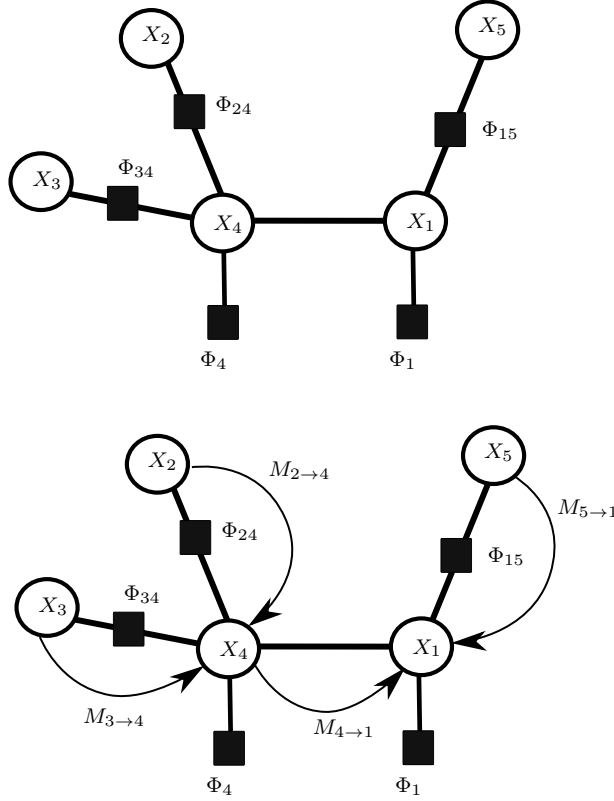


Figure C.2: Example of graph adopted for explaining the message passing algorithm. Below are reported the messages to compute for obtaining the marginal probability of variable x_1

The above computations are not actually done, since the number of combinations in the domain of H is huge also when considering a low-medium size graph. On the opposite, the marginal probability $\mathbb{P}(H_i = h_i | O = 0)$ of a single variable in $H_i \in \mathcal{H}$ is computationally tractable. Formally $\mathbb{P}(H_i = h_i | O = 0)$ is defined as follows:

$$\mathbb{P}(H_i = h_i | O = o) = \sum_{\tilde{h} \in \{\mathcal{H} \setminus H_i\}} \mathbb{P}(H_i = h_i, \tilde{h} | O = o) \quad (\text{C.12})$$

The above marginal distribution is essentially the conditional distribution of H_i w.r.t. O , regardless the values assumed by the other variables in \mathcal{H} .

C.0.1 The message passing algorithm

Message passing [132] is a powerful but conceptually simple algorithm adopted for propagating the belief across a net. Such a propagation is the starting point for performing many important operations, like computing the marginals of a variable or performing graph reductions.

An illustrative example

In order to discuss the message passing algorithm, consider the graph reported in Figure C.2, where, without loss of generality, all the factors were assume as simple potentials. Suppose also for the sake of simplicity that no evidences are available (i.e. $\mathcal{O} = \emptyset$).

We are interested in computing $\mathbb{P}(X_1)$, i.e. the marginal probability of X_1 . Recalling the previously introduced definition, the marginal probability can be obtained by the following computation:

$$\mathbb{P}(x_1) = \sum_{\tilde{x}_{2,3,4,5} \in \bigcup_{i=2}^5 X_i} \mathbb{P}(x_1, \tilde{x}_{2,3,4,5}) \quad (\text{C.13})$$

Simplifying the notation and getting rid of the normalization coefficient Z we can state the following:

$$\mathbb{P}(x_1) \propto \sum_{\tilde{x}_{2,3,4,5}} E(x_1, \tilde{x}_{2,3,4,5}) \quad (\text{C.14})$$

Adopting the algebraic properties of the sums-products we can distribute the computations as follows:

$$\mathbb{P}(x_1) \propto \Phi_1(x_1) \sum_{\tilde{x}_5} \Phi_{15}(x_1, \tilde{x}_5) \sum_{\tilde{x}_4} \Phi_{14}(x_1, \tilde{x}_4) \Phi_4(\tilde{x}_4) \sum_{\tilde{x}_2} \Phi_{24}(\tilde{x}_2, \tilde{x}_4) \sum_{\tilde{x}_3} \Phi_{34}(\tilde{x}_3, \tilde{x}_4) \quad (\text{C.15})$$

The first variable to marginalize can be \tilde{x}_2 or \tilde{x}_3 , since they are involved in the last terms of the sums products. The 'messages' $M_{2 \rightarrow 4}$, $M_{3 \rightarrow 4}$ are defined as follows:

$$\begin{aligned} M_{2 \rightarrow 4}(\tilde{x}_4) &= \sum_{\tilde{x}_2} \Phi_{24}(\tilde{x}_2, \tilde{x}_4) \\ M_{3 \rightarrow 4}(\tilde{x}_4) &= \sum_{\tilde{x}_3} \Phi_{34}(\tilde{x}_3, \tilde{x}_4) \end{aligned} \quad (\text{C.16})$$

Inserting $M_{2 \rightarrow 4}$ and $M_{3 \rightarrow 4}$ into equation (C.15) leads to:

$$\mathbb{P}(x_1) \propto \Phi_1(x_1) \sum_{\tilde{x}_5} \Phi_{15}(x_1, \tilde{x}_5) \sum_{\tilde{x}_4} \Phi_{14}(x_1, \tilde{x}_4) \Phi_4(\tilde{x}_4) M_{2 \rightarrow 4}(\tilde{x}_4) M_{3 \rightarrow 4}(\tilde{x}_4) \quad (\text{C.17})$$

At this point the messages $M_{4 \rightarrow 1}$ and $M_{5 \rightarrow 1}$ can be computed in the following way:

$$\begin{aligned} M_{4 \rightarrow 1}(x_1) &= \sum_{\tilde{x}_4} \Phi_{14}(x_1, \tilde{x}_4) \Phi_4(\tilde{x}_4) M_{2 \rightarrow 4}(\tilde{x}_4) M_{3 \rightarrow 4}(\tilde{x}_4) \\ M_{5 \rightarrow 1}(x_1) &= \sum_{\tilde{x}_5} \Phi_{15}(x_1, \tilde{x}_5) \end{aligned} \quad (\text{C.18})$$

After inserting $M_{4 \rightarrow 1}$ and $M_{5 \rightarrow 1}$ into equation (C.17) we obtain:

$$\begin{aligned} \mathbb{P}(x_1) &\propto \Phi_1(x_1) M_{4 \rightarrow 1}(x_1) M_{5 \rightarrow 1}(x_1) \\ \mathbb{P}(x_1) &= \frac{\Phi_1(x_1) M_{4 \rightarrow 1}(x_1) M_{5 \rightarrow 1}(x_1)}{\sum_{\tilde{x}_1} \Phi_1(\tilde{x}_1) M_{4 \rightarrow 1}(\tilde{x}_1) M_{5 \rightarrow 1}(\tilde{x}_1)} \end{aligned} \quad (\text{C.19})$$

which ends the computations. Messages are, in a certain sense, able to simplify the graph sending some information from an area of the graph to another one. Variables can be replaced by messages, that are treated like additional factors. Figure C.2 resumes the computations described. Notice that the computation of $M_{4 \rightarrow 1}$ must be done after computing the messages $M_{2 \rightarrow 4}$ and $M_{3 \rightarrow 4}$, while $M_{5 \rightarrow 1}$ can be computed independently from all the others.

Appendix C. Factor graphs

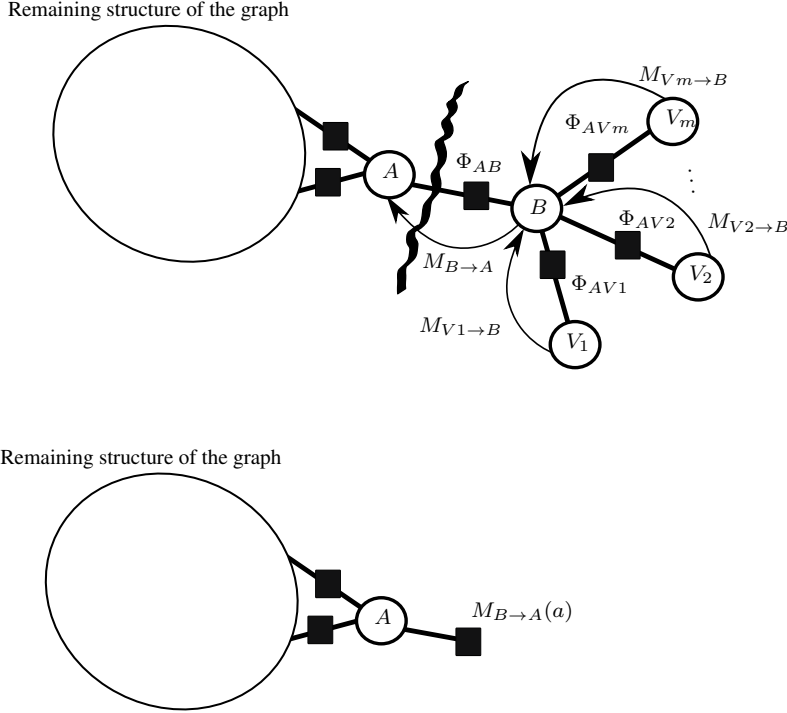


Figure C.3: On the top the general mechanism involved in the message computation; on the bottom the simplification of the graph considering the computed message.

The general case

The aforementioned considerations can be extended to a general structured graph. Look at Figure C.3: the computation of Message $M_{B \rightarrow A}$ can be performed only after having computed all the messages $M_{V_1, \dots, m \rightarrow B}$, i.e. the messages incoming from all the neighbours of B apart from A . Clearly $M_{B \rightarrow A}$ is computed as follows:

$$\begin{aligned}
 M_{B \rightarrow A}(a) &= \sum_{\tilde{b}} \Phi_{AB}(a, \tilde{b}) M_{V_1 \rightarrow B}(\tilde{b}) \cdots M_{V_m \rightarrow B}(\tilde{b}) \\
 &= \sum_{\tilde{b}} \Phi_{AB}(a, \tilde{b}) \prod_{i=1}^m M_{V_i \rightarrow B}(\tilde{b})
 \end{aligned} \tag{C.20}$$

Essentially, it's like having simplified the graph: we can append to A the message $M_{B \rightarrow A}(a)$ as it's a simple shape, deleting factor Ψ_{AB} and all the other portions of the graph, see Figure C.3. In turn, $M_{B \rightarrow A}(a)$ will be adopted for computing the message outgoing from A .

The above elimination is not actually done when performing belief propagation: all messages incoming to all nodes of the graph are computed and stored to be used for subsequent queries. This is partially not true when considering the evidences. Indeed, when the values of the evidences are retrieved, variables in \mathcal{O} can be deleted and replaced by messages, see Figure C.4. Suppose variable C is connected to a variable A through a binary potential $\Phi_{AC}(A, C)$ and to variable B through $\Phi_{B,C}$. Suppose also that variable C is an evidence assuming a value equal to \hat{c} , then the messages sent to A

Φ_{AB}	b_0	b_1	Φ_{XA}	x_0	x_1	Φ_{YB}	y_0	y_1
a_0	2	0	a_0	1	0.1	b_0	1	0.1
a_1	0	2	a_1	0.1	1	b_1	0.1	1

Table C.1: Value assumed by the energy function E , when having $X = 0$ and $Y = 1$ as evidences.

A	B	$E(A, B, X = 0, Y = 1)$
0	0	0.2
0	1	0
1	0	0
1	1	0.2

Table C.2: Factors involved in the graph of Figure C.7.

and B can be computed independently as follows:

$$\begin{aligned} M_{C \rightarrow A}(a) &= \Phi_{AC}(a, \hat{c}) \\ M_{C \rightarrow B}(b) &= \Phi_{BC}(b, \hat{c}) \end{aligned} \quad (\text{C.21})$$

The factors replacing the observed variables are considered for performing belief propagation on the hidden set, with the rules discussed so far.

The messages computation is possible as explained above only when considering politree, i.e. graph without loops. Indeed, for these kind of graphs the message passing algorithm is able in a finite number of iterations to compute all the messages, see Figure C.5. The same is not true when having loopy graphs (see Figure C.6), since deadlocking situations arise: no further messages can be computed since for every node some incoming ones are missing. In such cases a variant of the message passing called loopy belief propagation can be adopted. It initializes all the messages to basic shapes having the values of the image all equal to 1 and then recomputes all the messages many times till no changes are detected.

Maximum a posteriori estimation

Suppose we are not interested in determining the marginal probability of a specific variable, but rather we want the combination h_{MAP} for H that maximises the probability $\mathbb{P}(H = h_{MAP} | O)$. Clearly, we could try to compute the entire distribution $\mathbb{P}(H_{1, \dots, n} | O)$ and then take the value of H maximising that distribution. However, this is not computationally possible since even for low medium size graphs the size of $Dom(H)$ can be huge.

Maximum a posteriori estimations [37] (MAPs) solve this problem: the value maximising $\mathbb{P}(H_{1, \dots, n} | O)$ is computed, without explicitly building the entire distribution $\mathbb{P}(H_{1, \dots, n} | O)$. This is achieved by performing belief propagation with a slightly different version of the message passing algorithm presented before. Indeed, if we suppose to compute a MAP for the graph in C.3, the message to A would be computed as follows:

$$M_{B \rightarrow A}(a) = \max_{\tilde{b}} \left\{ \Phi_{AB}(a, \tilde{b}) \prod_{i=1}^m M_{V_i \rightarrow B}(\tilde{b}) \right\} \quad (\text{C.22})$$

Essentially, the summation in equation (C.20) is replaced with the max operator. After all messages are computed, the estimation $h_{MAP} = \{h_{1MAP}, h_{2MAP}, \dots\}$ is obtained

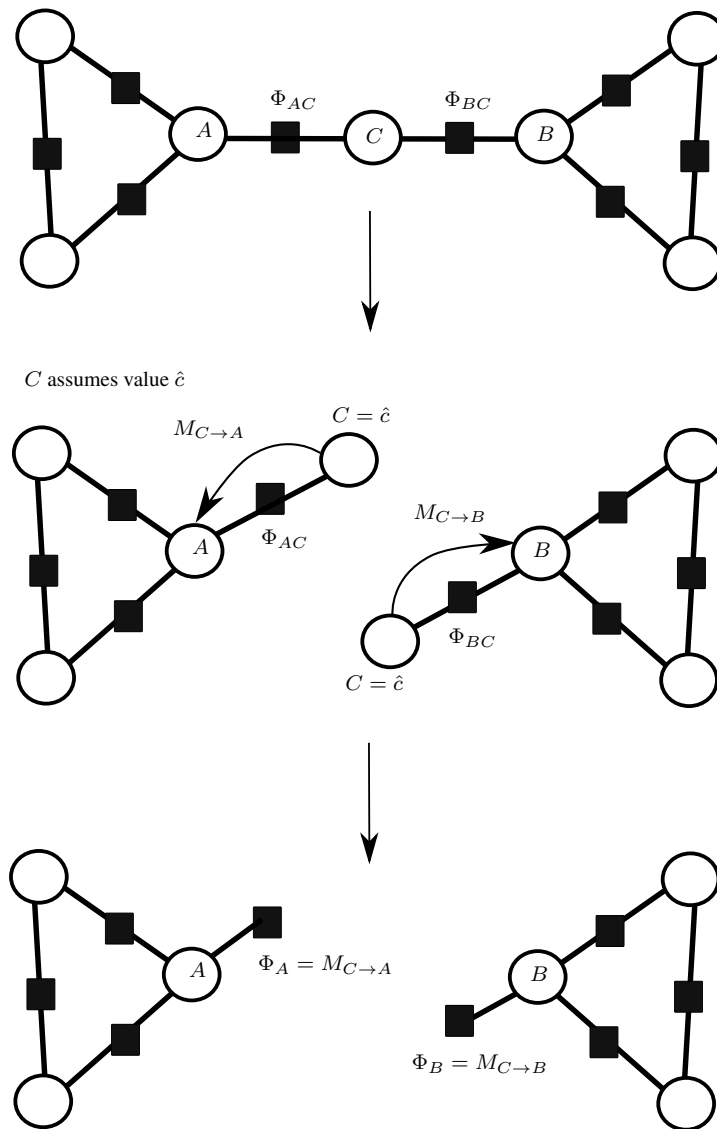


Figure C.4: When variable C become an evidence, is temporary deleted from the graph, replaced by messages.

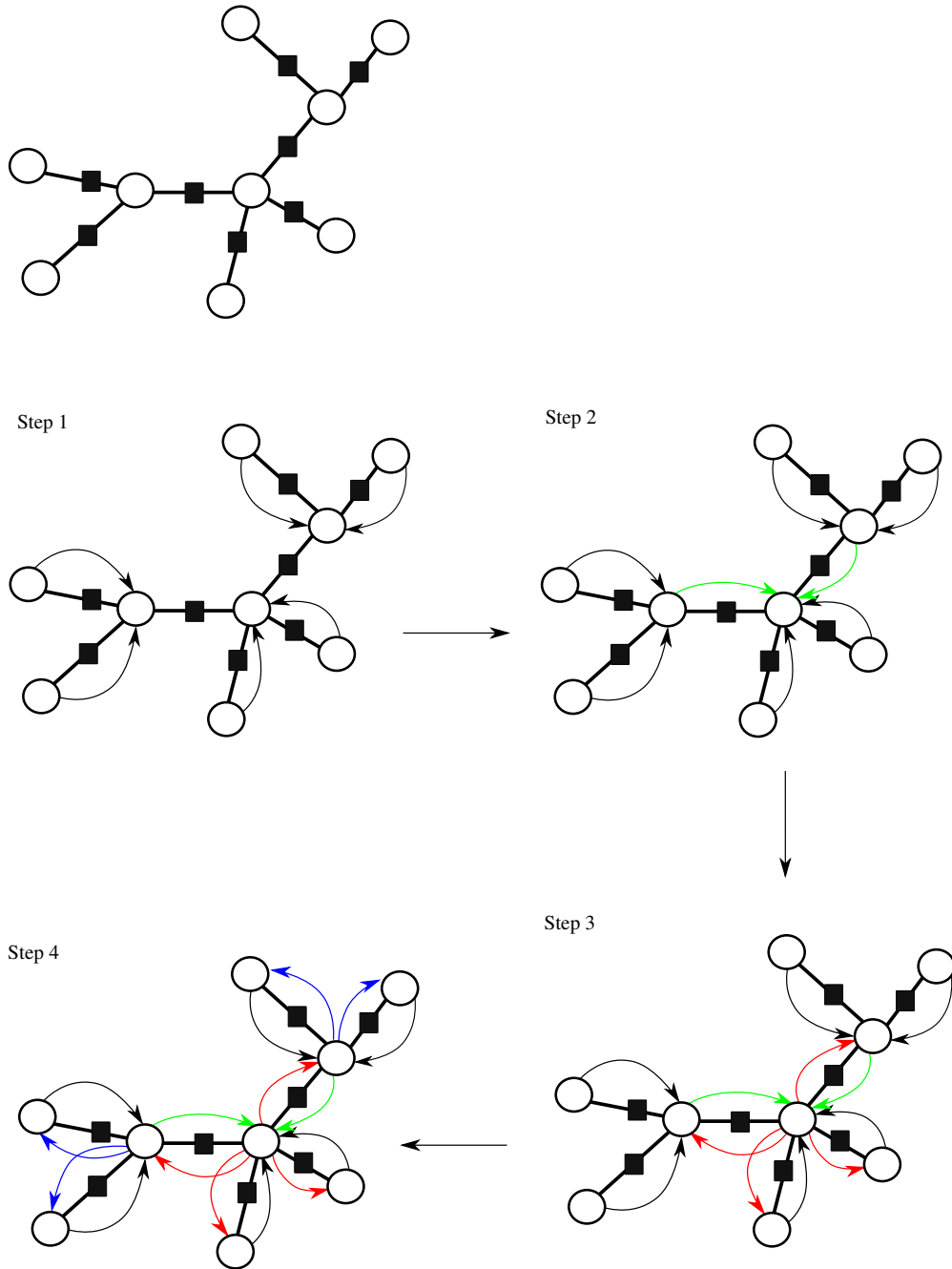


Figure C.5: Steps involved for computing the messages of the polintree represented at the top. The leaves are the first nodes for which the outgoing messages can be computed.

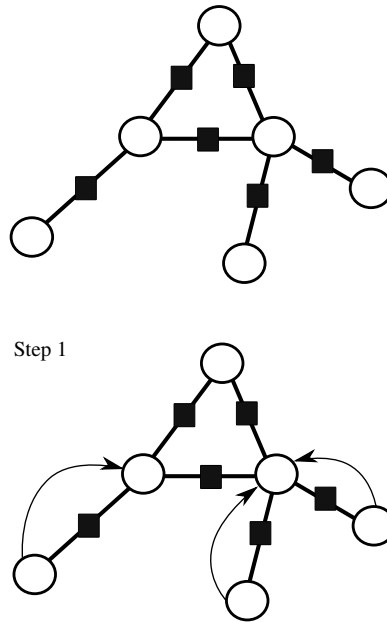


Figure C.6: Steps involved for computing the messages on a loopy graph: after computing the messages outgoing from the leaves, a deadlock is reached since no further messages are computable.

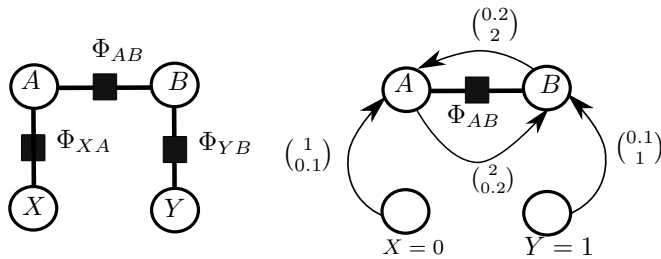


Figure C.7: Example of graph adopted. When the evidences are retrieved, the messages computed by making use of the message passing algorithm are reported below.

by considering for every variable in \mathcal{H} the value maximising:

$$h_{iMAP} = \operatorname{argmax} \left\{ \Phi_{H_i}(h_{iMAP}) \prod_{k=1}^L M_k(h_{iMAP}) \right\} \quad (\text{C.23})$$

where M_1, \dots, L refer to all the messages incoming to H_i . To be precise, this procedure is not guaranteed to return the value maximising $\mathbb{P}(H_1, \dots, H_n | O)$, but at least a strong local maximum was proved to be always obtained.

At this point it is worthy to clarify that the combination $h_{MAP} = \{h_{1MAP}, h_{2MAP}, \dots\}$ could not be obtained by simply assuming for every H_i the realization maximising the marginal distribution:

$$h_{MAP} \neq \{\operatorname{argmax}(\mathbb{P}(h_1)), \dots, \operatorname{argmax}(\mathbb{P}(h_n))\} \quad (\text{C.24})$$

This is due to the fact that $\mathbb{P}(H_1, \dots, H_n | O)$ is a joint probability distribution, while the marginals $\mathbb{P}(H_i)$ are not. For better understanding this aspect consider the graph reported in Figure C.7, with the potentials Φ_{XA} , Φ_{AB} and Φ_{YB} having the images defined in table C.1. Suppose discovering that $X = 0$ and $Y = 1$. Then, performing the standard message passing algorithm explained in the previous Section we obtain the messages reported in Figure C.7. Clearly individual marginals for A and B would be equal to:

$$\begin{aligned} \mathbb{P}(A) &= \begin{pmatrix} \mathbb{P}(A = 0) \\ \mathbb{P}(A = 1) \end{pmatrix} = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix} \\ \mathbb{P}(B) &= \begin{pmatrix} \mathbb{P}(B = 0) \\ \mathbb{P}(B = 1) \end{pmatrix} = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix} \end{aligned} \quad (\text{C.25})$$

Therefore, all the combinations $\{A = 0, B = 0\}$, $\{A = 0, B = 1\}$, $\{A = 1, B = 0\}$, $\{A = 1, B = 1\}$ maximise $\mathbb{P}(A, B | O)$. However, it is easy to prove that $E(A, B, X, Y)$ assumes the values reported in table C.2. Therefore, the combinations actually maximising the joint distribution $\mathbb{P}(A, B | O)$ are $\{A = 0, B = 0\}$ and $\{A = 1, B = 1\}$, leading to a different result.

C.0.2 Learning the weight parameters

In case of factor graphs, learning essentially consists in determining the optimal values for all the w (equation (C.7)), i.e. of all the weights of the exponential potentials involved in the graph. The set of tunable weights can be grouped into a vector $W = \{w_1, \dots, w_D\}$.

Learning can be done considering a training set $T = \{t_1, \dots, t_N\}$ made of realizations of the joint distribution correlating all variables in \mathcal{V} . As discussed at the beginning of this Appendix, when W is known, the probability of a combination t_j can be evaluated as follows:

$$\mathbb{P}(t_j) = \frac{E(t_j, W)}{Z(W)} \quad (\text{C.26})$$

At this point we can observe that the energy function is the product of two main factors: one depending from t_j and W and the other depending only upon t_j representing the

Appendix C. Factor graphs

contribution of all the non-tunable simple potentials:

$$\begin{aligned} E(t_j, W) &= \exp(w_1 \Phi_1(t_j)) \cdots \exp(w_D \Phi_D(t_j)) \cdot E_0(t_j) \\ &= \exp\left(\sum_{i=1}^D w_i \Phi_i(t_j)\right) \cdot E_0(t_j) \end{aligned} \quad (\text{C.27})$$

The likelihood function L can be defined as follows:

$$L = \prod_{t_j \in T} \mathbb{P}(t_j) \quad (\text{C.28})$$

passing to the logarithmic likelihood and dividing by the training set size N we obtain:

$$\begin{aligned} J = \frac{\log(L)}{N} &= \sum_{t_j \in T} \frac{\log(\mathbb{P}(t_j))}{N} \\ &= \sum_{t_j \in T} \frac{\log(E(t_j, W)) - \log(Z(W))}{N} \\ &= \frac{1}{N} \sum_{t_j \in T} \log(E(t_j, W)) - \log(Z(W)) \\ &= \frac{1}{N} \sum_{t_j \in T} \left(\sum_{i=1}^D w_i \Phi_i(t_j) \right) - \log(Z(W)) + \cdots \\ &+ \frac{1}{N} \sum_{t_j \in T} \log(E_0(t_j)) \end{aligned} \quad (\text{C.29})$$

The aim of learning is to find the value of W maximising J . This is done iteratively, exploiting a gradient ascent approach. The computations to perform for evaluating the gradient $\frac{\partial J}{\partial W}$ will be detailed in the following. Notice that in equation (C.29), term $\sum_{t_j \in T} \log(E_0(t_j))$ is constant and consequently will be not considered for computing the gradient of J . Equation (C.29) can be rewritten as follows:

$$J = \alpha(T, W) - \beta(W)$$

$$\alpha = \frac{1}{N} \sum_{t_j \in T} \left(\sum_{i=1}^D w_i \Phi_i(t_j) \right) \quad (\text{C.30})$$

$$\beta = \log(Z(W)) \quad (\text{C.31})$$

α is influenced by T , while the same is not valid for β .

Gradient of α

By the analysis of the equation (C.30) it is clear that:

$$\frac{\partial \alpha}{\partial w_i} = \frac{1}{N} \sum_{t_j \in T} w_i \Phi_i(t_j) \quad (\text{C.32})$$

Gradient of β

The computation of $\frac{\partial \beta}{\partial w_i}$ requires to manipulate a little bit more equation (C.31). Firstly the derivative of the logarithm must be computed:

$$\frac{\partial \beta}{\partial w_i} = \frac{1}{Z} \frac{\partial Z}{\partial w_i} \quad (\text{C.33})$$

The normalizing coefficient Z is made of the following terms (see also equation (C.4)):

$$Z(W) = \sum_{\tilde{V} \in \bigcup_{i=1}^p V_i} \left(\exp\left(\sum_{i=1}^D w_i \Phi_i(\tilde{V})\right) \cdot E_0(\tilde{V}) \right) \quad (\text{C.34})$$

Introducing equation (C.34) into (C.33) leads to:

$$\begin{aligned} \frac{\partial \beta}{\partial w_i} &= \frac{1}{Z} \frac{\partial}{\partial w_i} \left(\sum_{\tilde{V}} \exp\left(\sum_{i=1}^D w_i \Phi_i(\tilde{V})\right) E_0(\tilde{V}) \right) \\ &= \frac{1}{Z} \sum_{\tilde{V}} \frac{\partial}{\partial w_i} \left(\exp\left(\sum_{i=1}^D w_i \Phi_i(\tilde{V})\right) \right) E_0(\tilde{V}) \\ &= \frac{1}{Z} \sum_{\tilde{V}} \exp\left(\sum_{i=1}^D w_i \Phi_i(\tilde{V})\right) E_0(\tilde{V}) \Phi_i(\tilde{V}) \\ &= \sum_{\tilde{V}} \frac{\exp\left(\sum_{i=1}^D w_i \Phi_i(\tilde{V})\right) E_0(\tilde{V})}{Z} \Phi_i(\tilde{V}) \\ &= \sum_{\tilde{V}} \frac{E(\tilde{V})}{Z} \Phi_i(\tilde{V}) \\ &= \sum_{\tilde{V}} \mathbb{P}(\tilde{V}) \Phi_i(\tilde{V}) \end{aligned} \quad (\text{C.35})$$

Last term in the above equations can be further elaborated. Assume that the variables involved in potential Φ_j are $V_{1,2}$, then:

$$\begin{aligned} \frac{\partial \beta}{\partial w_i} &= \sum_{\tilde{V}} \mathbb{P}(\tilde{V}) \Phi_i(\tilde{V}) \\ &= \sum_{\tilde{V}_{1,2}} \Phi_i(\tilde{V}_{1,2}) \sum_{\tilde{V}_{3,4,\dots}} \mathbb{P}(\tilde{V}_{1,2,3,4,\dots}) \\ &= \sum_{\tilde{V}_{1,2}} \Phi_i(\tilde{V}_{1,2}) \mathbb{P}(\tilde{V}_{1,2}) \end{aligned} \quad (\text{C.36})$$

where $\mathbb{P}(\tilde{V}_{1,2})$ is the marginal probability of the variables involved in the potential Φ_i , which can be easily computable by considering the sub graph containing only V_1 and V_2 as variables. Notice that in case Φ_i is a unary potential the same holds, considering the marginal distribution of the single variable involved by Φ_i :

$$\frac{\partial \beta}{\partial w_i} = \sum_{\forall \tilde{V}_1} \Phi_i(\tilde{V}_1) \mathbb{P}(\tilde{V}_1) \quad (\text{C.37})$$

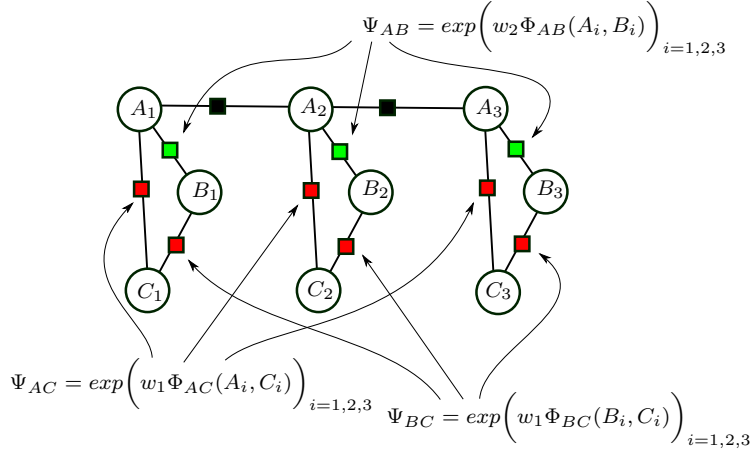


Figure C.8: Example of structure having many exponential potentials sharing the same modulating weight: w_1 is simultaneously involved into Ψ_{AC} and Ψ_{BC} , while w_2 is shared among the potentials connecting $A_{1,2,3}$ to $B_{1,2,3}$ respectively.

which can be easily obtained through the message passing algorithm (Section C.0.1).

After all the manipulations performed, the gradient $\frac{\partial J}{\partial w_i}$ has the following compact expression:

$$\frac{\partial J}{\partial w_i} = \frac{1}{N} \sum_{j=1}^N \Phi_i(D_j^i) - \sum_{\tilde{D}^i} \mathbb{P}(\tilde{D}^i) \Phi_i(\tilde{D}^i) \quad (\text{C.38})$$

C.0.3 Learning structures with shared weights

Suppose to have a structure for which many potentials share the same weight. In such cases, the learning must be done consistently with the aforementioned specification. Consider the example reported in Figure C.8. The computation of the gradients of the likelihood w.r.t. the weight involved in the graph will be detailed.

Gradient of α

Considering the model in Figure C.8, the α part of J (equation (C.30)) can be computed as follows:

$$\alpha = \frac{1}{N} \sum_{t_j} \left[w_1 \left(\sum_{k=1}^3 \Phi_{AC}(a_{kj}, c_{kj}) + \sum_{k=1}^3 \Phi_{BC}(b_{kj}, c_{kj}) \right) + w_2 \sum_{k=1}^3 \Phi_{AB}(a_{kj}, b_{kj}) \right] \quad (\text{C.39})$$

which leads to:

$$\begin{aligned} \frac{\partial \alpha}{\partial w_1} &= \frac{1}{N} \sum_{t_j} \left(\sum_{k=1}^3 \Phi_{AC}(a_{kj}, c_{kj}) + \sum_{k=1}^3 \Phi_{BC}(b_{kj}, c_{kj}) \right) \\ \frac{\partial \alpha}{\partial w_2} &= \frac{1}{N} \sum_{t_j} \left(\sum_{k=1}^3 \Phi_{AB}(a_{kj}, b_{kj}) \right) \end{aligned} \quad (\text{C.40})$$

Gradient of β

Regarding the β part of J we can write what follows:

$$\begin{aligned}
\frac{\partial \beta}{\partial w_1} &= \frac{1}{Z} \frac{\partial Z}{\partial w_1} \\
&= \frac{1}{Z} \frac{\partial}{\partial w_1} \left[\sum_{\tilde{V}} \exp \left(w_1 \left(\sum_{k=1}^3 \Phi_{AC}(\tilde{a}_k, \tilde{c}_k) + \Phi_{BC}(\tilde{b}_k, \tilde{c}_k) \right) + \dots \right) \right] \\
&= \sum_{\tilde{V}} \mathbb{P}(\tilde{V}) \left(\sum_{k=1}^3 \Phi_{AC}(\tilde{a}_k, \tilde{c}_k) + \Phi_{BC}(\tilde{b}_k, \tilde{c}_k) \right) \\
&= \sum_{k=1}^3 \left(\sum_{\tilde{V}} \mathbb{P}(\tilde{V}) \Phi_{AC}(\tilde{a}_k, \tilde{c}_k) + \mathbb{P}(\tilde{V}) \Phi_{BC}(\tilde{b}_k, \tilde{c}_k) \right) \\
&= \sum_{k=1}^3 \left(\sum_{\tilde{a}_k, \tilde{c}_k} \mathbb{P}(\tilde{A}_k, \tilde{C}_k) \Phi_{AC}(\tilde{a}_k, \tilde{c}_k) + \sum_{\tilde{b}_k, \tilde{c}_k} \mathbb{P}(\tilde{B}_k, \tilde{C}_k) \Phi_{BC}(\tilde{b}_k, \tilde{c}_k) \right) \tag{C.41}
\end{aligned}$$

and regarding the second weight

$$\begin{aligned}
\frac{\partial \beta}{\partial w_2} &= \frac{1}{Z} \frac{\partial Z}{\partial w_2} \\
&= \frac{1}{Z} \frac{\partial}{\partial w_2} \left[\sum_{\tilde{V}} \exp \left(w_2 \left(\sum_{k=1}^3 \Phi_{AB}(\tilde{a}_k, \tilde{b}_k) \right) + \dots \right) \right] \\
&= \sum_{\tilde{V}} \mathbb{P}(\tilde{V}) \left(\sum_{k=1}^3 \Phi_{AB}(\tilde{a}_k, \tilde{b}_k) \right) \\
&= \sum_{k=1}^3 \left(\sum_{\tilde{V}} \mathbb{P}(\tilde{V}) \Phi_{AB}(\tilde{a}_k, \tilde{b}_k) \right) \\
&= \sum_{k=1}^3 \left(\sum_{\tilde{a}_k, \tilde{b}_k} \mathbb{P}(\tilde{A}_k, \tilde{B}_k) \Phi_{AB}(\tilde{a}_k, \tilde{b}_k) \right) \tag{C.42}
\end{aligned}$$

Notice that the gradients $\frac{\partial J}{\partial w_1}$ and $\frac{\partial J}{\partial w_2}$ is the summation of many terms: the ones that would have been obtained considering separately the potentials in which w_1 or w_2 are involved (equation (C.38)). The above consideration has a general validity. Moreover, the same applies also when dealing with different structures, having some potentials that share some weights.

Fuzzy theory

Fuzzy numbers are particular kind of possibilistic distribution functions, adopted as an alternatively to probability density functions. Possibility theory was born as a need to represent vague predicates, enriching purely extensional concepts (for instance comparing two quantities $A > B$) with membership degrees, expressing the degree of truth of predicates.

The elements of a fuzzy set are characterized by a membership degree function μ , that assumes values between 0 and 1:

$$\mu : X \rightarrow [0, 1] \tag{D.1}$$

$\mu(x)$ represents the degree of truth about the fact that x is part of the set for which μ is the membership function. Equation (D.1) is similar to (3.3) as the intent of both probability and possibility theory is to model uncertain quantity. However, the main difference is that a membership degree function does not have to satisfy constraints similar to the one in equation (3.4). In a certain sense, possibility theory tries to extend results of the classic probability one, considering a greater class of distributions.

X in equation (D.1) is said to be the universe in which the fuzzy set is defined. The alpha-cut $\alpha_\varepsilon(\mu)$ groups all the values characterized by a membership degree greater or equal to a specific threshold. More formally:

$$\alpha_\varepsilon(\mu) = \{x \in X \text{ s.t. } \mu(x) \geq \varepsilon\} \tag{D.2}$$

A fuzzy set is said to be convex if all the possible alpha-cuts are compact sets.

The intersection of fuzzy sets μ_1, \dots, μ_n defined on the same universe X is computed as follows:

$$\mu = \mu_1 \cap \dots \cap \mu_n \Rightarrow \tag{D.3}$$

$$\mu(x) = \min(\mu_1(x), \dots, \mu_n(x)) \quad \forall x \in X \tag{D.4}$$

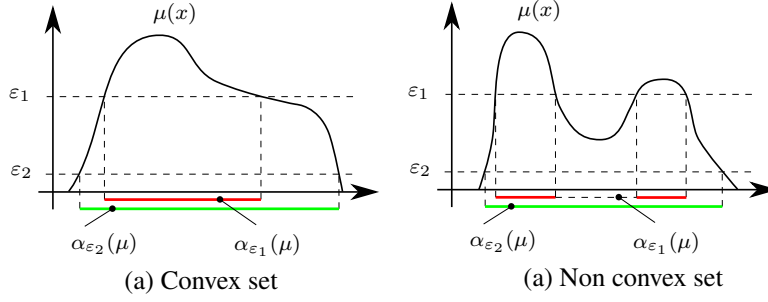


Figure D.1: Examples of a fuzzy convex set (on the left) and a non convex one (on the right). When considering the convex set, all the possible alpha-cuts are compact set, which is not verified for the set in the right.

Fuzzy numbers are particular fuzzy sets for which:

- convexity (as defined before) holds;
- $\sup(\mu) = 1$
- the membership function is piece-wise continuous

As a consequence, all fuzzy numbers membership functions are composed of a monotonically increasing part going from 0 up to possibility 1, and then a decreasing one ending in a value with a membership degree equal to 0. A conventional real number $r \in \mathbb{R}$, also called crisp number, can be seen as a particular fuzzy number having all possible alpha cuts that are singletons and a membership function which is a Dirac delta centred at r . Among all the possible classes of fuzzy numbers, the triangular one play a central role in fuzzy theory. Let be a a triangular number, then it holds that:

$$a(x) = \begin{cases} \frac{x-a_1}{a_M-a_1} & \text{if } x \in [a_1, a_M] \\ \frac{x-a_2}{a_M-a_2} & \text{if } x \in [a_M, a_2] \\ 0 & \text{otherwise} \end{cases} \quad (\text{D.5})$$

where a_1 and a_2 identify the boundaries of the support of A , while a_M is the peak, *i.e.* the value having the maximal possibility value. $\langle a_1, a_M, a_2 \rangle$ will be adopted for indicating in a compact way a .

All the main fuzzy algebraic operations are based upon the extension principle [30], that will be here briefly revised. Suppose to have a set of fuzzy numbers $\{\mu_1(x_1), \dots, \mu_n(x_n)\}$ and suppose we are interested in computing $y = g(x_1, \dots, x_n)$, where g is a non fuzzy standard function s.t. $g : \mathbb{R}^n \rightarrow \mathbb{R}$. The membership function describing y , $\mu_y(y)$, is defined as follows:

$$\begin{aligned} \mu_y & : Y \rightarrow [0, 1] \\ Y & = X_1 \times \dots \times X_n \\ \mu_y & = \sup \left\{ \min \left(\mu_{x_1}(x_1), \dots, \mu_{x_n}(x_n) \right) \text{ s.t. } g(x_1, \dots, x_n) = y \right\} \end{aligned} \quad (\text{D.6})$$

As a consequence of equation (D.6), the sum of two triangular numbers can be com-

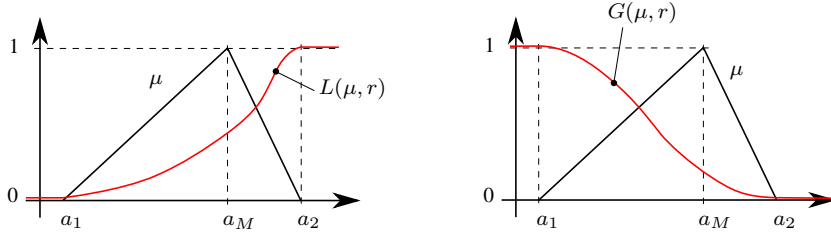


Figure D.2: Example of L (on the left) and G (on the right) of a triangular fuzzy number.

puted as follows:

$$\langle c_1, c_M, c_2 \rangle = \langle a_1, a_M, a_2 \rangle \oplus \langle b_1, b_M, b_2 \rangle \Rightarrow \quad (\text{D.7})$$

$$c_1 = a_1 + b_1$$

$$c_M = a_M + b_M$$

$$c_2 = a_2 + b_2 \quad (\text{D.8})$$

Another important operation is the comparison with a crisp number. Considering the fuzzy number $\mu(x)$ and a crisp number r , Firozja et al. [31] proposed the definition of a ranking function L extending the ordering relation $\leq r$, defined as follows:

$$L(\mu, r) : r \in \mathbb{R} \rightarrow [0, 1] \Rightarrow L(\mu, r) = \frac{\int_{-\infty}^r \mu(x) dx}{\int_{-\infty}^{\infty} \mu(x) dx} \quad (\text{D.9})$$

$L(\mu, r)$ expresses the degree of truth about the proposition $\mu \leq r$. Similarly, a dual ranking function $G(\mu, r)$ can be defined to address the proposition $\mu \geq r$:

$$G(\mu, r) : r \in \mathbb{R} \rightarrow [0, 1] \Rightarrow G(\mu, r) = \frac{\int_r^{\infty} \mu(x) dx}{\int_{-\infty}^{\infty} \mu(x) dx} \quad (\text{D.10})$$

Fig. D.2 reports an example of computation for G and L for a triangular number.

APPENDIX \mathcal{E}

Gaussian Processes

Gaussian Processes are a powerful tool for approximating unknown static mapping from an input space into an output one.

E.0.1 Scalar case

Suppose we need to approximate a function g defined as follows:

$$\begin{aligned} g &: \mathcal{X} \rightarrow \mathcal{Y} \\ \mathcal{X} &\subseteq \mathbb{R}^n \quad \mathcal{Y} \subseteq \mathbb{R} \end{aligned} \tag{E.1}$$

g is unknown and the only available information is represented by a training set S made of N samples $\left[\begin{array}{c} X^i \in \mathcal{X} \\ Y^i \in \mathcal{Y} \end{array} \right]$:

$$S = \left\langle \left[\begin{array}{c} X^1 \\ Y^1 \end{array} \right], \dots, \left[\begin{array}{c} X^N \\ Y^N \end{array} \right] \right\rangle \tag{E.2}$$

Since the values in S were generated by the same function g , they are in some way correlated. However, such a correlation is not known precisely. For this reason, Gaussian Processes approximate this correlation, assuming that all the values in S are jointly

Appendix E. Gaussian Processes

Gaussians, *i.e.*:

$$\begin{aligned}
 \begin{bmatrix} Y^1 \\ \vdots \\ Y^N \end{bmatrix} &\sim \mathcal{N}(0, K(X^1, \dots, X^N)) \\
 \mathbb{P}\left(\begin{bmatrix} Y^1 \\ \vdots \\ Y^N \end{bmatrix}\right) &= \frac{1}{\sqrt{(2\pi)^N |K|}} \exp\left(-\frac{1}{2} \begin{bmatrix} Y^1 & \dots & Y^N \end{bmatrix} K^{-1} \begin{bmatrix} Y^1 \\ \vdots \\ Y^N \end{bmatrix}\right) \\
 \mathbb{P}\left(\begin{bmatrix} Y^1 \\ \vdots \\ Y^N \end{bmatrix}\right) &= \frac{1}{\sqrt{(2\pi)^N |K|}} \exp\left(-\frac{1}{2} \text{Tr}\left(K^{-1} \begin{bmatrix} Y^1 \\ \vdots \\ Y^N \end{bmatrix} \begin{bmatrix} Y^1 & \dots & Y^N \end{bmatrix}\right)\right)
 \end{aligned} \tag{E.3}$$

The covariance matrix K , is a function of the inputs in the training set and it's defined as follows:

$$K = \begin{bmatrix} k(X^1, X^1) & \dots & k(X^1, X^N) \\ \vdots & \ddots & \vdots \\ k(X^N, X^1) & \dots & k(X^N, X^N) \end{bmatrix} \tag{E.4}$$

k is the kernel function and it's part of the model. As a general prescription, k must be defined in order to obtain a symmetric positive definite matrix K . For this reason, for any kind of kernel function it holds that $k(x, x') = k(x', x)$. k should be defined in order to assume low values for those entries that are strictly correlated. For example, when dealing with periodic function g , the kernel function k should be able to catch the periodicity, assuming a low value for a pair x, x' that is separated by approximately the value of the period. Common adopted functions are Radial Basis Function, Rational Quadratic kernel, Linear kernel, Periodic kernel, etc. [110].

A certain number of tunable parameters $\theta_{1,2,\dots}$, called hyperparameters, characterize the kernel function $k(\theta_{1,2,\dots})$. $\theta_{1,2,\dots}$ together with the training set S are actually what characterize a Gaussian Process model. The values of $\theta_{1,2,\dots}$ are determined after training, see Section E.0.1 and E.0.2. A Gaussian Process can be exploited for predicting the value assumed by $g(X)$ in a point X not present in S , see E.0.1. In other words, function g is approximated with a Gaussian Process $g_{GP}(X)$.

Prediction

Knowing S and $\theta_{1,2,\dots}$, a prediction $Y = g(X)$ for a generic entry X can be made. Indeed, $Y(X) = g(X)$ and the population of outputs present in S are assumed as

jointly Gaussian:

$$\begin{bmatrix} Y(X) \\ Y^1 \\ \vdots \\ Y^N \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} k(X, X) & k_X(X) \\ k_X(X)^T & K \end{bmatrix}, 0\right)$$

$$k_X(X) = [k(X, X^1) \quad \dots \quad k(X, X^N)] \quad (\text{E.5})$$

Therefore, since $Y^{1, \dots, N}$ are known, the conditional distribution is assumed as a prediction for Y^1 :

$$(Y|S) \sim \mathcal{N}\left(k_X(X)K^{-1} \begin{bmatrix} Y^1 \\ \vdots \\ Y^N \end{bmatrix}, k(X, X) - k_X(X)K^{-1}k_X(X)^T\right) \quad (\text{E.6})$$

As can be noticed, the prediction is not a value, but is a probability density function. Then, we can assume the mean of the above Gaussian (*i.e.* the value maximising the PDF) as a prediction, *i.e.*:

$$Y(X) \doteq g_{GP}(X) = k_X(X)K^{-1} \begin{bmatrix} Y^1 \\ \vdots \\ Y^N \end{bmatrix} \quad (\text{E.7})$$

Notice that to evaluate the expression in equation (E.6), the inverse of K is required. This is not computationally demanding, since after training K is a constant, meaning that the computation of K^{-1} can be done once for all.

Training

Training has the aim of tuning the hyperparameters $\theta_{1,2,\dots}$ characterizing the kernel function. The logarithmic likelihood of the model, see Appendix A appendix training, can be obtained considering the joint distribution of the samples in S , equation (E.3)²:

$$\begin{aligned} \mathcal{L} = & -\frac{N}{2} \log(|K(\theta)|) - \frac{1}{2} \text{Tr}\left(K(\theta)^{-1} \begin{bmatrix} Y^1 \\ \vdots \\ Y^N \end{bmatrix} [Y^1 \quad \dots \quad Y^N]\right) + \dots \\ & + \log\left(\mathbb{P}(\theta)_{\text{prior}}\right) \end{aligned} \quad (\text{E.8})$$

¹Here the expression of the conditional distribution of a multivariate Gaussian was exploited.

²Constant values are omitted

Appendix E. Gaussian Processes

The maximization of \mathcal{L} is typically done through gradient descend. Therefore, the gradient $\frac{\partial \mathcal{L}}{\partial \theta}$ must be evaluated ³

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \theta_i} &= -\frac{N}{2} \frac{\partial}{\partial \theta_i} \left(\log(|K(\theta)|) \right) - \frac{1}{2} \frac{\partial}{\partial \theta_i} \left(\text{Tr} \left(K(\theta)^{-1} \begin{bmatrix} Y^1 \\ \vdots \\ Y^N \end{bmatrix} [Y^1 \ \dots \ Y^N] \right) \right) + \dots \\
&+ \frac{\partial}{\partial \theta_i} \left(\mathbb{P}(\theta_i)_{prior} \right) \\
&= -\frac{N}{2} \frac{1}{|K(\theta)|} \frac{\partial}{\partial \theta_i} \left(|K(\theta)| \right) - \frac{1}{2} \left(\begin{bmatrix} Y^1 \\ \vdots \\ Y^N \end{bmatrix} [Y^1 \ \dots \ Y^N] \right)^T \frac{\partial}{\partial \theta_i} \left(K(\theta)^{-1} \right) + \dots \\
&+ \frac{\partial}{\partial \theta_i} \left(\mathbb{P}(\theta_i)_{prior} \right) \\
&= -\frac{N}{2} \frac{1}{|K(\theta)|} \text{Tr} \left(K(\theta)^{-1} \frac{\partial K(\theta)}{\partial \theta_i} \right) + \dots \\
&+ \frac{1}{2} \left(\begin{bmatrix} Y^1 \\ \vdots \\ Y^N \end{bmatrix} [Y^1 \ \dots \ Y^N] \right)^T K(\theta)^{-1} \frac{\partial K(\theta)}{\partial \theta_i} K(\theta)^{-1} + \frac{\partial}{\partial \theta_i} \left(\mathbb{P}(\theta_i)_{prior} \right) \quad (\text{E.9})
\end{aligned}$$

The expression of $\frac{\partial K(\theta)}{\partial \theta_i}$ depends on the kernel function adopted.

E.0.2 Vectorial case

Also vectorial functions can be approximated by Gaussian Processes. Suppose function g is defined as follows:

$$\begin{aligned}
g &: \mathcal{X} \rightarrow \mathcal{Y} \\
\mathcal{X} &\subseteq \mathbb{R}^n \quad \mathcal{Y} \subseteq \mathbb{R}^m \quad (\text{E.10})
\end{aligned}$$

The computations reported so far must be slightly modified for accounting the multidimensionality of \mathcal{Y} . Since g is vectorial, it's like having m distinct functions $g_{1,\dots,m}$:

$$g(X) = \begin{bmatrix} g_1(X) \\ \vdots \\ g_m(X) \end{bmatrix} \quad (\text{E.11})$$

Therefore, for approximating g , m distinct Gaussian Processes are required. The learning of such battery of Gaussian Processes, must be done considering a training set S , made of samples $Y^{1,2,\dots}$:

$$S = \left\langle \begin{bmatrix} X^1 \\ Y^1 = [Y_1^1 \ \dots \ Y_m^1] \end{bmatrix}, \dots, \begin{bmatrix} X^N \\ Y^N \end{bmatrix} \right\rangle \quad (\text{E.12})$$

³The derivatives were computed considering what reported in [99].

The single function g_i models the joint density of $\begin{bmatrix} Y_1^1 \\ \vdots \\ Y_1^N \end{bmatrix}$. Therefore, the joint density of $Y^{1,\dots,N}$ can be computed assuming m independent Gaussians:

$$\mathbb{P} \left(Y_1 = \begin{bmatrix} Y_1^1 \\ \vdots \\ Y_1^N \end{bmatrix} \right) \cdots \mathbb{P} \left(Y_m = \begin{bmatrix} Y_m^1 \\ \vdots \\ Y_m^N \end{bmatrix} \right) = \quad (\text{E.13})$$

$$= \left(\frac{1}{\sqrt{(2\pi)^N |K|}} \right)^m \prod_{i=1}^m \exp \left(-\frac{1}{2} \text{Tr} \left(K(\theta)^{-1} Y_i Y_i^T \right) \right) \quad (\text{E.14})$$

$$= \frac{1}{\sqrt{(2\pi)^{Nm} |K|^m}} \exp \left(-\frac{1}{2} \sum_{i=1}^m \text{Tr} \left(K(\theta)^{-1} Y_i Y_i^T \right) \right) \quad (\text{E.15})$$

$$= \frac{1}{\sqrt{(2\pi)^{Nm} |K|^m}} \exp \left(-\frac{1}{2} \text{Tr} \left(K(\theta)^{-1} [Y_1 | \cdots | Y_m] \begin{bmatrix} Y_1^T \\ \vdots \\ Y_m^T \end{bmatrix} \right) \right) \quad (\text{E.16})$$

Prediction

m distinct scalar predictions are made for predicting $g(X)$, leading to:

$$(Y|S) \sim \begin{bmatrix} \mathcal{N} \left(k_X(X) K^{-1} Y_1, k(X, X) - k_X(X) K^{-1} k_X(X)^T \right) \\ \vdots \\ \mathcal{N} \left(k_X(X) K^{-1} Y_m, k(X, X) - k_X(X) K^{-1} k_X(X)^T \right) \end{bmatrix}^T \quad (\text{E.17})$$

The value maximising the above conditioned probability is:

$$Y(X) \doteq g_{GP}(X) = k_X(X) K^{-1} [Y_1 | \cdots | Y_m] \quad (\text{E.18})$$

Training

Training is done analogously to the scalar case, considering a likelihood function that takes into account the joint distribution in equation (E.16):

$$\begin{aligned} \mathcal{L} &= -\frac{Nm}{2} \log(|K(\theta)|) - \frac{1}{2} \text{Tr} \left(K(\theta)^{-1} [Y_1 | \cdots | Y_m] \begin{bmatrix} Y_1^T \\ \vdots \\ Y_m^T \end{bmatrix} \right) + \cdots \\ &+ \log \left((\mathbb{P}(\theta))_{\text{prior}} \right) \end{aligned} \quad (\text{E.19})$$

List of Figures

2.1	Examples of goals related to actions in \mathcal{A}	9
3.1	Sets of evidences Φ considered by the approach in [135] and [17]. In case of [17], the estimation of z is made according to some detected facial points, which are depicted as blue points.	13
3.2	Pipelines of the inference approaches proposed in [135] and [17]. In [17], flag F is exploited to select the proper mixture model to use for computing the likelihood L	14
3.3	The uniform density between 0 and 1 is compared with an approximating mixture, varying the number of components.	17
3.4	Layout of the experimental robotic cell. Locations of possible human goals are indicated with white dotted circles.	19
3.5	Vibrotactile ring with its controller box. During the experiments the ring is worn on the operator's left hand and the box is attached to a Velcro bracelet worn on the forearm.	19
3.6	State machine adopted to send feedback to the operator. Note that vibration bursts are sent before the operator actually reaches the goals, according to the probabilities evolution.	21
3.7	Some trajectories taken from the experiments, going from goal 1 (blue) to goal 4 (red). Green markers are located at the points of the path for which the subject receives the haptic feedback.	21
3.8	The figure reports the likelihood of model GMM4d, varying the number of clusters considered when training the model. The selected number of clusters was 7.	22
3.9	Comparison of the three different inference methods regarding the goal recognition performance. Distance before recognition refers to the length of the path described by the wrist of the operator, from the instant at which the recognition of goal happens until the time at which the operator reaches goal 4.	23
3.10	24

List of Figures

3.11 Time elapsed from the instant when the tape is fixed until the one when the operator returns to goal 1. The overall population, composed of skilled and non-skilled subjects, was considered. 24

3.12 Subjects' answers in percentage to the question **Express how much you agree with the following statements concerning the vibrotactile ring.** 25

4.1 The approach followed to obtain the observations. The trajectory of the human is split into many windows of length equal to l_w . The mean and variance of some skeletal distance values are computed for obtaining the set of observations F_O^1, \dots, F_O^L 28

4.2 Example of graph construction. The observations are partitioned to actions, as described by the values contained in vector ρ . The observations $\hat{O}_{1, \dots, F}$ are computed as indicated in Section 4.1.1. 29

4.3 The intervals considered for computing \hat{O}_j from O_j , are obtained by considering equispaced portion of the image of the empirical cumulative distribution function describing the values that O_j can assume. 30

4.4 Examples of individual generation. At a random point, the solutions in the parents are broken and mixed to obtain the child. 32

4.5 Mechanism adopted for performing a sliding window segmentation. The first segmentation is computed as described in Section 4.1.3. Then, the surviving variables are designated (the gray area in the picture at the top) and the incoming messages are computed. The surviving graph is kept constant and taken into account for performing the segmentation on the second window of observations (picture in the middle). When the optimal segmentation for the second step is determined, the new surviving variables are identified and the procedure is iterated. 33

4.6 The procedure adopted for tuning the model parameters. The sequence in the training set (values for the hidden variables Y) are sampled from those consistent with a known set of precedence constraints. The values of the observed variables are sampled from the recorded examples of human motion (if a_i is sampled at a certain step as the performed human action, the value of the connected F_O is sampled from the ones associated with this particular action). 35

4.7 The experimental set-up taken by two distinctive perspectives. The operator's working area consists in 4 buffers storing parts and 3 assembly stations. Buffer A contains the first and the second kind of caps; B the springs and the lights; C the batteries and the batteries and the battery case while D the bodies of the torch. 36

4.8 Description of a_1, a_2 and a_4 . The locations of $M2$ and $M3$ is reported in Figure 4.7. 37

4.9 At the top, snapshots describing a_3 and a_5 , while at the bottom the precedence constraints existing among the actions. $M1$ is located as indicated in Figure 4.7. 38

4.10	Distributions of the segmentation error E_s on the data acquired by the experiments when considering: on the right an approach considering all the combinations of possible inter-skeletal distances and the distances of the wrists (right and left) w.r.t. the center of the buffers indicated in the left part of Figure 4.7; on the left a similar approach not taking into account the centres of the buffers storing raw materials (only the distance of the wrists w.r.t. the camera origin is considered).	38
4.11	An example of obtained segmentation. The black curve refers to value assumed in time by the time series $\hat{Y}_{1,2,\dots}, \hat{T}_{1,2,\dots}$, while $X_{1,2,\dots}, T_{1,2,\dots}$ is reported in blue. Since X_i is a marginal distribution of probability, a color code is exploited for representing it, refer to legend on the right.	39
5.1	Approach followed for predicting the waiting times.	42
5.2	Example of behaviour of the human: t_k represents the time instant corresponding to the activation of activity A_k , while \bar{t} represents the current time-stamp.	42
5.3	Example of transitions governing a Markov model. Transitions $a_1 \rightarrow a_1$, $a_1 \rightarrow a_2$, $a_1 \rightarrow a_3$ and $a_1 \rightarrow a_4$ are governed by the conditional probability distribution expressed by q^1 . q^1 has null values for rows 5 and 6, since the corresponding transitions are not allowed in this example.	43
5.4	Comparison of different methods in terms of prediction error: the presented algorithm ($n = 7$, blue), the VOMM method proposed in [114] (purple), a Markov Chain model ($n = 0$, red), a higher-order Markov Chain models trained with the algorithm proposed by Ching et al. in [26] ($n = 7$, yellow).	46
5.5	Examples of suffix tree updates. The structure of the tree after the update is reported for each example. The token sets Γ associated to the leaves are indicated in the lower part of the pictures containing the trees.	47
5.6	Example of domain ripartition. The left part of the Figure reports sets $\mathcal{D}_{1,2,3}$ when considering the suffix tree on the right.	50
5.7	Statistics of the prediction error obtained from the simulations. In all the figures, model order refer to the number of previous actions taken into account for computing the one-step probability prediction (in case of suffix trees is clearly σ) and the curve of the 50 th quartile is inserted into a shaded area delimited by the 80 th quartile and the 20 th one. The legend of reported in the right lower part applies. The pictures on the left part consider the complete assembly in Fig. 5.8, while the one the right takes into account the simplification reported in Fig. 5.9.	52
5.8	On the top left part the complete sequence of actions required for assembling an emergency button: the actions contained in a box can be done with no particular order, but before the actions contained in the boxes following in the sequence. A total number of 10 actions are needed to finalize the product. The top right part of the Figure reports the emergency button to assemble.	53
5.9	The Figure reports a simplification of the assembly reported in Figure 5.8, for which the size of set \mathcal{A} is equal to 5.	53

List of Figures

5.10	Example of prediction of human future activities. The transition probabilities associated to each arc are evaluated using a model for the activity sequence (see Section 5.1, 5.2). The lower bounds on the duration of each activities are used to prune branches of the tree that surely exceed the given prediction horizon ΔT . For all the remaining branches (three in the reported example), the corresponding distributions of waiting times τ_{branch} are computed and used within equation (5.37) to estimate the distribution of the waiting time needed for a certain activity to show up. In this example, the probability distribution of the waiting time of activity 4, <i>i.e.</i> τ^4 , is computed.	55
5.11	Example of sequence of activities (top) and corresponding typical behaviour of the estimate of the waiting time of activity 4, <i>i.e.</i> τ^4 , (bottom) evaluated and continuously updated during time.	56
5.12	Layout of the experimental setup: the human can access six stations, the central one being dedicated to the collaboration with the robot.	57
5.13	Different phases of the assembly procedure. IC insertion (top left): the human takes a PCB board from the red box on the left and an IC from the red rightmost box, inserts the IC in the pre-soldered socket, and finally fills the feeder. Quality check (top right): the robot takes a PCB from the feeder, accommodates it within a fixture, then it takes a picture of the PCB using the in-hand camera, and finally drops it on the feeder. Flat assembly and finalisation (bottom): the human takes a plastic enclosure from the left tray and places it in the fixture in front of the robot within the collaborative area, the robot picks a verified PCB and places it inside the enclosure, the human takes a flat cable from the right red box, meanwhile the robot takes the cap from a feeder and assists the human while fixing the cable on it, the robot accommodates the cap on the enclosure and finally stores the finished part.	57
5.14	Workflow of the robot program: based on <i>WaitingTime</i> , <i>i.e.</i> the p -percentile t_p returned by the algorithm, the first decision the robot takes is whether to wait for the human to initiate the collaborative operation (on the right, grey box) or to start the autonomous subtask (on the left, green box). The collaborative operation (in the middle, orange box) starts when initiated by the human.	58
5.15	Execution of the collaborative assembly experiment with the reactive (left) and the proactive (right) approach. The top Figures shows the sequence of activities of the human left hand and of the robot (blue and red represent autonomous activities, while the collaborative operation is marked in green). The bottom figures show the predicted time to collaboration (picking a box from the left tray, see Fig. 5.12) as compared to the ground truth (black).	59
5.16	Distribution of cycle times of the whole assembly sequence with the two approaches. The approach considering the predictive algorithm is responsible of a higher throughput as well as a reduced variability in cycle times.	60

5.17 Execution of the collaborative assembly when the human adopts a different pattern which consists in two consecutive IC insertions and two consecutive collaborative operations. The notation is identical to the one of Fig. 5.17, except from the blue curve which represents a purely data-drive approach.	60
6.1 Differences between a standard scheduling approach, on the top, and an assistive scheduling, on the bottom.	64
6.2 The assembly flow of a mix involving three different products. All actions with the same color have to be performed to produce a single finite product.	66
6.3 Example of task allocation. Actions refer to the assembly flow of Figure 6.2	66
6.4 Portion of a reachability tree. Nodes $S_{6,7,8}$ are reached by firing controllable transitions, <i>i.e.</i> $\alpha_4 = \alpha_{6,7,8}$. $m_9 = m_5$ since the same kind of transitions lead to the corresponding nodes. However, S_9 and S_5 are reached with a different order of firing, implying that the arrival times to that nodes are different.	69
6.5 Examples of TPN. All the transitions reported are uncontrollable. Transitions 3 and 4 in Example B are deterministic, while for both the examples transitions 1 and 2 are uniformly distributed.	70
6.6 Portions of the reachability trees of the temporal nets in Figure 6.5.	70
6.7 The depicted conflict is uncontrollable: all transitions leading to $S_{C1,2,\dots}$ are uncontrollable.	72
6.8 Steps involved for determining the distribution of the arrival time in S_1 of the Example A of Figure 6.6, and the same for S_2 of Example B. Notice that distributions G_f of the arrival time in the node preceding, equation (6.6), is not considered because for both the examples above, the transition leading to the node of interest begin to be enabled after arriving in S_f	73
6.9 The conditioned density of an exponential distribution is in turn the same exponential distribution.	74
6.10 PN model for actions executed by the robot.	77
6.11 PN model for actions executed by the human.	78
6.12 PN model for collaborative actions.	79
6.13 PN model for a mobile robot (cart) that transports items that are loaded manually and unloaded using a robot.	79
6.14 By superimposing the above nets, the pcTPN modelling the assembly in Figure 6.2 is obtained.	80
7.1 Schematic representation of the control architecture adopted for scheduling collaborative cells.	82

List of Figures

7.2 Sequence of operations required to obtain a finished product for the use case adopted to validate the best scenario approach scheduling. Red boxes refer to operations assigned to YUMI, while the blue one is the autonomous task of the human. The pink box indicates the collaborative action, executed simultaneously by both the human and the arms of YUMI. Feeders contain an infinite number of items, since they are externally fed when needed. 85

7.3 The pcTPN modelling the use case. Thick rectangles denote controllable transitions. Transitions \bar{t}_c and t_{ch} refer to the collaborative operation. As can be seen, the limited capacity of intermediate buffers is taken into account. 86

7.4 Cost curve for a feasible path, in case the arrival time s_f is known. c_H can be interpreted as the unit cost, for example $\$/s$, paid when the human is kept inactive (similarly c_R when maintaining the robots inactive). . . 87

7.5 The two rules adopted for back propagating particles of cost in a reachability tree. When considering the controllable conflict on the left, the edge leading to the node having the lowest β percentile of cost will be followed in case the system would arrive in the indicated node. Therefore, the other edge is in some way disabled by the optimal control policy. 91

7.6 Example of a PN compliant with the proposed modeling strategy. . . . 92

7.7 Cost aggregation for the single leaf of a reachability tree. 93

7.8 Excerpt of the reachability tree for the example in Fig. 7.6. Nodes correspond to firing events. Leaves are labeled with the probability distributions of waiting times for the human W_H (blue bars) and the robot W_R (yellow bars). Red segments correspond to the states where some places associated to waiting conditions of the human or the robot are marked (incurring in a cost). Filled [Empty] red nodes correspond to the firing of transitions that will mark [unmark] a red place. 93

7.9 Propagation of the costs for uncontrollable transitions: the probability distribution of the parent node is the union of the probability distributions of its children nodes. 95

7.10 Propagation of the costs for controllable transitions: the distribution of the costs is inherited from the child having the lowest cost. In this case firing transition t_1 is evaluated to be more convenient than firing t_2 , therefore the branch starting from the latter is pruned from the tree. . 95

7.11 Assembly addressed by off line simulations. 96

7.12 Cycle time distributions obtained in the off line comparative simulations: A refers to [93], B to [76] while C to the Monte Carlo scheduler in [19]. 96

7.13 Examples of computation of $\tilde{\alpha}$. For the left picture $\max_t (\tilde{\alpha}_{C_i}(t)) = 1$, as the support of the arrival time of the child is completely above the one of the father, while for the picture in the middle $\max_t (\tilde{\alpha}_{C_i}(t)) = 0$ as the opposite situation arises, meaning that for the situation in the middle, that node would have been removed from the children list. The situation depicted on the right is intermediate between the aforementioned cases. 97

7.14 Workspace shared by the human operator and YUMI. 101

7.15	In all the above figures the top line reports the actions executed by right arm of YUMI over time, the middle one refers to the actions executed by the left arm, while the bottom line is related to the actions executed by the human operator. The legenda related to the operations represented is depicted in Figure 7.16. Dotted gray lines indicate time instants at which the human starts to be available for a new collaboration, while dotted black line refers to instants at which YUMI begins the collaborative task (those two kinds of lines are coincident when considering the scheduling obtained with the use of pcTPN).	102
7.16	Legenda related to Figure 7.15.	102
7.17	The experimental setup with the two robots, the carts and the human operator.	102
7.18	Schematic representation and equivalent PN adopted for the use-case.	103
7.19	Distributions of the measured cycle times, for the two assembly patterns considered.	106
7.20	Distributions of the measured durations of some uncontrollable transitions of the net reported in Figure 7.18 in the middle (yellow) and at the end (blue) of the experiment.	106
7.21	Distributions of the human inactivity times, for the two assembly patterns considered.	107
7.22	Distributions of the agents inactivity times, for the two assembly patterns considered: the summations of all the idling times of the agents in the system is reported.	107
7.23	The evolution of the overall wait time of the agents, during the experiments of Group 1.A and Group 1.B. The reported values, refer to the summation of the idle times of YUMI, IRB 140 and the human operator. The dashed red curves are the regressed lines interpolating all the data.	108
7.24	Sequence of events occurred for two particular experiments: one from Group 1.A (top) and one from Group 1.B (bottom), see Figure 7.5 for the legend. The temporal duration of the activities performed by the agents is proportional to the length of the corresponding coloured bar (waiting activities are not reported). The dashed vertical lines refer to time instants when a new finite product is available (<i>i.e.</i> the end of the storing operation done by the IRB 140): the red ones referring to the experiment of Group 1.A while the blue ones to that of Group 1.B.	109
7.25	On the top, the layout adopted for the experiments. On the bottom left part, a detailed view of the layout: the violet area contains the stations used by the robots to perform the assigned intermediate assemblies, while the green ones are the buffers through which the human and the robots exchange components. The pictures on the bottom right part depicts the two products to be assembled.	111
7.26	On the top the actions required for the assembly of a single torch and the corresponding assembly flow, on the bottom the same for a clock.	112
7.27	Idling times of the operator. Every sample of the reported distributions refers to the idling measured within the assembly of a single product.	114

List of Figures

7.28 Measured cycle times. 114

8.1 Difference between a reactive approach and a proactive one. The motion resulting from a reactive approach guarantees the absence of collisions with a moving obstacle, but it's not globally optimized, since with a certain frequency is constantly recomputed considering the current position of the obstacle. On the opposite, the proactive path is computed from its start to its end once; by taking into account a prediction of the entire motion of the yellow obstacle. A combination of the two approaches is also possible. 118

8.2 On the left, (a) and (b), the kinematic model adopted to describe the human posture. For the base of the human, a unicycle model is considered, while for the arms a spherical joint is centred at the shoulder, with an additional joint located at the elbow. On the right, (c), an example of swept volumes bounding the future occupancy of the anatomical parts of the human. In blue the volume swept by the torso, in green those for the forearms, while in yellow those for the upper arms. 121

8.3 On the top, examples of cumulative distributions for Δ_k^+ and Δ_k^- (the kinematic chain is assumed to be made of a single joint), while on the bottom examples of swept volumes for a single joint mechanism, depicted on the left bottom corner, when assuming different percentiles for Δ_k^+ and Δ_k^- 124

8.4 Pipeline describing the developed approach. The picture on the right details the steps involved for updating h_{GP} , *i.e.* the model describing human motion. 125

8.5 Experimental set-up considered for the experiments. (a): YUMI of ABB, is visible in the centre; with a MICROSOFT KINECT vision system on top. (b): Locations of buffers containing the items required to assemble the box with the USB and the PCB board. (c): Some approximative trajectories followed by both the human (in clear green) and the robot (dark blue). The dashed orange area is the portion of space shared by the human and the robot, while the green shaded one is the portion of the space occupied by the operator's torso during the experiments. . . . 127

8.6 Distributions of ϵ^+ and ϵ^- , for joint q_2 and q_3 of the kinematic model reported in Figure 8.2. The legend of Figure 8.7 applies. The red horizontal line, divide conservative predictions from the ones for which a violation happens. Results for similar joints of both arms are condensed in a single boxplot. 128

8.7 Legend to consider for Figure 8.6 and 8.8 128

8.8 Distributions of ϵ^+ and ϵ^- , for joints $q_{1,4,5,6,7,8,9,10,11,12}$ (*i.e.* the rotating ones) of the kinematic model reported in Figure 8.2. Results for similar joints are condensed in a single boxplot. The same legend in Figure 8.7 applies. 129

8.9 Comparisons between the swept volumes obtained considering [107] (the blue ones on the first row) against those obtained applying the hybrid approach with $P = 4$ (the green ones on the second row), for two sampled instants of the logged data. 129

8.10 Statistics about the distances between skeletal points and predicted swept volumes, in case of violation of the predicted joint excursion.	129
8.11 The entire pipeline of the approach. A depth camera records the motion of the human, whose trajectories are then segmented and stored in different databases, one for each human action. Such samples are considered by GPDM to compute regressed trajectories for the human motion, which in turn are exploited to compute the corresponding probability clouds and the proactive paths.	132
8.12 Pictures on the left depict an example of regressed trajectory taken from two distinct views. The bounding box containing entirely the trajectory (red box in the figure) is considered for the definition of the g points contained in C . The volume swept by the arm of the operator, between two consecutive poses, is approximated by the dark green convex set in the middle, which in turn is approximated by an OctTree, indicated in the right with light green (only the OctTree of the upper part of the arm is reported in the figure).	134
8.13 Proactive planning.	136
8.14 The experimental setup. The left arm of YUMI is involved in the collaborative assembly.	137
8.15 The activities executed during time by both the human and the robot. Green vertical line indicates instants at which a proactive path is recomputed. Sk indicates the k -th invocation of the proactive planning algorithm. When $A(t) = 1$ the human was performing a_{S1} , while $A(t) = 2$ refers to a_{S2}	138
8.16 The pictures report a scalability analysis, varying the grid resolution G (see Section 8.2.2). The pictures on the top and on the bottom report the computational times for obtaining a single probability cloud and a single proactive path respectively. The picture in the middle reports the approximation error introduced when describing the human trajectories with different resolutions for the discrete grid. As can be seen, the computational times grows faster than the E_{cloud} decrease. Therefore, reducing G leads to minor computation times, without severely compromise the way the human motion is approximated.	139
8.17 On the top the distribution of the distance between the human and the robot during the experiments; while on the bottom, the distribution of the percentage of idle time.	139
8.18 The nominal path realizing r_1 from two different views. The blue curve refers to the trajectory of the end effector. Red shapes represent the fixed obstacles considered by STOMP. Green capsules are adopted to depict the robot links for the initial and the final pose of the path.	140

List of Figures

8.19 Subjects rate in percentage to the following quotes. R1: "The robot movements were unnatural and strange"; R2: "The interaction with the robot was fluent"; R3: "The interaction with the robot was safe"; R4: "The robot was able to efficiently forecast the human trajectories". Picture on the left refers to the group of participants for which pro active paths were executed. Picture on the right, to the group for which nominal paths were executed. 140

8.20 Examples of proactive paths obtained for some invocations of STOMP (Red shapes are the fixed obstacles populating the scene) during the experiments. For every column: on the top, the probabilities considered for the computation of the probabilistic cloud; in the middle and in the bottom, two distinct views of the proactive paths computed (cyan), compared with the initial nominal one (blue). The probability clouds considered for planning are depicted as a series of blue points whose intensity is proportional to the probabilities contained in γ 141

9.1 Skeletal approximation of the human silhouette. 145

9.2 On the left, an example of occlusion, blue dotted lines represent information gained from the depth map. On the right, estimation of the pose using PF, blue dots are the positions of particles in set \bar{S} , *i.e.* the one computed propagating set S of the previous step using the process equation. The object at step k is visible, then for the subsequent step goes under an occlusion. Red dots are used to indicate those particles not consistent with the depth map, which will not survive after the resampling step of PF. 148

9.3 Possible occlusion cases considered for left arm. The shapes of the occlusions are depicted as a gray area, while the the light purple area delimits the area admitted by the skeletal distances constraints. 149

9.4 Pipeline of the proposed approach. y_{k+1} contains the estimated position of skeletal points, acquired from a depth camera sensor. 151

9.5 The robotic cell considered for the experiment 152

9.6 The approaches considered to tackle occlusions. On the left, the Kalman filter formulation (KF), on the right the PF formulation. Red edges are those delimiting the workspace of the robot. In red are indicated the skeletal points estimated using KF or PF, while in blue are depicted the measurements retrieved from the MICROSOFT KINECT (values are always returned, even though their status indicate that is an occlusion present). KF goes in open loop when occlusion occur. This reflects in the growing of the uncertainty covariance ellipsoid, which can occupies regions inconsistent with the occlusion. 153

9.7 Histograms relative to the robot cycle time for the approaches considered 153

C.1 Example of graph made of 4 variables: A, B, C and D . α, β, γ and δ are assumed as weights for the exponential potentials $\Psi_{AC}, \Psi_{AB}, \Psi_{CD}$ and Ψ_B respectively. 171

C.2	Example of graph adopted for explaining the message passing algorithm. Below are reported the messages to compute for obtaining the marginal probability of variable x_1	172
C.3	On the top the general mechanism involved in the message computation; on the bottom the simplification of the graph considering the computed message.	174
C.4	When variable C become an evidence, is temporary deleted from the graph, replaced by messages.	176
C.5	Steps involved for computing the messages of the politree represented at the top. The leaves are the first nodes for which the outgoing messages can be computed.	177
C.6	Steps involved for computing the messages on a loopy graph: after computing the messages outgoing from the leaves, a deadlock is reached since no further messages are computable.	178
C.7	Example of graph adopted. When the evidences are retrieved, the messages computed by making use of the message passing algorithm are reported below.	178
C.8	Example of structure having many exponential potentials sharing the same modulating weight: w_1 is simultaneously involved into Ψ_{AC} and Ψ_{BC} , while w_2 is shared among the potentials connecting $A_{1,2,3}$ to $B_{1,2,3}$ respectively.	182
D.1	Examples of a fuzzy convex set (on the left) and a non convex one (on the right). When considering the convex set, all the possible alpha-cuts are compact set, which is not verified for the set in the right.	186
D.2	Example of L (on the left) and G (on the right) of a triangular fuzzy number.	187

List of Tables

3.1	Percentages of false positives and true negatives are with respect to the total number of times the operators went to goal 4 for the upper table. The lower one refers to goal 1.	23
5.1	Results obtained when applying operators \mathcal{I} and \mathcal{O} on the series Y reported at the top.	48
6.1	Descriptive formalism adopted in this work and equivalent PN blocks for automatic translation.	76
7.1	Timing parameters of the transitions of the TPN of Fig. 7.6.	92
7.2	Sampled trajectories for the system in Figure 7.6. On the left the sequence of events involved in every single trajectory, with the corresponding times. On the right the progressive update of the reachability tree.	94
7.3	Excerpt of the PN of Figure 7.18 and example of synchronization between the station and the PN model used for scheduling.	105
7.4	Composition of the times spent by agents performing the assigned tasks, see Figure 7.5 for the legend. For each task, the overall time spent doing that action is considered for creating the proposed histogram charts, summing the values of all the experiments in a specific group.	109
7.5	Color legend for Figures 7.4 and 7.24.	110
C.1	Value assumed by the energy function E , when having $X = 0$ and $Y = 1$ as evidences.	175
C.2	Factors involved in the graph of Figure C.7.	175

Bibliography

- [1] Arash Ajoudani, Andrea Maria Zanchettin, Serena Ivaldi, Alin Albu-Schäffer, Kazuhiro Kosuge, and Ousama Khatib. Progress and prospects of the human–robot collaboration. *Autonomous Robots*, pages 1–19, 2017.
- [2] Amin Amini and Navid Nikraz. A method for constructing non-isosceles triangular fuzzy numbers using frequency histogram and statistical parameters. *Soft Computing in Civil Engineering*, 1(1):65–85, 2017.
- [3] Jimmy Baraglia, Maya Cakmak, Yukie Nagai, Rajesh PN Rao, and Minoru Asada. Efficient human-robot collaboration: when should a robot take initiative? *The International Journal of Robotics Research*, pages 1–17, 2017.
- [4] F. Basile, M. P. Cabasino, and C. Seatzu. State estimation and fault diagnosis of labeled time petri net systems with unobservable transitions. *IEEE Transactions on Automatic Control*, 60(4):997–1009, April 2015.
- [5] Roman Bednarik, Hana Vrzakova, and Michal Hradis. What do you want to do next: a novel approach for intent prediction in gaze-based interaction. In *Proceedings of the symposium on eye tracking research and applications*, pages 83–90. ACM, 2012.
- [6] Richard Bellman. A markovian decision process. *Indiana Univ. Math. J.*, 6:679–684, 1957.
- [7] Bernard Berthomieu and Michel Diaz. Modeling and verification of time dependent systems using time Petri nets. *IEEE Transactions on Software Engineering*, 17(3):259–273, 1991.
- [8] Karin Bogner, Ulrich Pferschy, Roland Unterberger, and Herwig Zeiner. Optimised scheduling in human–robot collaboration—a use case in the assembly of printed circuit boards. *International Journal of Production Research*, 56(16):5522–5540, 2018.
- [9] Matthieu Bray, Esther Koller-Meier, and Luc Van Gool. Smart particle filtering for 3d hand tracking. In *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, pages 675–680. IEEE, 2004.
- [10] Giulia Bruno and Dario Antonelli. Dynamic task classification and assignment for the management of human-robot collaborative teams in workcells. *International Journal of Advanced Manufacturing Technology*, 98(9-12):2415–2427, 2018.
- [11] Miguel Carrasco and Xavier Clady. Prediction of user’s grasping intentions based on eye-hand coordination. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 4631–4637. IEEE, 2010.
- [12] Andrea Casalino, Davide Bazzi, Andrea Maria Zanchettin, and Paolo Rocco. Optimal proactive path planning for collaborative robots in industrial contexts. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6540–6546. IEEE, 2019.
- [13] Andrea Casalino, Alberto Brameri, Andrea Maria Zanchettin, and Paolo Rocco. Adaptive swept volumes generation for human-robot coexistence using gaussian processes. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019.
- [14] Andrea Casalino, Filippo Cividini, Andrea Maria Zanchettin, Luigi Piroddi, and Paolo Rocco. Human-robot collaborative assembly: a use-case application. *IFAC-PapersOnLine*, 51(11):194–199, 2018.

Bibliography

- [15] Andrea Casalino, Sebastian Guzman, Andrea Maria Zanchettin, and Paolo Rocco. Human pose estimation in presence of occlusion using depth camera sensors, in human-robot coexistence scenarios. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–7. IEEE, 2018.
- [16] Andrea Casalino, Eleonora Mazzocca, Maria Grazia Di Giorgio, Andrea Maria Zanchettin, and Paolo Rocco. Task scheduling for human-robot collaboration with uncertain duration of tasks: a fuzzy approach. In *2019 International Conference on Control, Mechatronics and Automation (ICCMA)*. IEEE, 2019.
- [17] Andrea Casalino, Costanza Messeri, Maria Pozzi, Andrea Maria Zanchettin, Paolo Rocco, and Domenico Praticchizzo. Operator awareness in human–robot collaboration through wearable vibrotactile feedback. *IEEE Robotics and Automation Letters*, 3(4):4289–4296, 2018.
- [18] Andrea Casalino, Paolo Rocco, and Maria Prandini. Hybrid control of manipulators in human-robot coexistence scenarios. In *2018 Annual American Control Conference (ACC)*, pages 1172–1177. IEEE, 2018.
- [19] Andrea Casalino, Andrea Zanchettin, Luigi Piroddi, and Paolo Rocco. Optimal scheduling of human-robot collaborative assembly operations with time petri nets. *IEEE Transactions on Automation Science and Engineering*, 2019.
- [20] Adamo Castelnovo, Luca Ferrarini, and Luigi Piroddi. An incremental Petri net-based approach to the modeling of production sequences in manufacturing systems. *IEEE Transactions on Automation Science and Engineering*, 4(3):424–434, 2007.
- [21] Nicola Maria Ceriani, Andrea Maria Zanchettin, Paolo Rocco, Andreas Stolt, and Anders Robertsson. Reactive task adaptation based on hierarchical constraints classification for safe industrial robots. *IEEE/ASME Transactions on Mechatronics*, 20(6):2935–2949, 2015.
- [22] Crystal Chao and Andrea Thomaz. Timed Petri nets for fluent turn-taking over multimodal interaction resources in human-robot collaboration. *International Journal of Robotics Research*, 35(11):1330–1353, 2016.
- [23] Fei Chen, Kosuke Sekiyama, Ferdinando Cannella, and Toshio 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.
- [24] Fei Chen, Kosuke Sekiyama, Jian Huang, Baiqing Sun, Hironobu Sasaki, and Toshio Fukuda. An assembly strategy scheduling method for human and robot coordinated cell manufacturing. *International Journal of Intelligent Computing and Cybernetics*, 4:487–510, 11 2011.
- [25] Andrea Cherubini, Robin Passama, André Crosnier, Antoine Lasnier, and Philippe Fraisse. Collaborative manufacturing with physical human–robot interaction. *Robotics and Computer-Integrated Manufacturing*, 40:1–13, 2016.
- [26] Wai Ki Ching, Eric S. Fung, and Michael K. Ng. Higher-order Markov chain models for categorical data sequences. *Naval Research Logistics (NRL)*, 51(4):557–574, 2004.
- [27] Giovanni Chiola, Marco Ajmone Marsan, Gianfranco Balbo, and Gianni Conte. Generalized stochastic petri nets: A definition at the net level and its implications. *IEEE Transactions on software engineering*, 19(2):89–107, 1993.
- [28] Valerie Cross. Semantic similarity: a key to ontology alignment. In *OM@ ISWC*, pages 61–65, 2018.
- [29] Jin Dai, Alessandro Benini, Hai Lin, Panos J Antsaklis, Matthew J Rutherford, and Kimon P Valavanis. Learning-based formal synthesis of cooperative multi-agent systems with an application to robotic coordination. In *Control and Automation (MED), 2016 24th Mediterranean Conference on*, pages 1008–1013. IEEE, 2016.
- [30] Laécio Carvalho de Barros, Rodney Carlos Bassanezi, and Weldon Alexander Lodwick. The extension principle of zadeh and fuzzy numbers. In *A First Course in Fuzzy Logic, Fuzzy Dynamical Systems, and Biomathematics*, pages 23–41. Springer, 2017.
- [31] Luis Miguel de Campos Ibanez and Antonio Gonzalez Munoz. A subjective approach for ranking fuzzy numbers. *Fuzzy sets and systems*, 29(2):145–153, 1989.
- [32] Ana M Djuric, RJ Urbanic, and JL Rickli. A framework for collaborative robot (cobot) integration in advanced manufacturing systems. *SAE International Journal of Materials and Manufacturing*, 9(2):457–464, 2016.
- [33] Jill L Drury, Jean Scholtz, and Holly A Yanco. Awareness in human-robot interactions. In *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, volume 1, pages 912–918. IEEE, 2003.
- [34] A. Elfes, C. R. Weisbin, H. Hua, J. H. Smith, J. Mrozinski, and K. Shelton. The huron task allocation and scheduling system: Planning human and robot activities for lunar missions. In *2008 World Automation Congress*, pages 1–8, Sept 2008.

- [35] C. Fiedler and W. Meyer. Transitory assembly scheduling based on deterministic correlation functions. In *2007 IEEE International Symposium on Assembly and Manufacturing*, pages 239–244, July 2007.
- [36] F. Flacco, T. Kroger, 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, May 2012.
- [37] James Foulds, Nicholas Navaroli, Padhraic Smyth, and Alexander Ihler. Revisiting map estimation, message passing and perfect graphs. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 278–286, 2011.
- [38] Broz Frank and Gordon Geoffrey. Better motion prediction for people-tracking. 03 2004.
- [39] Brendan J Frey. Extending factor graphs so as to unify directed and undirected graphical models. In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, pages 257–264. Morgan Kaufmann Publishers Inc., 2002.
- [40] David E Goldberg. Genetic algorithms in search, optimization, and machine learning, addison wesley, reading, ma. *SUMMARY THE APPLICATIONS OF GA-GENETIC ALGORITHM FOR DEALING WITH SOME OPTIMAL CALCULATIONS IN ECONOMICS*, 1989.
- [41] Matthew C Gombolay, Ronald Wilcox, and Julie A Shah. Fast scheduling of multi-robot teams with temporospatial constraints. In *Robotics: Science and Systems*, 2013.
- [42] Manish K. Govil and Michael C. Fu. Queueing theory in manufacturing: A survey. *Journal of Manufacturing Systems*, 18(3):214–240, 1999.
- [43] Fredrik Gustafsson. Particle filter theory and practice with positioning applications. *IEEE Aerospace and Electronic Systems Magazine*, 25(7):53–82, 2010.
- [44] Sami Haddadin, Simon Haddadin, Augusto Khoury, Tim Rokahr, Sven Parusel, Rainer Burgkart, Antonio Bicchi, and Alin Albu-Schaffer. On making robots understand safety: Embedding injury knowledge into control. *Int. J. Rob. Res.*, 31(13):1578–1602, November 2012.
- [45] Sami Haddadin, Michael Suppa, Stefan Fuchs, Tim Bodenmüller, Alin Albu-Schäffer, and Gerd Hirzinger. *Towards the Robotic Co-Worker*, pages 261–282. Springer Berlin Heidelberg, 2011.
- [46] K. S. Hale and K. M. Stanney. Deriving haptic design guidelines from human physiological, psychophysical, and neurological foundations. *IEEE Computer Graphics and Applications*, 24(2):33–39, March 2004.
- [47] Kelsey P Hawkins, Nam Vo, Shray Bansal, and Aaron F Bobick. Probabilistic human action prediction and wait-sensitive planning for responsive human-robot collaboration. In *Humanoid Robots (Humanoids), 2013 13th IEEE-RAS International Conference on*, pages 499–506. IEEE, 2013.
- [48] Zhou He, Zhiwu Li, and Alessandro Giua. Cycle time optimization of deterministic timed weighted marked graphs by transformation. *IEEE Transactions on Control Systems Technology*, 25(4):1318–1330, July 2017.
- [49] Zhou He, ZW Li, and Alessandro Giua. Optimization of deterministic timed weighted marked graphs. *IEEE Trans. Autom. Sci. Eng.*, 2016.
- [50] Chien-Ming Huang and Bilge Mutlu. Anticipatory robot control for efficient human-robot collaboration. In *Human-Robot Interaction (HRI), 2016 11th ACM/IEEE International Conference on*, pages 83–90. IEEE, 2016.
- [51] A. Papandreou-Suppappola I. Kyriakides, D. Morrell. Multiple target tracking with constrained motion using particle filtering methods. 01 2006.
- [52] Tariq Iqbal, Samantha Rack, and Laurel D Riek. Movement coordination in human–robot teams: A dynamical systems approach. *IEEE Transactions on Robotics*, 32(4):909–919, 2016.
- [53] Michael Isard and Andrew Blake. A mixed-state condensation tracker with automatic model-switching. In *Computer Vision, 1998. Sixth International Conference on*, pages 107–112. IEEE, 1998.
- [54] Lars Johannsmeier and Sami Haddadin. A hierarchical human-robot interaction-planning framework for task allocation in collaborative industrial assembly processes. *IEEE Robotics and Automation Letters*, 2(1):41–48, 2017.
- [55] Vijay John, Emanuele Trucco, and Spela Ivekovic. Markerless human articulated tracking using hierarchical particle swarm optimisation. *Image and Vision Computing*, 28(11):1530 – 1547, 2010.
- [56] Teegan Johnson, Gilbert Tang, Sarah R Fletcher, and Phil Webb. Investigating the effects of signal light position on human workload and reaction time in human-robot collaboration tasks. In *Advances in Ergonomics of Manufacturing: Managing the Enterprise of the Future*, pages 207–215. Springer, 2016.

Bibliography

- [57] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. Stomp: Stochastic trajectory optimization for motion planning. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4569–4574. IEEE, 2011.
- [58] Harmish Khambhaita and Rachid Alami. Viewing robot navigation in human environment as a cooperative activity. *Springer Proceedings in Advanced Robotics*, 10, 2017.
- [59] Jun Kinugawa, Akira Kanazawa, Shogo Arai, and Kazuhiro Kosuge. Adaptive task scheduling for an assembly task coworker robot based on incremental learning of human’s motion patterns. *IEEE Robotics and Automation Letters*, 2(2):856–863, 2017.
- [60] Chris L Kleinke. Gaze and eye contact: a research review. *Psychological bulletin*, 100(1):78, 1986.
- [61] Hema S Koppula and Ashutosh Saxena. Anticipating human activities using object affordances for reactive robotic response. *IEEE transactions on pattern analysis and machine intelligence*, 38(1):14–29, 2016.
- [62] J. Krüger, T.K. Lien, and A. Verl. Cooperation of human and machines in assembly lines. *CIRP Annals*, 58(2):628 – 646, 2009.
- [63] J. Kruger, G. Schreck, and D. Surdilovic. Dual arm robot for flexible and cooperative assembly. *CIRP Annals*, 60(1):5 – 8, 2011.
- [64] Dana Kulić and Elizabeth Croft. Pre-collision safety strategies for human-robot interaction. *Autonomous Robots*, 22(2):149–164, 2007.
- [65] Woo Young Kwon and Il Hong Suh. A temporal bayesian network with application to design of a proactive robotic assistant. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3685–3690. IEEE, 2012.
- [66] Przemyslaw A Lasota and Julie A Shah. Analyzing the effects of human-aware motion planning on close-proximity human–robot collaboration. *Human factors*, 57(1):21–33, 2015.
- [67] Jay Lee, Behrad Bagheri, and Hung-An Kao. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing letters*, 3:18–23, 2015.
- [68] Claus Lenz, Suraj Nair, Markus Rickert, Alois Knoll, Wolfgang Rosel, Jurgen Gast, Alexander Bannat, and Frank Wallhoff. Joint-action for humans and industrial robots for assembly tasks. In *Robot and Human Interactive Communication, 2008. RO-MAN 2008. The 17th IEEE International Symposium on*, pages 130–135. IEEE, 2008.
- [69] Claus Lenz, Alice Sotzek, Thorsten Röder, Helmuth Radrich, Alois Knoll, Markus Huber, and Stefan Glasauer. Human workflow analysis using 3d occupancy grid hand tracking in a human-robot collaboration scenario. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 3375–3380. IEEE, 2011.
- [70] Kang Li and Yun Fu. Prediction of human activity by discovering temporal sequence patterns. *IEEE transactions on pattern analysis and machine intelligence*, 36(8):1644–1657, 2014.
- [71] Yanan Li, Keng Peng Tee, Rui Yan, Wei Liang Chan, and Yan Wu. A framework of human–robot coordination based on game theory and policy iteration. *IEEE Transactions on Robotics*, 32(6):1408–1418, 2016.
- [72] Adam Lipowski and Dorota Lipowska. Roulette-wheel selection via stochastic acceptance. *Physica A: Statistical Mechanics and its Applications*, 391(6):2193 – 2196, 2012.
- [73] Xianghang Liu and Jian Zhang. Active learning for human action recognition with gaussian processes. In *Image Processing (ICIP), 2011 18th IEEE International Conference on*, pages 3253–3256. IEEE, 2011.
- [74] Z. Liu, L. Zhou, H. Leung, and H. P. H. Shum. Kinect posture reconstruction based on a local mixture of gaussian process models. *IEEE Transactions on Visualization and Computer Graphics*, 22(11):2437–2450, Nov 2016.
- [75] Daniel Losch and Jurgen Rossmann. Visual programming and development of manufacturing processes based on hierarchical Petri nets. In *3rd International Conference on Soft Computing & Machine Intelligence*, pages 154–158, 2016.
- [76] Ping Lou, Quan Liu, Zude Zhou, Huaiqing Wang, and Sherry Sun. Multi-agent-based proactive reactive scheduling for a job shop. 59, 03 2012.
- [77] Ruikun Luo, Rafi Hayne, and Dmitry Berenson. Unsupervised early prediction of human reaching for human–robot collaboration in shared workspaces. *Autonomous Robots*, pages 1–18, 2017.
- [78] Vito Magnanimo, Matteo Saveriano, Silvia Rossi, and Dongheui Lee. A bayesian approach for task recognition and future human activity prediction. In *Robot and Human Interactive Communication, 2014 RO-MAN: The 23rd IEEE International Symposium on*, pages 726–731. IEEE, 2014.

- [79] J. Mainprice, E. Akin Sisbot, L. Jaillet, J. Cortes, R. Alami, and T. Simeon. Planning human-aware motions using a sampling-based costmap planner. In *2011 IEEE Int. Conf. on Robotics and Automation*, pages 5012–5017, May 2011.
- [80] Alexandros Makris, Nikolaos Kyriazis, and Antonis A. Argyros. Hierarchical particle filtering for 3d hand tracking. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2015.
- [81] Sotiris Makris, Panagiotis Karagiannis, Spyridon Koukas, and Aleksandros-Stereos Matthaiakis. Augmented reality system for operator support in human–robot collaborative assembly. *CIRP Annals-Manufacturing Technology*, 65(1):61–64, 2016.
- [82] Donald JR Meagher. *Octree encoding: A new technique for the representation, manipulation and display of arbitrary 3-d objects by computer*. Electrical and Systems Engineering Department Rensselaer Polytechnic Institute Image Processing Laboratory, 1980.
- [83] L. S. H. De Mello and Arthur C. Sanderson. Representations of mechanical assembly sequences. *IEEE Transactions on Robotics and Automation*, 7(2):211–227, 1991.
- [84] Brad L. Miller, Brad L. Miller, David E. Goldberg, and David E. Goldberg. Genetic algorithms, tournament selection, and the effects of noise. *Complex Systems*, 9:193–212, 1995.
- [85] Abdullah Mohammed, Bernard Schmidt, and Lihui Wang. Active collision avoidance for human–robot collaboration driven by vision sensors. *International Journal of Computer Integrated Manufacturing*, 30(9):970–980, 2017.
- [86] Michael K. Molloy. Performance analysis using stochastic Petri nets. *IEEE Transactions on Computers*, 31(9):913–917, 1982.
- [87] László Monostori, Botond Kádár, T Bauernhansl, S Kondoh, S Kumara, G Reinhart, O Sauer, G Schuh, W Sihn, and K Ueda. Cyber-physical systems in manufacturing. *Cirp Annals*, 65(2):621–641, 2016.
- [88] Rafael Mosberger and Henrik Andreasson. An inexpensive monocular vision system for tracking humans in industrial environments. In *ICRA*, pages 5850–5857, 2013.
- [89] Matteo Munaro, Christopher Lewis, David Chambers, Paul Hvass, and Emanuele Menegatti. Rgb-d human detection and tracking for industrial environments. In *Intelligent Autonomous Systems 13*, pages 1655–1668. Springer, 2016.
- [90] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, Apr 1989.
- [91] Kevin Patrick Murphy and Stuart Russell. Dynamic bayesian networks: representation, inference and learning. 2002.
- [92] Nima Najmaei, Mehrdad R Kermani, and Mohammed 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.
- [93] Nikolaos Nikolakis, Niki Kousi, George Michalos, and Sotiris Makris. Dynamic scheduling of shared human-robot manufacturing operations. *Procedia CIRP*, 72:9–14, 2018.
- [94] Ben North, Andrew Blake, Michael Isard, and Jens Rittscher. Learning and classification of complex dynamics. *IEEE Transactions on pattern analysis and machine intelligence*, 22(9):1016–1034, 2000.
- [95] Tetsuya Ogata, Shigeki Sugano, and Jun Tani. Open-end human–robot interaction from the dynamical systems perspective: mutual adaptation and incremental learning. *Advanced Robotics*, 19(6):651–670, 2005.
- [96] Vladimir Pavlovic, James M Rehg, and John MacCormick. Learning switching linear models of human motion. In *Advances in neural information processing systems*, pages 981–987, 2001.
- [97] Stefania Pellegrinelli, Federico Lorenzo Moro, Nicola Pedrocchi, Lorenzo Molinari Tosatti, and Tullio Tolio. A probabilistic approach to workspace sharing for human–robot cooperation in assembly tasks. *CIRP Annals*, 65(1):57–60, 2016.
- [98] Stefania Pellegrinelli and Nicola Pedrocchi. Estimation of robot execution time for close proximity human-robot collaboration. *Integrated Computer-Aided Engineering*, (Preprint):1–16, 2017.
- [99] Kaare Brandt Petersen, Michael Syskind Pedersen, et al. The matrix cookbook. *Technical University of Denmark*, 7(15):510, 2008.
- [100] Elizabeth Kathleen Phillips and Florian 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.

Bibliography

- [101] F. Pini, F. Leali, and M. Ansaloni. A systematic approach to the engineering design of a hrc workcell for bio-medical product assembly. In *2015 IEEE 20th Conference on Emerging Technologies Factory Automation (ETFA)*, pages 1–8, Sept 2015.
- [102] Joanna Zietkiewicz Piotr Kozierski, Marcin Lis. Resampling in particle filtering , comparison. 2013.
- [103] Adrian Raftery and Simon Tavaré. Estimation and modelling repeated patterns in high order markov chains with the mixture transition distribution model. *Applied Statistics*, pages 179–199, 1994.
- [104] Adrian E Raftery. A model for high-order markov chains. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 528–539, 1985.
- [105] 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, May 2014.
- [106] M. Ragaglia, A. M. Zanchettin, and P. Rocco. Safety-aware trajectory scaling for human-robot collaboration with prediction of human occupancy. In *2015 Int. Conf. on Advanced Robotics*, pages 85–90, July 2015.
- [107] Matteo Ragaglia, Andrea Maria Zanchettin, and Paolo Rocco. Trajectory generation algorithm for safe human-robot collaboration based on multiple depth sensor measurements. *Mechatronics*, 2018.
- [108] SM Mizanoor Rahman, Behzad Sadrfaridpour, and Yue Wang. Trust-based optimal subtask allocation and model predictive control for human-robot collaborative assembly in manufacturing. In *ASME 2015 Dynamic Systems and Control Conference*, pages V002T32A004–V002T32A004. American Society of Mechanical Engineers, 2015.
- [109] Peter J Ramadge and W Murray Wonham. Supervisory control of a class of discrete event processes. *SIAM journal on control and optimization*, 25(1):206–230, 1987.
- [110] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Advanced lectures on machine learning*, pages 63–71. Springer, 2004.
- [111] Harish Chaandar Ravichandar and Ashwin 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.
- [112] Dominik Riedelbauch, Tobias Werner, and Dominik Henrich. Enabling domain experts to model and execute tasks in flexible human-robot teams. In *Tagungsband des 2. Kongresses Montage Handhabung Industrieroboter*, pages 13–22. 2017.
- [113] S Robla-Gómez, Victor M Becerra, JR Llata, Esther González-Sarabia, Carlos Torre-Ferrero, and J Pérez-Oria. Working together: a review on safe human-robot collaboration in industrial environments. *IEEE Access*, 5:26754–26773, 2017.
- [114] Dana Ron, Yoram Singer, and Naftali Tishby. The power of amnesia: Learning probabilistic automata with variable memory length. *Machine Learning*, 25(2-3):117–149, 1996.
- [115] Anas Salmi, Pierre David, J.D. Summers, and Eric Blanco. A modelling language for assembly sequences representation, scheduling and analyses. *International Journal of Production Research*, 52(13):3986–4006, 2014.
- [116] Riccardo Schiavi, Antonio Bicchi, and Fabrizio Flacco. Integration of active and passive compliance control for safe human-robot coexistence. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 259–264. IEEE, 2009.
- [117] Xinguang Shao, Biao Huang, and Jong Min Lee. Constrained bayesian state estimation: a comparative study and a new particle filter based approach. *Journal of Process Control*, 20(2):143 – 157, 2010.
- [118] Zhongshun Shi, Longfei Wang, Pai Liu, and Leyuan Shi. Minimizing completion time for order scheduling: Formulation and heuristic algorithm. *IEEE Transactions on Automation Science and Engineering*, 2015.
- [119] Leonid Sigal. *Human Pose Estimation*, pages 362–370. Springer US, Boston, MA, 2014.
- [120] José Reinaldo Silva and Pedro MG del Foyo. Timed petri nets. In *Petri Nets-Manufacturing and Computer Science*. IntechOpen, 2012.
- [121] Emrah Akin Sisbot, Luis F Marin-Urias, Rachid Alami, and Thierry Simeon. A human aware mobile robot motion planner. *IEEE Transactions on Robotics*, 23(5):874–883, 2007.
- [122] Aaron St Clair and Maja Mataric. How robot verbal feedback can improve team performance in human-robot task collaborations. In *Proceedings of the tenth annual acm/ieee international conference on human-robot interaction*, pages 213–220. ACM, 2015.

- [123] Holger Täubig, Berthold Bäuml, and Udo Frese. Real-time swept volume and distance computation for self collision detection. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 1585–1592. IEEE, 2011.
- [124] Panagiota Tsarouchi, Alexandros-Stereos Matthaiakis, Sotiris Makris, and George Chryssolouris. On a human-robot collaboration in an assembly cell. *International Journal of Computer Integrated Manufacturing*, 30(6):580–589, 2017.
- [125] Raquel Urtasun, David J Fleet, and Pascal Fua. 3d people tracking with gaussian process dynamical models. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 238–245. IEEE, 2006.
- [126] Gino Van Den Bergen. Proximity queries and penetration depth computation on 3d game objects. In *Game developers conference*, volume 170, 2001.
- [127] Jack M Wang, David J Fleet, and Aaron Hertzmann. Gaussian process dynamical models for human motion. *IEEE transactions on pattern analysis and machine intelligence*, 30(2):283–298, 2008.
- [128] Z. Wang, G. Liu, and G. Tian. Human skeleton tracking using information weighted consensus filter in distributed camera networks. In *2017 Chinese Automation Congress (CAC)*, pages 4640–4644, Oct 2017.
- [129] Ziyou Wang, Jun Kinugawa, Hongbo Wang, and Kosuge Kazahiro. A human motion estimation method based on gp-ukf. In *Information and Automation (ICIA), 2014 IEEE International Conference on*, pages 1228–1232. IEEE, 2014.
- [130] C. Yang, C. Zeng, Y. Cong, N. Wang, and M. Wang. A learning framework of adaptive manipulative skills from human to robot. *IEEE Transactions on Industrial Informatics*, pages 1–9, 2018.
- [131] Mao Ye, Qing Zhang, Liang Wang, Jiejie Zhu, Ruigang Yang, and Juergen Gall. *A Survey on Human Motion Analysis from Depth Data*, pages 149–187. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
- [132] Jonathan S Yedidia, William T Freeman, and Yair Weiss. Understanding belief propagation and its generalizations. *Exploring artificial intelligence in the new millennium*, 8:236–239, 2003.
- [133] Lotfi Asker Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy sets and systems*, 1(1):3–28, 1978.
- [134] A. M. Zanchettin and P. Rocco. Path-consistent safety in mixed human-robot collaborative manufacturing environments. In *2013 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 1131–1136, Nov 2013.
- [135] 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)*, pages 6595–6600, Sept 2017.
- [136] Andrea Zanchettin, Andrea Casalino, Luigi Piroddi, and Paolo Rocco. Prediction of human activity patterns for human-robot collaborative assembly tasks. *IEEE Transactions on Industrial Informatics*, 2018.
- [137] Alessandro Zanella, Alessandro Cisi, Marco Costantino, Massimo Di Pardo, Giorgio Pasquettaz, and Giulio Vivo. Criteria definition for the identification of HRC use cases in automotive manufacturing. *Procedia Manufacturing*, 11:372–379, 2017.
- [138] L. Zhang, J. Sturm, D. Cremers, and D. Lee. Real-time human motion tracking using multiple depth cameras. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2389–2395, Oct 2012.
- [139] Zhijun Zhang, Aryel Beck, and Nadia Magnenat-Thalmann. Human-like behavior generation based on head-arms model for robot tracking external targets and body parts. *IEEE transactions on cybernetics*, 45(8):1390–1400, 2015.