Robotics and Autonomous Systems 116 (2019) 162–180

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Cobot programming for collaborative industrial tasks: An overview

Shirine El Zaatari [a], Mohamed Marei [a], Weidong Li [a,*], Zahid Usman [b]

[a] Faculty of Engineering, Environment and Computing, Coventry University, UK
[b] Rolls Royce, UK

ARTICLE INFO

Article history:
Received 24 September 2018
Received in revised form 16 January 2019
Accepted 11 March 2019
Available online 18 March 2019

Keywords:
Human–robot collaboration
Intuitive programming
Human-awareness
Cobot

ABSTRACT

Collaborative robots (cobots) have been increasingly adopted in industries to facilitate human–robot collaboration. Despite this, it is challenging to program cobots for collaborative industrial tasks as the programming has two distinct elements that are difficult to implement: (1) an intuitive element to ensure that the operations of a cobot can be composed or altered dynamically by an operator, and (2) a human-aware element to support cobots in producing flexible and adaptive behaviours dependent on human partners. In this area, some research works have been carried out recently, but there is a lack of a systematic summary on the subject. In this paper, an overview of collaborative industrial scenarios and programming requirements for cobots to implement effective collaboration is given. Then, detailed reviews on cobot programming, which are categorised into communication, optimisation, and learning, are conducted. Additionally, a significant gap between cobot programming implemented in industry and in research is identified, and research that works towards bridging this gap is pinpointed. Finally, the future directions of cobots for industrial collaborative scenarios are outlined, including potential points of extension and improvement.

1. Introduction

Manufacturing in the Industry 4.0 era necessitates rapid, proactive responses to ever-changing consumers' demands. This had led to a trend of mass customisation, where certain aspects of the product, hence manufacturing processes, are tailored to meet the requirements of individual customers. Meanwhile, manufacturers need to continuously improve sustainability, production efficiency and quality throughout the product life cycle to ensure their competitive edge. Industrial automation is capable of maintaining high efficiency and repeatability for mass production. However, it lacks flexibility to deal with uncertainties in work spaces resulting from mass customisation. While humans, in such situations, can deal with such uncertainties and variability, they are restricted by their physical capabilities, in terms of repeatability, physical strength, stamina, speed etc. [1]. These limitations often result in reduced efficiency and quality [2]. A balance of automation and flexibility is thus required to achieve these overarching manufacturing goals during mass customisation. That promotes research in combining the benefits of automation and manual labour. This research has culminated in Human–Robot Collaboration (HRC), a promising robotics discipline focusing on enabling robots and humans to operate jointly to complete collaborative tasks.

HRC refers to application scenarios where a robot, usually a collaborative robot (cobot), and a human occupy the same workspace and interact to accomplish collaborative tasks [3]. Following the introduction of the UR5, a cobot produced by Universal Robotics in 2008 [4], industrial interest in applying HRC and cobots to factory floors has escalated. Many other robotics manufacturers, such as KUKA, ABB and Rethink Robotics, have also developed their own cobots, each tackling a particular niche. An overview of cobots on the market is discussed [5], comparing their costs, payload capabilities, and safety features. Since cobots are built for close-proximity interaction with humans, they must adhere to stringent safety requirements, such as power and speed limiting, soft padding, and the absence of trap points (i.e. points that can trap body parts or clothing) (ISO-TS 15066 [6]). Considering the distinguishing characteristics of cobots over regular robots, they are envisioned to pave the way for mass customisation, decrease required floor space, increase product quality and production efficiency and improve working conditions for humans [1,7].

Relevant research in HRC and cobots has revolved around enhancing particular enabling functions like visual perception, action recognition, intent prediction, safe on-line motion planning, etc. These technologies enable *human-awareness*, which result in flexible cobot behaviour as opposed to traditional fixed action-sequence cobot programs. Another line of research has revolved around Learning from Demonstration (LfD), Reinforcement Learning (RL), human–robot communication, collaborative task

* Corresponding author.
E-mail address: weidong.li@coventry.ac.uk (W. Li).

https://doi.org/10.1016/j.robot.2019.03.003
0921-8890/

semantics, etc. These fields enable *intuitive* cobot programming, allowing non-expert operators to create and alter robot programs quickly and intuitively. This paper explores the union between these two research directions, resulting in *human-aware, intuitive* robot programs for collaborative industrial tasks. Imbuing cobots with flexibility, reliability and autonomy is indeed a persistent research bottleneck for HRC scenarios in industry and elsewhere.

It is noted that several papers have provided related literature reviews, such as on HRC applications [8], methods for safe HRC [9] and more specific topics such as LfD [10,11], gesture recognition [12] and Augmented Reality [13]. Bauer et al. reviewed the technologies enabling HRC such as machine learning, action planning and intention estimation [14], which are all relevant to the programming of cobots. However, their review only covers works until 2008, prompting an update considering the surge of relevant recent works.

The goal of this paper is to provide research communities with a guide on deploying cobots in collaborative industrial scenarios. In particular, programming features that support cobots for collaborative scenarios will be reviewed. The work scopes and contributions of this paper are:

- an overview of collaborative industrial scenarios
- a general structure for a cobot programming, including safety measures and on-line/off-line human involvement
- a review of three main programming features for cobots; communication, optimisation and learning

This paper is organised as follows. Section 2 presents an overview of HRC scenarios, applied safety measures and general program structure. Sections 3–5 elaborates on cobot programming features used for collaborative industrial scenarios. Section 6 concludes the work by providing recommendation regarding the deployment of cobots in industrial settings and providing general guidance for the advancement of HRC-related research towards expanding and improving HRC industrial implementations.

2. Overview on collaborative scenarios and cobot programming

A human operator and a cobot can collaborate on a variety of industrial tasks, which are defined here as collaboration scenarios. In such a scenario, the human operator and the cobot share the same workspace to perform manufacturing processes on work pieces, such as pick-and-place, assembly, screwing or inspection. That is, each scenario involves a cobot, a human operator, work piece(s) and manufacturing process(es). Collaboration scenarios, safety measures, and cobot programming to support the scenarios are summarised below.

2.1. Collaborative scenarios

Definitions have varied over what constitutes human–robot "collaboration", versus "interaction" and "cooperation". For example, Haddadin and Croft defined their categorisation on physical proximity between a human and a robot and deem a cooperative robot works at a closer proximity to a human than a collaborative robot does [15]. Sylla and Mehta defined their categorisation on the type of allowed contact between a human and a robot. They state that a human can contact a cooperative robot if it is static, whereas the human can contact the collaborative robot even if it is moving [16]. In this paper, we adopt the most lenient opinion in [17], which states that any robot operating without a fence alongside a human is a collaborative robot. This definition is in line with the market definition of a cobot. This also achieves the widest scope in this review.

2.1.1. Categories of collaborative scenarios

This paper builds up on the categorisation in [17], which divides tasks according to the relation between a cobot, an operator, work piece(s) and the process(es) being performed on the work piece(s). That is, the categorisation is defined according to the degree of task intersection and dependency between the operator and the cobot (Fig. 1). This categorisation draws a clear line between the various required capabilities of a cobot in different industrial scenarios. This is helpful to programmers and researchers trying to define a direction for their work/research. The categories are:

- Independent: An operator and a cobot operate on separate workpieces (W_1 and W_2 illustrated in Fig. 1) independently for their individual manufacturing processes (P_1 for W_1 and P_2 for W_2). The collaborative element is due to the co-presence of the operator and cobot in the same workspace without a fence or guard. That is, safety is achieved through the cobot's intrinsic safety and/or added hardware/software safety elements. Therefore, the cobot is aware of the operator's presence and acts safely.
- Simultaneous: An operator and a cobot operate on separate processes (P_1 and P_2 respectively) on the same work piece (W) at the same time. There is no time or task dependency between them. However, the cobot needs to be spatially aware of the operator and his/her task requirements in order to respect the operator's space. Being able to concurrently operate on the work piece will minimise the transmit time of the work piece between the cobot and human, thereby improving productivity and space utilisation.
- Sequential: An operator and a cobot perform sequential manufacturing processes (P_1 and P_2) on the same work piece. There are time dependencies between the cobot and operator for their processes. For instance, the cobot works on P_1 for the work piece as an input to support the operator to carry on P_2 for the work piece. In most cases, the cobot is arranged to handle tedious processes to improve the operator's working conditions.
- Supportive: An operator and a cobot work towards the same process (P) on the same work piece (W) interactively. There is dependency between the actions of the cobot and the operator. That is, without one, another cannot perform the task. The cobot needs to understand the operator's intent and the task requirements in order to provide appropriate assistance. For instance, the operator fastens screws on a toolbox while the cobot holds it in place [18]. The role of the cobot is to physically assist the operator with work pieces which improves ergonomics.

2.1.2. Examples from industry

Manufacturing companies are eager to deploy cobots due to their affordability, built-in safety and intuitive User Interfaces (UIs). That is especially true for SMEs that have difficulties automating their manufacturing using traditional robots [19]. Mass production companies, particularly the automotive manufacturers, are equally eager to implement HRC to boost their competitiveness and take their factories to the next level of automation and manufacturing advancement, i.e. Industry 4.0. For instance, the BMW Group's Spartanburg site introduced cobots to improve ergonomics by taking over the repetitive and precise task of equipping the inside of car doors with sound and moisture insulation [20]. Audi introduced a UR3 cobot to apply an adhesive on a car roof, which saves factory floor space since the cobot does not have to be separated from the human by a fence [21]. The Volkswagen plant in locations that are inconvenient to reach by a human operator [22]. The cobot works alongside the human,

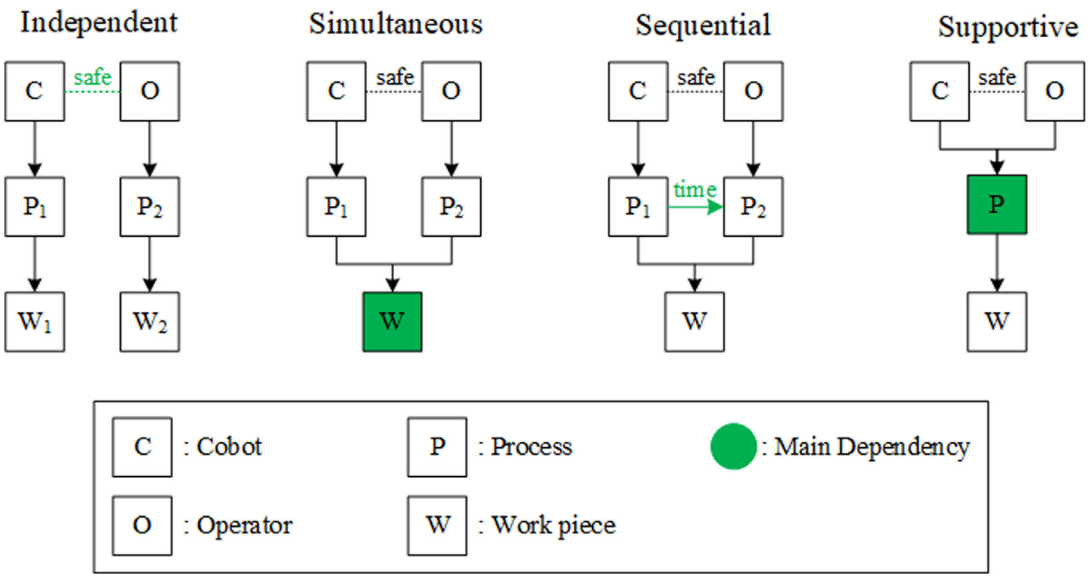

Fig. 1. Degrees of collaboration in industrial scenarios.

who is screwing at other easily accessible locations. Nissan's large-scale Yokohama plant deployed UR10 cobots to loosen bolts and carry heavy components to relieve the workforce of these arduous tasks and speed up the manufacturing process [23]. Skoda also introduced a KUKA cobot to work alongside humans in the production of direct-shift-gearboxes [24]. In all the aforementioned tasks, cobots are required to operate safely alongside humans, which is a built-in feature in cobots. However, they do not strictly require advanced perception, human-awareness or decision-making capabilities since the parts handled are kept in pre-determined positions, the tasks of the human and the cobot are relatively independent, and the cobot adheres to a relatively fixed action/motion plan. Therefore, it is noticeable that industrial implementation of HRC scenarios fall under the category "Independent" or "Simultaneous". However, by enforcing additional constraints upon the cobots' environment (in terms of fixed parts or equipment positions), most of such implementations fail to showcase the utility and versatility of cobots in a partially unstructured work environment.

2.1.3. Examples from research

As research strives to explore new potential use cases for HRC and cobots in industry, a wider range of HRC scenarios can be found in academia. Table 1 highlights a few examples of references that have included interesting set-ups of HRC scenarios in their experiments.

Most HRC-related research works fall under the categories of "Sequential" or "Supportive", generally focusing on involving a human in a cobot's task plan more than industrial examples have permitted so far. The research topics include understanding human intent and social cues, natural human–robot communication, optimising behaviour for human comfort and trust and learning tasks from humans, all of which enable cobot *flexible* behaviour.

To enable collaboration, equipping a cobot with cognitive abilities is essential. In traditional non-collaborative scenarios, programming a robot to follow fixed paths and sequences of actions sufficed to perform tasks successfully. However, with humans in close proximity, such as in an Independent or Simultaneous scenario, special safety measures should be taken. This can be achieved using the embedded safety features in cobots, such as limiting velocity and detecting collisions, or using distance sensors that allow a cobot to slow down or stop when something/one is in close proximity, e.g. [36]. More sophisticated research works achieved safety by detecting objects/people and planning real-time optimised collision-free paths, e.g. [37]. As the degree of collaboration increases, such as in Sequential or Supportive tasks, a cobot needs to have semantic understanding of the task goal and the human's actions and intent. Moreover, the human needs to be able to communicate with the cobot through intuitive ways such as body language, speech and intuitive UIs. This communication should be reliably understood by the cobot to act upon. Moreover, the human needs to be able to teach the cobot new tasks intuitively. If the task is complicated beyond what the human can coordinate, the cobot should be able to autonomously select and execute an optimal action, while accounting for the human's choices, actions and task goal. Hence, more advanced programming techniques, such as optimisation and learning, are imperative to address the above abilities that enable sequential and supportive HRC scenarios and improve independent and simultaneous HRC scenarios.

2.2. Safety measures for HRC

Sylla and Mehta summarised four scenarios of HRC [16], adapted from ISO 10218-1 [38], for collaborative operation requirements. The categorisation for collaboration is defined mainly from the perspectives of the safety strategy and the spatial relation between a cobot and a human operator. That is,

- Safety Monitored Stop: A cobot operates normally on a work piece in a well-defined workspace. If an operator enters the workspace, the cobot completely stops so that the operator can perform operations on the work piece.
- Hand Guiding: A cobot is compliant and moved manually by a human. This allows intuitive easier path teaching. It makes complex and interactive collaboration possible.
- Speed and Separation Monitoring: A cobot's workspace is divided into zones. The closer a human operator gets, the slower the cobot moves. The cobot reaches a complete halt at a certain threshold. This will enhance the safety between the operator and cobot.
- Power and Force Limiting: A cobot is programmed to operate only within tolerable levels of force and torque. An operator can be as spatially close to the cobot as needed without relying on external safety sensors.

Table 1
Examples of HRC scenarios from research.

Scenario	E.g.	Human Task	Cobot Task
Co-manipulation	[25]	The human and the cobot both hold and move an object. The human guides the object's path.	The cobot handles the object's weight.
Fixture	[26]	The human polishes the held box.	The cobot hold the box in a position according to learnt human preference.
Handover	[27]	The human takes objects from the cobot and places them aside.	The cobot hands objects to the human. Handover pace changes according to the human's readiness to take an object.
Assembly	[28]	Assembly actions are distributed between the human and the cobot according to expected workload and energy consumed.	
Pick-and-place	[29]	The human chooses objects randomly to pick and place.	The cobot chooses objects to pick and place while accounting for distance, reachability and the human's predicted motion plans.
Fetch	[30]	The human takes the table part from the cobot and performs assembly actions.	The cobot fetches table part according to the human's progress in the assembly task.
Soldering	[31]	The human adjusts the pose at which the cobot is holding the solder wire, and then he performs soldering.	The cobot holds the soldering wire at the tip of soldering point, in human-controlled orientation and position.
Illumination	[32]	The human operates on parts on a table.	The cobot, with a light source mounted as a tool, provides direct illumination on the human's work space while avoiding collision.
Inspection	[33]	The human screws bolts in holes.	The cobot inspects if all holes are screwed and issues a warning in case of missing bolt.
Drilling	[34]	The human specifies the drill location, during run-time.	The cobot drills a hole in specified location while having motion automatically constrained to drill bit's axis.
Surface Finish	[34]	The human specified surface to sand.	The cobot sands the surface, while having motion automatically constrained parallel to the surface.
Screwing	[35]	The human inserts bolts in holes on one side of a plate.	The cobot tightens the bolts from the other side of the plate.

Most cobots come with built-in features that ensure the above safety requirements [5]. For example, the MRK SYSTEM-KR 5 SI and the COMAU are equipped with tactile sensors to detect contact. The BOSCH APAS has smart capacitive skin that detects the proximity of a human and stops before contact. In general, most cobots are built to comply with ISO 10218-1 [38] for the safety requirements of robotic devices and ISO/TS 15066 [6], which is more specific for cobots.

Despite that, cobots still need to undergo risk assessment before being implemented on a factory floor, such as in [39]. That is because dangers might arise from the nature of the task rather than from the cobot itself. For example, even if a cobot moves with a safe speed and force, and stops upon collision, the cobot is dangerous if it is holding sharp tools or parts. Moreover, the cobot cannot discern whether it is colliding with the human's arm, which is uncomfortable but permissible, or with his/her head, which is unacceptable regardless of speed and force. Therefore, a cobot should be enhanced with additional intelligence and perception abilities to be fully safe. A lot of work has been done on collision avoidance [32,37,40,41], human motion prediction [42,43], risk assessment through simulation and VR [44] and other safety enabling technologies. However, what might be holding back the implementation of collaborative systems, in terms of safety, is that many of the safety enabling systems have not been officially certified. A tighter collaboration between the industry, academia and standardisation bodies is needed to approve and launch safety-related research work. Moreover, an added burden on industrial parties is the cost of training staff on new certified safety systems and risk assessment techniques. Therefore, researchers should also consider their developed safety systems from a user-perspective.

2.3. Cobot programming

The programming process entails providing a cobot with the ability to understand the state of the environment and perform actions that advance the system towards a planned collaborative goal. Traditionally, a human, the programmer, is only involved off-line for an industrial robot program. These programs are inflexible and not human-aware, and cannot be altered during runtime, unless an error occurs and debugging is needed. Based on that, a robot functions in a deterministic environment in which an operator is not part of. However, in HRC, an operator adds stochasticity and unpredictability to the environment. The human involvement in the cobot's program goes beyond the programmer's traditional off-line role. The operator also becomes involved in the cobot's program during run-time, or on-line.

An operator can be involved in modifying or affecting a cobot's program either explicitly or implicitly. Explicit involvement occurs in the form of direct communication, i.e., the human sends information or instructions to the cobot. Implicit involvement occurs such that the cobot observes the human's states and alters its policy accordingly. The policy can be learnt from prior data or modelled manually by programmers. Based on these different modes of operator involvement, this paper identifies three different programming features that give the cobot the ability to act flexibly and/or be programmed intuitively. These programming features are especially essential for Sequential and Supportive HRC scenarios. The programming features identified are:

- Communication: An operator controls a cobot through a communication channel that can be verbal (speech) or non-verbal. Non-verbal communication includes gestures, gaze, head pose, haptics and UIs. The off-line role of the programmer is to program and define possible cobot actions and the underlying motion control. The on-line involvement

of the operator is mostly explicit, triggering the cobot into pre-defined actions.

- Optimisation: Important aspects of a cobot's surroundings, such as obstacles and tool positions, are mathematically modelled as a function of the cobot's actions. Those form cost functions that are optimised to generate desirable performance. The cobot's program can be made to minimise an operator's workload, energy consumed and time wasted, or maximise physical comfort and trust, product quality, etc. During off-line development, the programmer designs cost functions and optimisation algorithms. During runtime, the operator usually impacts the cobot's performance implicitly, since he/she will be a part of a cost function. The advantage of this method is the higher likelihood of performing more optimally than a human operator.
- Learning: A cobot learns a skill similar to how a human would, e.g., through observing demonstrations, trial and error, receiving feedback and asking questions. The off-line role of a programmer is to design the learning algorithm and provide initial data for the cobot to learn from. That could be in the form of demonstrations, trial-and-error iterations (resulting in a policy), training data, etc. During runtime, an operator might be able to explicitly affect the cobot's policy by providing additional data, such as feedback, answers to questions, personalised demonstrations, etc. Moreover, the operator might serve as a prior in the cobot's probabilistic learning algorithm, i.e., affecting the cobot implicitly by being part of the observed environment.

Table 2 details some examples of cobot programs found in literature, highlighting the above three different features above. Sections 3–5 elaborate on these programming features, their variations and implementation in HRC scenarios.

3. Communication

Humans rely heavily on communication to work in teams and complete tasks fluently and efficiently. Communication can be made to issue orders, convey intention and ask/answer questions. Researchers have been working on enabling communication between humans and cobots such that the human is able to command the cobot through different communication modes. The works mentioned in this subsection are categorised by communication mode: body language and speech, user interfaces and haptics.

3.1. Body language and speech

Body language as a means of commanding a cobot includes using gestures, pointing, head pose, and gaze. Speech refers to uttering commands verbally. These two communication modes are combined in the same subsection since they share a similar algorithmic pipeline: First, a communication guideline must be defined, i.e. in what ways of language/words/gestures, will the operator communicate with the cobot? Communication signals are detected, recognised and mapped to executable actions for a cobot. The rest of this subsection tackles the different research works done towards these different steps in the 'body language and speech' communication pipeline.

3.1.1. Communication guideline

Defining an effective communication guideline involves specifying useable communication signals, such as a set of gestures or phrases, and when and why to use them. Various approaches have been found in the literature to define a communication guideline. To begin with, a set of useable gestures or phrases can be predefined strictly in a fixed set. A gesture lexicon can be extracted from observing human–human interactions [47–49]. However, gestures extracted from observing human–human interactions will not necessarily be easy to recognise and differentiate. That is because many of them tend to be very subtle, context-specific and sometimes person-specific.

In other cases, a set of gesturing rules is used to create a gesture set. Berattini et al. proposed a standard set of gestures for a given task (the gestures must be distinct from other task actions), and evaluated a gesture recognition algorithm on the proposed set [50]. However, even with optimally chosen gestures, having to memorise and adhere to a fixed set of signals can be mentally draining and unintuitive for the operator (see Fig. 2).

Allowing a human to communicate with a cobot in his/her own way results in more effective, natural and intuitive communication. Cheng et al. designed a framework to extract robotic operations from natural language based on relationships between mentioned work pieces and representing the relationships in matrix form [51]. She and Chai used interactive learning to learn verb semantics in a noisy incomplete environment [52]. The cobot is capable of asking the right questions to learn required actions and corresponding objects, states and tools. Maurtua et al. also analysed natural language in light of task ontology to extract commands [53].

Generating a natural language system is challenging since the language use differs drastically as the operator progresses with work. Nakata et al. showed that in a collaborative task where only verbal communication is allowed, the frequency of morphemes (i.e. words belonging to these certain types: object, modifier, robot action, user action) decreases as the number of task trials increase [54]. That is because humans naturally start emitting words as they become accustomed to the task. They naturally start considering and accommodating their team-mates' needs without those needs being explicitly expressed. Kobayashi et al. also showed that the use of descriptive words decreases as the number of task trials increase [55]. Therefore, any language model between a human and a cobot should account for the change in human language as the human becomes more accustomed to the task.

3.1.2. Multi-modal communication

Different research works have shown that communication modes can be concatenated in different ways for better context understanding. Using multi-modal communication can outperform single mode communication. Multi-modal communication can sometimes be complementary such as a point-and-command system. It can also be redundant such as a same-command speech and gesture system [53]. The challenge lies in how to combine information from different communication modes to successfully draw conclusions. Srimal et al. used fuzzy logic to combine pointing gestures with speech in order to identify pointing targets or execute spatial commands [56]. Giuliani and Knoll used a score-based system to identify which action to perform on which object [57]. They represent an object, its corresponding action, and a score R as one tuple. When a speech or pointing command is uttered mentioning an object or an action, the scores of the tuples including the object or the action are increased. When the score exceeds a threshold, the action is executed on the object.

Maurtua et al. used a fusion engine to ensure voice and pointing commands are not contradictory and combine them into a single command output to the execution manager [53]. They designed a gesture, including pointing, and voice command system with safety functions integrated and implemented it on a KUKA IIWA. Their system was rated promisingly in terms of naturalness, usefulness and reliability in an extensive study.

It is important to consider if/when multi-modal communication is needed, before considering how to implement it. Admoni

Table 2
Examples of HRC programs with different programming features, showing the distribution of roles between the programmer and the operator.

Feature	Off-line programmer role	On-line operator role
Communication e.g. [45]	Design speech recognition algorithm and define task plan	Use verbal commands to trigger actions from the task plan (explicit involvement)
Optimisation e.g. [29]	Design cost function and optimisation algorithm to make the cobot choose optimal object to grasp	Pick up objects which affects the cobot's cost function and hence, changes which object the cobot picks up (implicit involvement)
Learning e.g. [18]	Design an interactive learning algorithm and provide initial policy	Provide feedback which alters the cobot's assembly action sequence (explicit involvement)
Learning e.g. [46]	Design LfD algorithm and provide demonstrations	Act similar to demonstrations. The cobot observes the human and provides complementary movements according to demonstrations (implicit involvement)

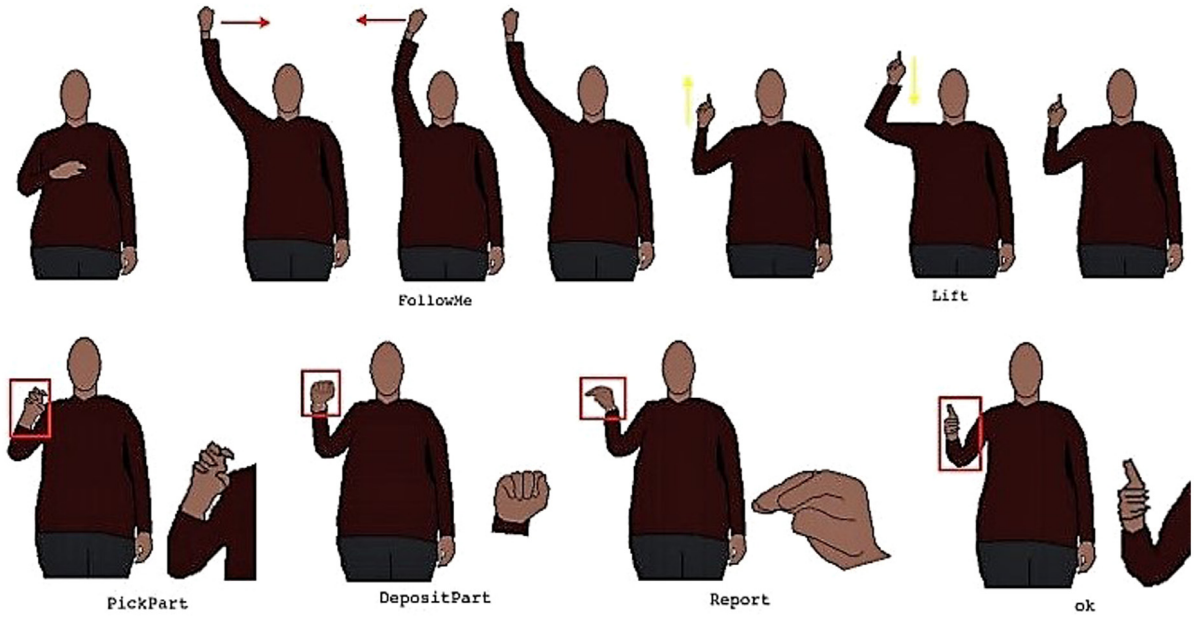

Fig. 2. An example of a gesture lexicon adapted from [50].

et al. worked on determining when a pointing gesture is necessary along with a verbal description to identify an object on a table [58]. A gesture only be necessary, for example, if there were several objects of the same verbal description in close proximity. Their work can be used to guide and prompt communication to only when it is needed, which would improve efficiency and lessen chances of error and confusion.

3.1.3. From communicated signals to executable actions

After specifying a communication guideline, the permissible communicated signals must be mapped to executable cobot actions, i.e. a signal should be made a command. This can be done manually or through learning:

- Manually: A programmer manually assigns gestures to cobot actions off-line according to task needs [50]. Human–human interactions can help programmers understand which gestures map best to which actions [47,48]. If the action domain is continuous such as in [31], then the gesture-action mapping is done through calibration.
- Learning: Interactive learning can be used so that an operator plays a role in the signal-action mapping. Shukla et al. taught a cobot required actions to perform given a gesture using incremental human feedback [59]. However, this can present unnecessary complications in an industrial environment where insufficient variability in the mapping is expected. In a continuous action domain, such as in [60], the gaze-object associations are obtained by a pre-trained Support Vector Machine (SVM) in order to ultimately predict the human's intent (i.e. the object the human is looking at). This helps the cobot start acting towards the object before an explicit command is uttered. This is particularly useful when the communication channel domain is continuous, such as gaze direction, and requires segmentation before mapping.

3.1.4. Signal recognition

Delving into the technicalities of signal recognition, whether it is gestures, speech, haptics, etc. is beyond the scope of this paper and will only be discussed briefly, as numerous relevant reviews already exist. Readers are referred to [12] for an extensive review on gesture recognition technologies in light of industrial HRC. Similarly, Benzeghiba et al. provided a review on speech recognition technologies [61].

To recognise gestures, the human skeletal frame, or pose, must be detected. Table 3 shows the advantages and disadvantages of different sensing technologies for pose detection. Given recent advancements in deep learning pose detection algorithms, such as OpenPose [62], the prospects of 2D cameras being the most suitable option is increasing.

Modelling the gesture depends on whether it is static or dynamic. Static gestures are modelled as single poses. For a pose to be detected as a static gesture, its temporal length needs to exceed a specific threshold [64]. Then, the angles of the different segments of the human skeleton are thresholded to classify the gesture. In the case of dynamic gestures, the human pose

Table 3
Advantages and disadvantages of different pose detection technologies.

Sensor	Advantages	Disadvantages
3D-cameras, e.g. Kinect v2 [33,63] and ASUS Xtion PRO Live [64]	Non-intrusive, easy to setup	Restricted detection region, prone to occlusion, dependent on lighting conditions, detection algorithm dependent on 3D sensing output (point cloud, depth map...)
RGB cameras	Non-intrusive, easy and affordable to setup, availability of reliable algorithms	Restricted detection region, prone to occlusion, dependent on lighting conditions
IMU Jackets, e.g. [65]	No occlusion, no dependency on lighting and environmental factors	Restrict mobility, not one-size-fits-all, does not measure hand gestures
Wrist bands, e.g. [66]	No occlusion, no dependency on lighting and environmental factors	Restrict mobility, difficult time-consuming setup
Motion Capture, e.g. [46]	High accuracy, computationally effective, less prone to occlusion	Expensive and time-consuming setup, restricted detection region

sequence, sampled at a certain rate, is modelled as a time series (such as Hidden Markov Models, Recurrent Neural Networks…) which is then used for detection.

Pointing gestures are different since they are related to the space and objects. Recognising or classifying them constitutes identifying the object or location being pointed at. In [53], Euclidean cluster extraction was used to detect the human's forearm. The forearm is modelled as a cylinder, and the pointing target is identified as the cylinder's axis intersection with the workspace. Srimal et al. also used skeletal tracking obtained from a 3D depth sensor to detect the direction of pointing in a similar manner [56].

For speech recognition, some have used Google API [53] or the Microsoft Speech API [45,60] for speech to text transformation. In the case of natural language, morphological analysis is performed to identify word morphemes and understand context. For example, Maurtua et al. [53] used FreeLing for morphosyntactic analysis while Nakata et al. [67] used the MOR and the POST program of CLAN. Nevertheless, Gustavsson et al. pointed out that relying on speech commands can be very problematic in the presence of background noise and chatter [45].

The intuitiveness of body language and speech communication is often traded with reliability. Therefore, exploring less human-like but more reliable communication modes, such as haptics and Graphical User Interfaces (GUIs), might be more suitable for industrial scenarios.

3.2. User interfaces

Since cobots work closely with operators, cobots need to be equipped with intuitive UIs. These interfaces are used by operators to alter/create/customise cobot programs, whether off-line or on-line. The previous works related to cobot UIs and research challenges are discussed below.

3.2.1. User interface mediums

A UI is an essential differentiator of a cobot from traditional robots. Besides the user-interfaces being developed in research communities, several industrial solutions are already available. UIs are categorised such as:

- Cobot teaching pendant: Market cobots, such as Universal Robots (UR), ABB's YuMi and KUKA LBR iiwa, are labelled as intuitive and user-friendly due to their modular symbolic programming UIs. For example, the UR UI allows a user to specify way points and create arrayed motion patterns. The YuMi teaching interface is similar, with commands for both arms easily synchronised and parametrised. Teaching pendants are the easiest to utilise since they are built-in with the cobot. However, at a surface level, their capabilities are limited and do not enable human-awareness and action plan flexibility.
- Icon-based programming: A visual library of built-in functionalities can be utilised to create the program. For example, in MORPHA, the icons are connected to form a series of cobot commands. In LabVIEW, a data flow diagram is created in which values flow across the icons and trigger actions on hardware. Although an icon-based program is easy to build (in small-scale programs), it is difficult to debug, maintain and alter. Therefore, they have not been popular in the manufacturing industry [68]. Moreover, these methods have not yet provided options to easily integrate the operator within the cobot's program so that the cobot is human-aware.
- CAD-based programming: Robot manufacturers and third parties have provided solutions such as V-REP, Visual Components and ABB's RobotStudio, in which performance can be validated and assessed. V-REP and Visual Components come with integrated human models that can also be programmed to help in validating the cobot's safety and collaborative functions around humans. In V-REP, a human can move according to a real-life actor through augmenting 3D sensor data, such as from Kinect v2, of a real-life human into the simulation. In Visual Components, the human can experience 3D simulation using VR which can be useful for training operators. However, since simulation has to match the real environment in order to achieve valid results, using CAD-based tools might be time consuming when changing work space design and reiterating the program, unless an automated method is devised to scan, map and build the environment.
- Task-based programming: This is the most popular research direction in intuitive cobot programming and will be further discussed in this subsection. The developed approach is based on a primitive-skill-task hierarchy [69]: Primitives are cobot motion commands or sensory inputs values, such as open gripper and sense torque. Skills are object-oriented and achieve goals such as pick object and tighten screw. Tasks are a sequence of skills and achieve the over-all goal, which is the industrial scenario being implemented.

3.2.2. Skill architectures

Skills are the building blocks of task-based programming. They present a balance between specificity and abstraction, i.e. skills are general enough to be building blocks of a wide range of tasks while maintaining a level of abstraction understandable to humans. The skill structure, in Fig. 3, was designed by Schou et al. [70]. A skill transfers the environment from a state to another. Skills have preconditions that need to be checked before implementation. They also have post conditions that are checked to make sure that the skill is correctly implemented. Moreover, skills need to be parametrised depending on their input states, and continuous evaluation takes place during execution to ensure safety and right progress. Steinmetz et al. identified four key considerations for efficient skill parametrisation [71]:

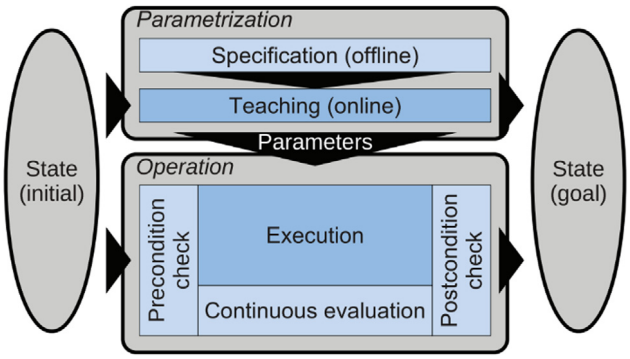

Fig. 3. "Skill" architecture derived from [70].

- It is better to teach a parameter when needed so that an operator can assess the environment and choose the parameter accordingly.
- If the cobot fails to perform a task after parametrisation, it should solicit the operator to edit the parameter.
- To avoid excessive parametrisation, static knowledge about relationships between parameters can be utilised to reliably derive some parameters from others.
- Instead of being specified by shop-floor operators, some parameters should be set at defaults.

3.2.3. UI capabilities

For more sophisticated behaviour, researchers have worked on incorporating smarter motion generalisation, information display and cognitive abilities (including perception) in GUI architectures.

- Guerin et al. designed a GUI that allows the user to specify cobot capabilities and constraints [34]. Constraints include tool linear or planar path constraints. The user is also able to record tool affordances which include movement primitives (recorded paths). For example, using their GUI, the user is able to record a drilling action (straight constrained motion along the drill axis) and reproduce the action in novel drill locations.
- Pedersen et al. represented a small set of skills needed in industrial cobots in a GUI [64]. A set of high-level skills (e.g. pick-and-place) can be parametrised by a single input (object or location) through pointing gestures. The skill set supported is limited, but they elaborated on their work in Pedersen et al. [72].
- Steinmetz et al. improved on the skill architecture and parametrisation to support more complex skills, such as screwing [71]. Their work was formalised as a UI called RAZER (Fig. 4) and evaluated in [73]. RAZER allows an expert user to intuitively design new cobot skills and parametrise them. It presents these skills and parameter options to shop-floor operators for easy task-programming.
- Schou et al. designed an interface that allows users to sequence skills and specify some pre-defined parameters [69]. Locational parameters, e.g. having to do with the location of pick up, are specified using kinaesthetic teaching.
- Koch et al. incorporated the skill architecture in a software system named Skill Based System (SBS) that enables the creation of skills for complex tasks such as screwing and assembly [74].
- Paxton et al. designed the GUI for cobot programming, based on Robot Operating System (ROS), which is symbolic, modular and expandable [75]. Objects and agents (humans and cobots) are represented in a natural abstraction the human understands. These abstractions are used to generate Behaviour tree-based task planners using pre-defined actions. The operator can also specify way points for the cobot's path.

The aforementioned UIs provide intuitive solutions for programming a cobot for industrial tasks by workers with minimal programming experience. A flexible cobot behaviour obtained by the UI is a result of its use on-the-fly according to the operator's plans. In some cases in task-based programming, an expert designs the skill sequence and leaves some of the parametrisation to be done by the operator on-line. This parametrisation is done by inputting values on the UI, by kinaesthetic teaching or by pointing gestures. The last two are only available for specifying locational parameters. A UI is an essential part of programming a cobot system whether by an expert or an operator, unlike the other technologies discussed in this paper that can be scenario/task-specific and optional.

3.3. Haptics and force

Commercial cobots come with a built-in "compliant" mode, i.e. the cobot moves according to the forces the human exerts on its body. It can be considered, thus, that the human commands the cobot explicitly through touch and force. Researchers have extended on the default "compliant" mode to increase the cobot's intelligence and user-friendliness. That is, beyond detecting a force, understanding and reacting to it, researchers have worked on predicting user intent and negotiating plans.

3.3.1. Reactive compliance: Understanding and reacting to the force

In reactive compliance, a cobot senses the forces exerted on its body and actively moves such that the forces are minimised. The challenge in reactive compliance is correctly mapping between the forces sensed by the cobot and the required motion to be done.

Most cobots are not equipped with tactile sensors on their bodies, which makes it hard to interpret exerted forces and understand the intent behind them. The cobot, for example, cannot identify the point at which contact with the human occurs and whether this contact is accidental or deliberate [76]. Magrini et al.'s work helps localise the forces being applied on the cobot [77]. That allows the cobot to respond to the contact force as desired or regulate it. Kouris et al. differentiated between collision and cooperation contact in a computationally efficient manner by thresholding the Fourier transform of the applied force/torque [78]. Gaz et al. differentiate between forces applied due to a polishing task and forces applied to move the cobot body by using a model-based approach [79]. This allows the human to smoothly and safely switch from moving the cobot compliantly and performing the polishing task.

Forces can also be difficult to interpret when they are exerted on the object a cobot is holding rather than directly on its body. The force that a human exerts on an object can signify an intent of motion in different directions. Wojtara et al. devised algorithms that differentiate between rotation and translation motion intent in a collaborative object-positioning scenario [80]. The first algorithm relies on degree-of-freedom (DOF) switching where the human explicitly specifies his/her intent (rotation or translation) and the cobot acts such that the right DOF are varied or fixed. In another algorithm, "Partner-that-follows", Wojtara et al. interpreted force as translation and torque as rotation intent above the human's axis [80]. The results section is used to assess the different algorithms proposed and compare them. All the algorithms present a reliable way of controlling a cobot for co-manipulation. However, there is no evolutionary element that

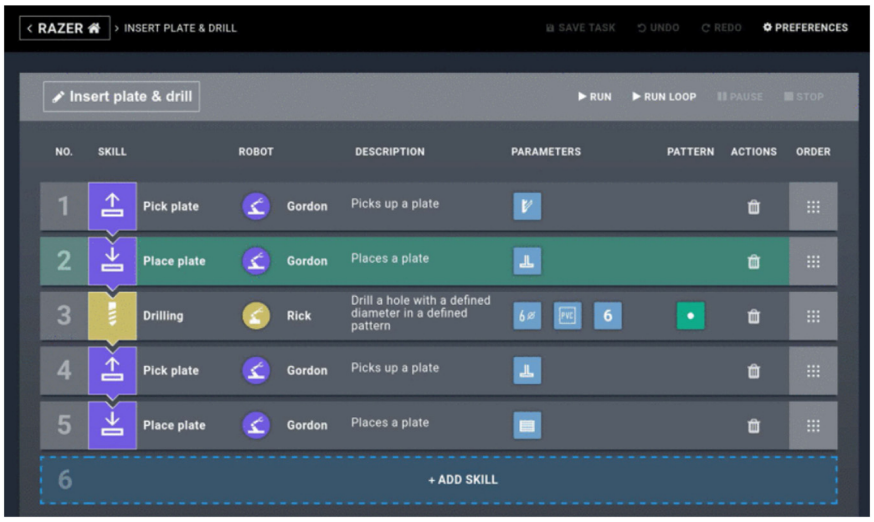

Fig. 4. The front panel of the RAZER interface from [73]. The figure shows an order of pre-defined parametrised skills for the task of drilling a plate.

allows the cobot to adjust with time to its human partner and the algorithm is very human-led. However, the work of Wojtara et al. is the closest to industrial feasibility due to its reliability and predictability [80].

As the number of potential directions of motion increase, it becomes difficult to manually toggle between them. Dumora et al. used learning algorithms to map from sensed forces to required direction of motion [81]. A Naive Bayes classifier is trained with the input vector of static forces on hand-held nob, and the output being the intent of direction of motion. The cobot then provides compliance in the intended direction.

Reactive compliance produces reliable results, which increases the level of trust the operator has in the cobot. However, since the cobot only moves under the influence of the operator, he still carries a mental and physical burden. To decrease this burden, the deeper understanding of the human intent and goal are needed in order to take a more proactive role.

3.3.2. Proactive compliance: Predicting the human's intent

Researchers have worked on deepening the cobot's understanding of the human's exerted forces to behave in a more proactive manner. The challenge in this degree of compliance is the accuracy of the inferences made from the exerted forces and the validity of their utilisation. For example, Li et al. used force to estimate desired target positions [82]. Estimating the human's desired target position decreases the amount of force he/she should exert as the cobot takes a more proactive role. This is achieved by integrating the predicted motion intent of the human into an impedance controller. The algorithm, however, assumes that the human's intended motion path is smooth and continuous. Therefore, a sharp change in intent results in higher needed torque and more time than a regular impedance controller. Lichiardopol et al. worked on decreasing the physical load on the human while assigning the cognitive responsibility to him/her [25], i.e. the human guides the path of the co-manipulation task with minimal exerted force. They assumed that the object's weight is unknown and potentially time-varying. Therefore, the algorithm estimates the force the human is applying based on the cobot's control torque and the position change. Then, the cobot amplifies its torque to decrease the estimated human exerted force. Moreover, the mentioned estimation and amplification steps happen in periodic cycles to cater for changing object weight.

Incorporating more intelligence and inference/prediction abilities in cobot programs decreases the physical and mental load on the operator. However, it also increases the probability of failure and unexpected cobot motions. Therefore, a more clear-cut between autonomy and reactive compliance would potentially avoid uncertainty and relieve the human of burden at the same time.

3.3.3. Mixed-initiative compliance: Trading-off between the cobot's and human's plans

A cobot has a goal path or position and acts autonomously to fulfil the goal. When an operator exerts force on the cobot, the system assesses how autonomous it should be as opposed to compliant. In some cases, the switch between the two modes is clear-cut, while in other cases the trade-off is smooth. The trade-off can be done by adjusting the stiffness values in impedance control or by weighting the autonomous and compliant components to achieve a combined result.

For example, consider the case where a cobot knows a predefined path while the human's intended path does not fully align with it. In such a case, the cobot must know when to favour its own path and when to switch to being compliant to the human, i.e., when to use its control torque input and when to use the human's applied control force. Li et al. solved this using a game theoretic approach in which the reliance on the two control inputs is weighted [83]. The weights are adjusted to minimise the difference between the applied human force and the "optimal human force" given the current motion direction. Briefly, when the applied human force matches the pseudo-force applied in the direction of motion, the cobot relies more on its own controllers to maintain the direction of motion. However, when the human force changes and is not aligned with the current direction of motion, the cobot becomes compliant and relies more on the forces to move rather than on its torques. This is similar to an impedance controller with autonomously varying damping and stiffness. The proposed algorithm creates smoother compliant motion while relieving the human from the continuous needed effort to push the cobot (Fig. 5).

However, in an industrial scenario, an operator will perform similar compliant motions for numerous times. The algorithm can also incorporate a learning element that compiles observed motion patterns and seeks to reproduce them while also being flexible to deviate from learned paths according to the human's current plans. An example scenario is co-moving a heavy object from Zone A and performing a precise positioning in Zone B. When approaching the object of an uncertain position, the human would naturally lead the cobot since he/she is equipped with better perception skills that allow more precise positioning.

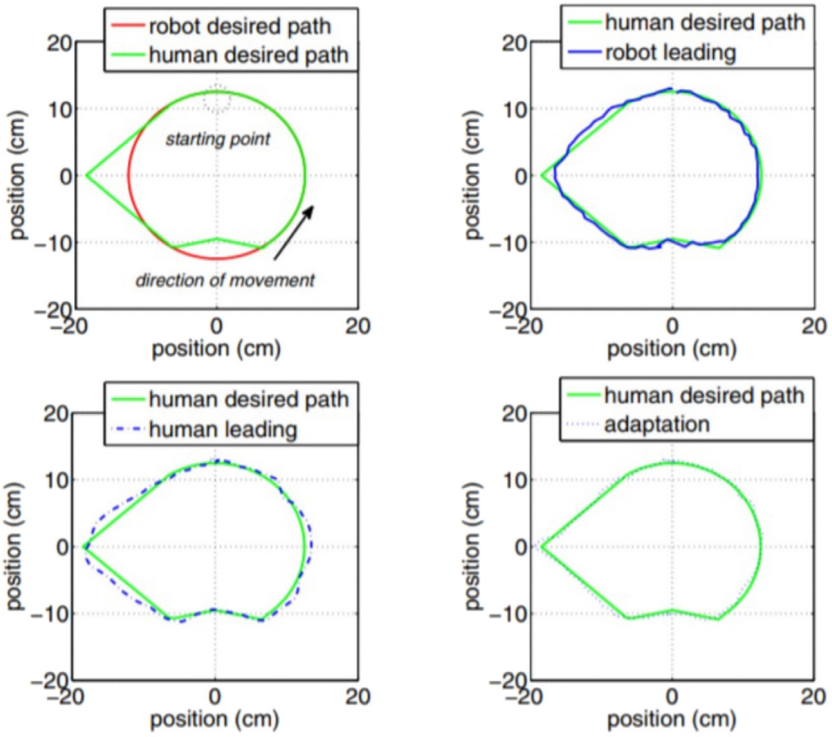

Fig. 5. Results from [83] showing that path is smoothest when a cobot switches modes between compliant and autonomous. The switching occurs according to the human's exerted force with respect to the expected path. When the force is in accordance with the path, the cobot tends to autonomy. When the force opposes the path, the cobot tends to compliance.

Similarly, the human would tend to take the lead when precisely positioning the object in Zone B. However, moving between zones can be done by the cobot after being led a few times by the human. Rozo et al. implemented a "learning from demonstration" algorithm that learns cobot stiffness from a set of kinaesthetic demonstrations [84]. The demonstrations are parametrised according to the position of objects, obstacles and the human and represented as a Gaussian Mixture Model (GMM). This algorithm proved more robust against unobserved positions and varying forces exerted by the human, as opposed to control algorithms with fixed stiffness.

Agravante et al. combined reactive and proactive behaviours by relying on both haptic and visual inputs [85]. The task handled is co-lifting a table while keeping a ball on it. The impedance controller which relies on haptic information, i.e. the forces sensed from the human, provides compliant behaviour in all 6 DOF. The vision controller only controls 2 DOF (z and φ_x) such that the ball is "attracted" to the centre of the table. In the case of intent conflict, the impedance parameters are adjusted such that the cobot becomes more compliant and less stiff. Sheng et al. also merged proactive and reactive behaviours [86]. In the problem, a cobot has to grasp a table side (gross motion) and then co-lift it with a human such that it remains horizontal (fine motion). The cobot learns how to approach and grasp the table using LfD. When the cobot successfully holds the table, is uses an RL-based reactive controller to keep the table horizontal with the human. A proactive controller predicts the human's position (equal to the cobot's required action) in future time steps using a Kalman filter and assuming constant acceleration. A behaviour gain controller then merges the suggested next-step action from the reactive and proactive controllers. The integrated algorithm combining the reactive and proactive performed better than just reactive algorithms. In conclusion, the trade-off degree of compliance provides a balance between minimising mental and physical load on the operator while also yielding predictable controllable cobot actions.

Aside from the challenges of designing the communication channel between a human and a cobot, Unhelkar et al. tackled the issues related to decision making in communication [87]. That includes the question of if and when to communicate, which relates to the cost and benefit of communication and the estimation of the human's mental state. They present open questions of how to quantise the cost of communication and its benefit to decide whether/how communication should be used. Since communication required explicit involvement from the operator, it can be mentally and physically tiring in repetitive long industrial tasks. Incorporating flexible autonomy through optimisation or learning, which will be discussed below, is an alternative.

4. Optimisation

Optimality is a primary goal during industrial design processes (product, process and production line design) since it ultimately yields a "maximum" profit. The main challenge in HRC scenarios is to optimise around the human, i.e. modelling and incorporating the human in the cost function. This subsection reviews the works done on optimising different aspects to yield optimal and semi-optimal cobot action in different industrial HRC scenarios.

4.1. Modelling different human states

Usually in repetitive non-collaborative industrial scenarios, processes are optimised with regard to minimising time, waste and maximising quality and profit. Obtained parameters from the optimisation process are incorporated in programs and control algorithms that dictate cobot actions. In HRC scenarios, however, the human is a central part of the cobot's surrounding, affecting its performance. Modelling the human is a challenge due to the

high number of factors and their unpredictable variability. Researchers have attempted to quantify or estimate human factors such as trust, physical load and mental state using observable and measurable states.

Since ergonomics is a main driver of implementing HRC, much has revolved around producing cobot behaviour that maximises humans' physical comfort and health. Modelled human factors related to the human's physical state include:

- Static ergonomic posture according to REBA: Busch et al. optimised cobot pose during handover to achieve human ergonomic posture [88]. They account for left/right handedness and avoid intimate body parts, all while keeping the human body in a safe comfortable posture according to the Rapid Entire Body Assessment (REBA).
- Muscle fatigue: Peternel et al. measured human muscle fatigue in order to adjust cobot's behaviours such that it handles more physical load and the human takes a more supervisory role [89]. The cobot does not take on all the physically-loaded tasks from the start since it needs to learn them from the human first. Hu et al. estimated dynamic human fatigue (varying with time as the human works more) per assembly action and accordingly distributes tasks between operator and cobot [90].
- Human joint torques: A key work in optimising co-manipulation for ergonomics is done by Peternel et al. [7], a follow-up work of [91]. In both, the human's pose is optimised during co-manipulation or handover task, such that the torque on the human joints are minimised. Table 4 highlights the main differences between the two works. An extension of these works would be to merge the benefits of both together by mathematically remodelling the problem (Fig. 6).

Optimising for ergonomics also includes accounting for the human's mental model/state, including:

- Human knowledge of task: A mental model includes the human's knowledge of a task which, if known, helps the cobot assist only where and when needed. Milliez et al. designed a task planner that enables a cobot to decide when to instruct the human through a task, when to do the task itself and when to monitor the human's performance [92]. Their planner accounts for the human's expertise which is based on successful task attempts. Such a planner is useful in industrial situations since the cobot can know how much interference in the task is required depending on operator experience. Devin and Alami designed a framework that estimates the human's mental states, i.e. the human's knowledge of the environment, plans, progress and goal, and triggers the cobot to only communicate with new information to the human when needed [93]. In tasks where several goals are possible, Zhu et al. estimated the goal belief of the human and optimised their action sequence such that the wrong goal of the highest probability is eliminated [94]. Several other works studied the preference of humans for proactive (perform sub-tasks autonomously) versus reactive (perform tasks when triggered or asked for help) cobots [95,96].
- Trust in cobot: The mental state also includes a human's emotional state, i.e. stress/trust level. Sadrfaridpour et al. controlled the cobot joint velocity while accounting for the estimated human trust level [97]. The trust level is estimated based on the progress of the human along his path while working alongside the cobot. For instance, a human is moving unusually slowly is being wary and careful and thus assigned a low trust value. The trust value is then fed into a non-linear model predictive controller (NMPC) to obtain control inputs. Compared to a controller that only aims at synchronising the human and cobot's motions, the trust-integrated NMPC resulted in a higher trust level and less perceived workload for the co-worker human. In scenarios where the human and the cobot co-lift a work piece, the human performs better with a cobot that moves along a path in a biological velocity pattern rather than a fixed velocity [98]. Huang et al. created an algorithm that slows down its motion to match the human's pace and task progress [27]. It shows that this is preferable as opposed to a cobot that executed its motion at a fixed pace and remains idle until the human catches up. Research was also done to produce legible cobot behaviour, which helps the human anticipate the cobot's intentions and increases the trust level [99,100].

4.2. Balancing between human and task benefits

However, as mentioned earlier, the goal of optimisation from the industry's standpoint is not merely to ensure better comfort for the human operator. Task parameters should be selected to minimise loss and time. Besides accommodating the human operator, the industry is interested in optimising towards task efficiency, i.e. improving product quality and decreasing production time (which can be estimated [101]). Faber et al. used CAD information to optimise assembly sequence to achieve low mental and physical load on the human, and minimise the number of cobot tool switches and human–cobot switches [102]. Johannsmeier and Haddadin distributed assembly sub-tasks between a human and a cobot as to minimise workload or energy consumption per subtask [28]. Hawkins et al. predicted human actions probabilistically to optimally enlist cobot help and minimise wait time [103]. This probabilistic prediction is based on observations of the human's previous actions and observation reliability and trust.

The advantage of using optimisation is that it yields optimal cobot behaviours. However, an optimal behaviour may sometimes conflict with the human's plans or preferences. Game theory enables cooperation between agents (humans and cobots) such that mutual benefit is realised, which is why it has been utilised to produce rational cobot behaviour accommodating human interest. It combines intelligence and rationality in persuading one's goal while considering the plans and benefit of other agents. Gabler et al. modelled a human–cobot close proximity pick-and-place problem as a two-player game and uses a Nash Equilibrium to solve a cost function that accounts for travel effort, object reachability, object preference as well as collision risk [29]. Li et al. used game theory to switch the cobot role from leading (assuming a predetermined path) to compliant (deflecting from the planned path) in a co-manipulation task, based on forces exerted by the human [83]. Gombolay et al. extended a dynamic scheduling algorithm, Tercio, to accommodate human preference, workload and situational awareness [104], applied in object fetching.

Banziger et al. used a simulation tool in order to optimise task allocation in several collaborative tasks [105]. The use of a simulation tool allows to calculate several human and task parameters such as ergonomics and production time. It also allows to measure these parameters in multiple task distributions, which enables finding the optimal task allocation.

The prevalent limitation with optimisation is that optimisation models are manually designed. Human states, although can be captured in a model, remain relatively simple. This gives rise to learning algorithms that allow the modelling of actions and human states automatically learnt from data.

Table 4
Comparison between [91] and [7].

Kim et al. [91]	Peternel et al. [7]
Only semi-static forces applied on the human were accounted for.	Forces obtained from dynamic motion were also accounted for.
The centre of pressure (CoP) was measured using a pressure sensor plate the human stands on.	The CoP was estimated using the weight of held object.
The forces need to be applied on the human before they can be minimised.	The forces are predicted and optimised before applying them on the human.
Can be used in co-manipulation scenarios.	Can only be used in handover scenarios in which the human carries the entire load.
Any force applied on the human by the object or the cobot can be accounted for.	Only vertical forces applied by weights of the object held are accounted for.

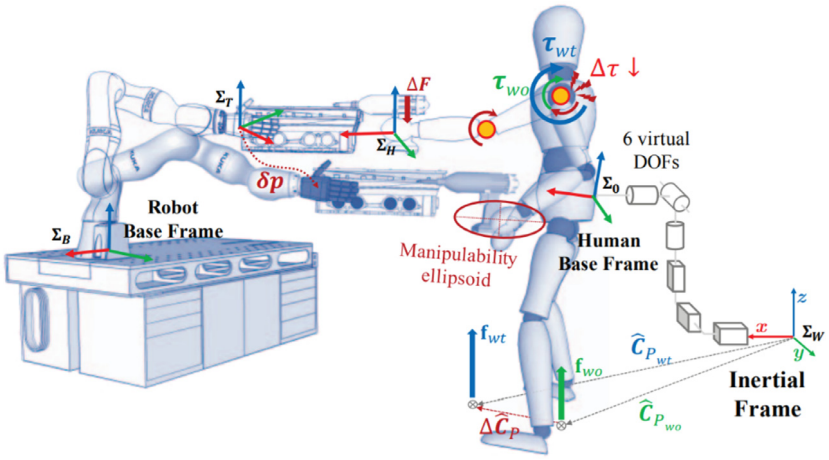

Fig. 6. The force modelling used to calculate the human's joint torques [7]. The model is used to optimise the cobot's position such that the joint torques are minimised. Only the held object's weight is accounted for in the model. Reaction forces from tools, such as a drill, are not accounted for.

5. Learning

Humans learn new tasks by observing them being done, by trying to do them and by asking questions and receiving feedback on performance. A human teacher serves to demonstrate a task, answer questions and provide feedback, all of which do not require programming skills. Researchers have attempted to enable a learner–teacher relationship between the cobot and the operator due to its naturalness and wide potential. In HRC, it is advisable to equip the cobot with learning capabilities since the operator might have to expand its skill set due to unforeseen working circumstances. Moreover, learning provides a balance between allowing the operator to make decisions first, then relieving him/her of the mental load as the cobot learns to operate autonomously.

5.1. Learning from demonstration

LfD is a very popular programming method in HRC due to its apparent intuitiveness and convenience. Research focus has revolved around capturing demonstrations reliably and easily, and encoding accurate state–action information to reproduce the task robustly in a new environment.

5.1.1. Recording demonstrations

Recording or displaying demonstrations can be done in several ways, each with advantages and disadvantages (Table 5). Aside from human demonstrations, kinaesthetic teaching is the fastest way of recording a demonstration [106] and is generally preferred by users [107]. Teleoperation is highly dependent on the device used and it would yield comparable results to kinaesthetic teaching depending on the design [106].

Other solutions might fall in a grey area between the aforementioned methods. When teaching by kinaesthetic demonstration, the human is often in an uncomfortable position moving the cobot and teaching the cobot's path causing unnecessary jerks. Also, only the trajectory knowledge is transferred and not stiffness information. Therefore, Yang et al. presented a hand bracket interface that allows a human to naturally and comfortably move a cobot [109]. Moreover, electromyography (EMG) signals are measured from the human's muscles which are transferred to the cobot as stiffness information for the impedance control and as open/close commands for the gripper state control. However, with their current hardware design (Fig. 7), the cobot must have two arms. Different methods can also be used concurrently in the same system, to learn different aspects of the task [110]. This depends on the demonstration encoding requirements and the pros and cons of the different methods.

5.1.2. Encoding information from demonstrations

LfD algorithms differ in what information they encode from the demonstrations, making it difficult to design one that caters for all expected variability in the environment and capture all requirements. Encodings can be motion-level or task-level. Motion-level encodings include:

- Motion with respect to obstacles: Ghalamzan and Ragaglia encoded obstacle presence from demonstrations, so that the reproduced cobot actions could avoid moving obstacles to reach a target location [111].
- Motion of two agents (humans or cobots) with respect to each other: Vogt et al. encoded correlation-based interaction meshes (Fig. 8) from one human–human demonstration where one human led whereas the other followed. Then, they reproduced cobot motion that matches the human follower's pose with respect to the human leader while avoiding the correspondence problem [46].

Table 5
Method of recording demonstrations and their advantages and disadvantages.

Method	Advantages	Disadvantages
Human demonstration, e.g. [108]: the human records him/herself doing the task. The cobot needs to extract object-goal relations or other useful information from the observed demo or relevant human joint paths to replicate.	Easiest for the human to perform	Not applicable to scenarios only done by the cobot (lifting heavy objects), detecting the human pose might be inaccurate, mapping the human pose to cobot pose is a challenge (correspondence problem)
Kinaesthetic teaching: the human holds the cobot and moves it as required by the task. The cobot is in compliant mode.	Straight-forward to perform and setup since compliant mode is a built-in feature for market cobots	Is difficult to perform with bulky or heavy cobots (e.g. UR10), may generate a shaky paths, not suitable for high precision tasks, not suitable when there are spatial constraints around the cobot
Teleoperation, e.g. [106]: the cobot is controlled remotely using an external device and the path generated is recorded as the task demonstration.	Might be intuitive and fun for some operators, can yield very smooth and precise paths depending on the device used, its sensitivity and calibration	Setting up and calibrating the device is a lengthy process prior recording the demonstrations, causes discomfort for some operators who find cobot motion "unpredictable"
User-Interface: the cobot is controlled using teaching pendent, whether by joint movement or end-effector movement.	Some movements are easier such as gripper rotation, predictive and consistent	Not instinctive, takes a long time, tedious, reaches a lot of singularities and needs resetting often

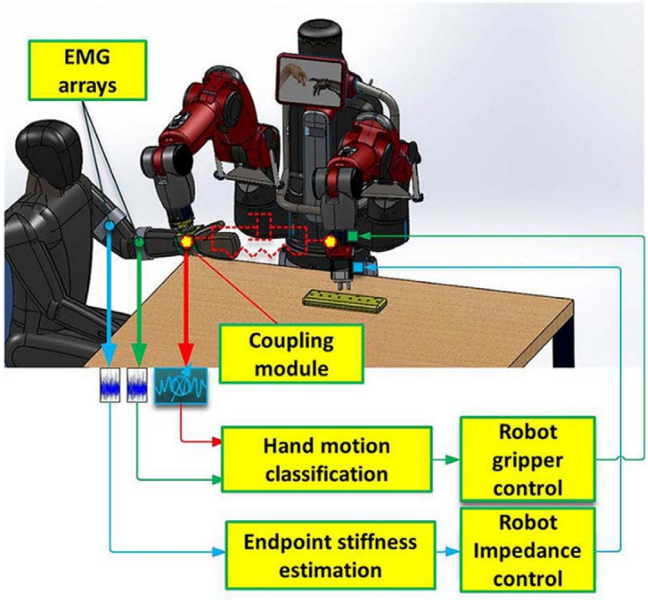

Fig. 7. Demonstration-capturing interface designed by [109].

- Constrained position and path with respect to objects and tools: Perez-D'Arpino and Shah encoded required postural (relative to work pieces) and path constraints for multi-step tasks [112].
- Compliance level (Stiffness) as a function of path: In a co-manipulation task, Rozo et al. encoded compliance level (stiffness) given force and position inputs and could therefore reproduce co-manipulation behaviour with the right stiffness [84].
- Path dependent on position of landmarks: In task-parametrised LfD [113], the path of the cobot is encoded with respect to multiple landmarks as opposed to one. The importance/relevance of these landmarks to the path is automatically calculated from the variance of the path between multiple demonstrations. Task parametrised LfD has been used to generate trajectories for assembly tasks [114].

Task-level encodings include:

- Encoding action preconditions and effects: Liang et al. encoded task-level information from kinaesthetic demonstrations [115]. The task preconditions and effects were extracted and used to create action models. During run-time, pre-conditions were identified by the cobot and the suitable action model chosen to create the desired effect.
- Encoding action sequence: Maeda et al. used demonstrations to encode different sequences of human–robot actions to accomplish the same task [30]. During runtime, a lookup table is used to identify the most likely sequence followed according to the human's observed actions to predict and provide the complimentary cobot actions. Also, Hamabe et al. also generated the task finite state machine (FSM) from a set of demonstrations which was used during runtime to identify the required supportive action [116].

Other algorithms encode both motion and task-level information. For example, Gu et al. created the Portable Assembly Demonstration (PAD) system that learns task-level and motion-level skills from human demonstrations and kinaesthetic teaching, respectively [110]. The system detects parts and tools and automatically recognises assembly states, actions and parts/tools involved, after observing the human demonstration. Kinaesthetic teaching was used to learn primitive actions that enabled these skills. Their system is robust to occlusions and environment changes, and is able to handle complex assembly tasks such as screwing, wrenching and hammering.

5.1.3. Time aligning demonstrations

Time alignment is important when demonstrations and execution are not guaranteed to run exactly the same rate. Time alignment, such as dynamic time warping (DTW) [46], is used to temporally match the demonstration with the sequence of states observed so far. However, problems might arise when the performance velocity differs drastically from demonstration to execution. Therefore, Maeda et al. rely on phase estimation instead which accommodates different velocities of human motion [117].

5.1.4. Expanding the demonstration set

Another challenge in LfD is how to generate enough demonstrations showing the right variability, as doing so is time consuming. Moreover, how should one make sure that demonstrations are being generated usefully, and are not being redundant? Forbes et al. relied on a seed demonstration and then solicited a crowd to edit the demonstration using a GUI for all the scenarios this demonstration would fail in [118]. Luo et al. used on-line learning to expand the library of demonstrations when required [119]. Arm reaching motions are encoded as a GMM library, and during run-time, the partial trajectory is identified in the GMM. A GMR is used to predict the rest of it, allowing the prediction of reaching target. When a new reaching motion that does not resemble the existing GMM is recognised, the

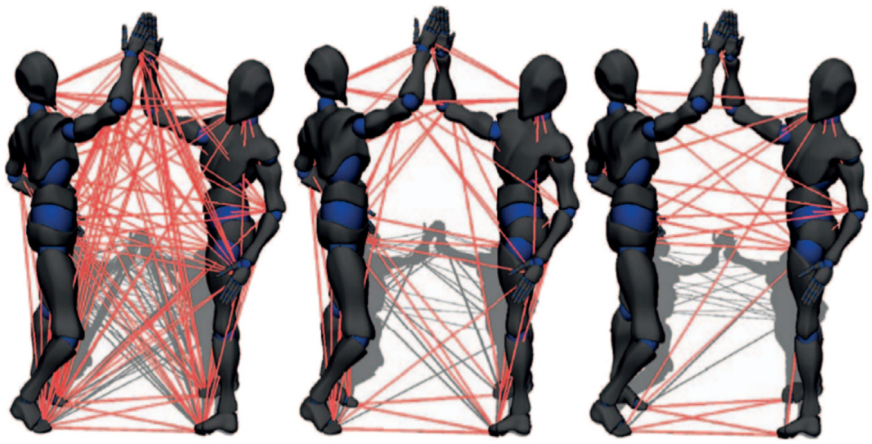

Fig. 8. Left: Fully connected interaction mesh. Middle: Interaction mesh created using Delaunay triangulation. Right: Correlation-based interaction mesh from [46]. *Source:* Figure adapted from [46]

GMM library grows. Mohan and Bhat presented a growing, multimodal memory framework that encodes diverse experiences of the cobot in HRC settings [120]. It recalls past experiences in present context to plan future action. Their framework contains a perception system that stores object information as well as an action system that stores motion plans. The two systems interact together and with the Episodic Memory system that encodes experiences, infers goals and plans.

LfD is a special case of supervised learning, in which a set of truths are given and learned from. Supervised learning can also be used to map between states and required actions. For example, in [81], a Naive Bayes classifier was trained to output the direction of cobot compliant motion when given a vector of static human-applied forces on end effector.

5.2. Reinforcement learning

RL has been used to teach intelligent robots skills such as walking [121], grasping objects of irregular shapes [122] and flipping a UAV [123]. Robots are left for extended periods of time training the RL policy. In some cases, [124], for grasping tasks, several robots are trained at the same time and knowledge is shared. In an industrial situation, such training time and resources might not be available. To combat the time limitation, demonstrations are incorporated in RL in order to facilitate and guide the learning process. Rajeswaran et al. used a human demonstration and RL to teach a robotic hand dexterous tasks, such as nail hammering [125]. The demonstrations are used to initialise the RL policy, which facilitates convergence towards optimality. The demonstrations are also augmented in the loss function so that the converged policy maintains a similarity to them. In [126], the cobot uses active learning to expand the applicability of a given basic behaviour to convert state A to state B, i.e. perform a certain task. In other words, given state C, the cobot autonomously explored a set of actions that would change it to state A so that the basic behaviour can be applied to convert to state B. Moreover, the cobot autonomously finds suitable perceptual actions that capture useful information about the environment given the task at hand. Their method was also integrated within a GUI that allows the user to easily program the initial basic behaviour [127].

In HRC cases where a human is part of the cobot's environment (observed states) specifically, standard (or "vanilla") RL is used since the operator cannot reasonably complete the numerous learning iterations with the cobot. In [26], cross-training (in which the human and the cobot switch roles during the training process to facilitate learning) is used to learn the reward function for the cobot's collaborative actions. Gu et al. used RL to collaboratively balance a table with a human. The reward, rather than being human feedback, is the change of slope of the table [128]. Sheng et al. added to that a proactive element to predict the human's intention and varies the table's slope accordingly [86].

6. Discussion and conclusion

6.1. Recommendations for industrial parties

Different programming features enable different degrees and forms of cobot autonomy. As cobot autonomy increases, an operator is more likely to feel unease due to the cobot's decreased predictability. However, as the cobot autonomy decreases, the operator is required to make decisions on behalf of both, which increases the mental workload. Therefore, one programming feature is not strictly better than the other, but can be mixed to exploit their benefits while negating, or limiting, their drawbacks. The programming feature supported should also be chosen in light of industrial scenario complexity and the operator's knowledge of the task.

Communication-based programming, where an operator commands or designs the cobot's program through communication mediums, gives a level of direct authority from the operator. Whether that is desirable or not depends on the complexity of the task, the knowledge of the operator and the industrial party's choice. However, it would certainly increase an operator's trust in a cobot and the willingness to work alongside it. This would aid in the introduction and normalisation of cobots in manufacturing. However, communication mediums vary drastically in intuitiveness and reliability. The following are recommendations regarding communication features:

- Attempting to use natural speech and gesture communication in an industrial environment is problematic since there is not enough industry-specific data to train models. Therefore, until natural speech and gesture understanding reaches a reliable level for industrial use, it is advisable to stick to a fixed set of verbal or non-verbal commands which are easier to recognise. However, issuing such commands should not be in each task iteration, since that would be mentally and physically tedious on the operator. But rather, that should be in special cases such as error handling. Moreover, such a communication scheme is especially suitable for Independent and Simultaneous scenarios in which the action sequence is more or less fixed and only occasional interference in required.

- A UI is essential during the operation of a cobot in different scenarios, due to the high chance of needed human interference. UIs are used to pre-specify an action sequence, or trigger an action and specify parameters on-the-fly. If other technologies are used to create/alter the cobot's program, a UI is still necessary to override any of them since it is the most reliable means of controlling the cobot.
- Haptics, coupled with the compliant mode in cobots, is a reliable method to move the cobot as desired. It can be especially useful in Supportive scenarios in which the operator can adjust the cobot's pose as needed. Works such as [25] would ensure that ergonomic benefit is attained during cobot compliance in supportive scenarios. Works such as [84] would ensure that even when the cobot is in compliance, it can follow optimal/learnt paths (achieved through other programming features), to relief the human of repetitively moving the cobot.

In optimisation-based programming, the main decision maker during run-time is the intelligent strategies the cobot is equipped with. Therefore, the mental load on the operator decreases as the cobot gains more autonomy. However, that may also increase the operator's level of discomfort and alertness if the cobot becomes more unpredictable and the operator loses direct authority. The programmer is needed to redesign optimisation objectives and algorithms when optimisation requirements are changed. However, optimisation features remain highly requested by industrial parties that seek optimal performance and outcomes from manufacturing processes. The following are recommendation regarding incorporating optimisation features in cobot programs:

- Optimising for the benefit of the human's physical health and safety is one of the industrial priorities. This includes optimising paths for human collision avoidance, which is especially essential for Independent and Simultaneous tasks. In Sequential (namely handover tasks) and Supportive tasks, works such as [7] can ensure that the cobot's pose yields a healthy human posture.
- Accounting for the human's mental model/state might be regarded as an overshoot by industrial parties. Moreover, since the mental state/model is estimated rather than measured, this presents unnecessary uncertainties in manufacturing processes, especially when a task-level decision is being made. However, during a motion-level decisions, giving indications of the cobot's intent is desirable since it boosts the human's comfort and trust in the cobot. This can be done by moving in a legible path [100], by communicating through light signals [129], by displaying facial expressions [130].
- The question of optimal task distribution is key when discussing HRC, especially Simultaneous and Sequential scenarios. The work in [105] is a good example of distributing tasks according to human and cobot capabilities, based on offline calculations. The work in [29] is a great example on assigning cobot tasks according to real-time conditions, such as positions of objects and the human, while leaving freedom of choice to the human.

Learning-related program features provide autonomy, while enabling the operator to intuitively program the cobot via learning data. The cobot does not need continuous commanding from the operator, yet behaves while showing awareness to the operator's presence and actions. Moreover, the operator can choose to alter the cobot's program by providing new training data, such as new demonstrations for the LfD algorithms. However, as aforementioned, since such algorithms are sensitive to the quality of data, not all data provided by an operator can yield desirable results. Moreover, since policies generated by such algorithms are usually probabilistic, unexpected outlier results might occasionally be encountered. The following are recommendations regarding learning features:

- LfD is the most promising way of teaching a cobot, since it does not require a large set of data to train, can capture a wide range of task dependencies (depending on the chosen LfD algorithm) and is relatively intuitive for operators to perform. Although the process of using LfD is intuitive, operators are still encouraged to understand the theory behind it, since there are many decisions that need to be taken by them that require knowledge of how LfD works.
- Task parametrised LfD is effective at capturing dependencies between different states in the environment, such as positions of objects and humans, and producing a cobot action accordingly. For Independent and Simultaneous scenarios, if objects have predetermined positions, then it is advisable to program the cobot by specifying fixed key points using built-in options in the cobot's teaching pendant. If positions of objects vary, TP-LfD can cater for this variance provided that the cobot is able to detect the positions of these objects. TP-LfD can even cater for position of the human's hand with respect to these objects.
- One problem with LfD is that most algorithms need accurate perception abilities, i.e. the cobot should detect objects in the environment. This requires at least one camera setup, and being mindful of avoiding occlusion and maintaining acceptable lighting conditions. Moreover, the detection of large objects especially with regular shapes and outlines (e.g. plates, boxes, wheels...) can be reliably done using sticker markers (e.g. ArUco). However, small objects (e.g. bolts, batteries...) present a real challenge and might need customised image processing or machine learning vision algorithms. Therefore, it is advisable, given the current state of computer vision solutions available, to use learning programming features when humans and large objects are involved in the task, rather than small objects.
- LfD is good when variations in the positions of objects is expected. However, even when it is not, LfD can be useful for capturing fixed path constrained motions, such as in [112], or capturing compliant motion segments, such as in [84].

Regardless of which programming features are used in the cobot's program, or a combination of thereof, the cobot's safety functionalities such as collision avoidance, should always be overlaid on the program and given priority. Finally, although HRC is becoming increasingly popular in manufacturing, the question of whether it adds value to a manufacturing process is case-dependent. Careful considerations for data security should also be taken. This is especially true when considering computationally expensive algorithms that require data to be transmitted and shared from shop floors to remote clouds.

6.2. Future research directions

The main goal of the technologies, i.e. the programming features, mentioned in Sections 3–5 is to aid an operator in programming a cobot intuitively and/or to enable the cobot to act flexibly around the operator. These two program attributes are what expand the applicability of HRC in a wider range of industrial scenarios. We identify four major research directions that help achieve this goal.

First, researchers should output their work in ready-to-use software and UIs. This requires tighter collaboration with industrial parties and start-ups that are willing to work on creating

products rather than answer research questions. These solutions need to be and approved by industrial standards to relieve industries of legal pressures. Moreover, this requires extensive user-studies. That is because some of the programming algorithms that are deemed intuitive might not be easy to use by operators that do not understand underlying theory. For example, choosing gestures for human–robot communication might be thought of as an easy intuitive task. However, gestures chosen should stand out from other task actions and ensure a high recognition accuracy. Therefore, a systematic, similar to [50], yet automated way of choosing pre-defined gestures is needed. Moreover, setting up LfD algorithms, i.e. recording demonstrations, etc., has been done by researchers that understand the underlying theory of LfD, which helped boost the performance of the algorithms. For operators, however, LfD algorithms should be presented in a more understandable way. This can be through providing UIs that explain algorithms and their parameters to enable non-experts to use these algorithms to train cobots. This can be in the form of guiding and assessing the demonstrations provided for an LfD algorithm, or by presenting the learnt policy in a user-friendly and editable way through a UI. Moreover, even though the "learning" in learning algorithms is automated, the operator, i.e. teacher, is still a main decision maker. He/she records the demonstrations and specifies parameters. For example, in task-parametrised LfD, the operator needs to specify the landmarks with respect to which the cobot path will be encoded. If researchers automate more processes in the learning pipeline such that less decisions are made by the operator, this will yield more optimal results, free of human error. Indeed, learning-related program features have many capabilities and are still rapidly advancing, but need a bridge to enable the non-expert user to fully exploit their capabilities. Therefore, in parallel to working on advancing these algorithm, i.e. increasing training speed, accuracy and applications, it is important to recognise the specific industrial requirements and creating UIs that allow industrial parties to utilise these technologies.

Another main research direction is evaluating technologies in experimental setups that are similar to industrial scenarios. Experimental setups are often overly simplified versions of industrial scenarios. The main reason behind this is limitations in perception abilities. When perception is irrelevant to the research question at hand, researchers choose to utilise easy vision solutions such that objects and tools in the experiment are simplified. Therefore, bringing reliable perception solutions for industrial scenarios is necessary, such as Zidek et al.'s convolutional neural network for detecting generic industrial parts (screws, nuts, etc.) [131]. This brings forward the need for more annotated data relevant to industrial collaborative tasks, potentially through a platform to share and evaluate data sets. However, a challenge arises since such data related to the industry comes with intellectual property restrictions that might render it confidential. Alternatively, non-supervised or semi-supervised learning algorithms that do not require ground labelled data can be utilised. For some applications, such as activity recognition in an work cell, non/semi-supervised algorithms are proven to perform almost as accurately as supervised algorithms [132].

The third setback of deploying cobots with intelligent programs is the lack of trust and understanding from operators and industrial parties. Therefore, a needed research direction, applicable to communication, optimisation and learning, is to improve the representation of a cobot's mental model. Most HRC programs cannot be briefly summed up and represented in a set of rule-based commands. This makes understanding errors and predicting behaviour a difficult task for operators. For that, several research efforts revolved around providing visual indications around cobot motion intent [133], which presents intent in rough granularity, or verbal description of the cobot's policy [134], which does not operate continuously but only when prompted. Therefore, an advisable direction of research is to display a cobot's "thought process" and plan real time in a digital-twin user-interface. This should be done after surveying industrial needs and requirements as to what elements of the cobot's mental model should be displayed. For example, representing a cobot's mental model can include an interface showing perception information, i.e. detected objects and recognised human actions. Not only can it show what is detected, but also more detailed cues concerning the detection accuracy and prominent visual features that aided with detection. This would help the operator implicitly understand the perception abilities of the cobot and its limitation. The operator will be able to move and hold objects to maximise the cobot's perception. This brings forward the need for a flexible ontological representation of HRC tasks. This ontological representation would be needed to overlay preset rules on probabilistic policies obtained from optimisation problems or machine learning algorithms. These preset rules would increase the industry's confidence in non-traditional cobot programs, i.e. those incorporating probabilistic command outcomes.

6.3. Conclusion

Human–robot collaboration and cobots are emerging technologies in supporting increasingly flexible and complex manufacturing processes. Among which, intuitive and human-aware programming technologies that can support cobots in collaborating with human operators more intelligently and adaptively, are the key enablers. In this topic, research has been actively conducted in recent years. In this paper, the latest research on cobot programming is summarised. Firstly, an overview of collaborative industrial scenarios and programming requirements for cobot applications is provided. Then, the cobot programming technologies are categorised into three features, that are communication, optimisation and learning features, and relevant research works on the defined features are reviewed in detail. Communication features enable a collaborative human operator to transfer intent or commands directly to a cobot, thus affecting its course of action to support collaboration. Optimisation features are algorithms developed by programmers on-line enabling a cobot to observe its collaborative operator and behave adaptively according to a pre-modelled optimised policy. Learning features allow a cobot to learn its own policy after receiving guidance from its collaborative operator. In the review, how intuitive and human-aware elements are considered in the communication, optimisation and learning features are elaborated. Furthermore, gaps from viewpoints of industrial requirements and the-state-of-art research for cobot programming are identified. Finally, future research directions and recommendations for cobot programming to better support industrial collaborative scenarios are outlined.

Acknowledgements

This work is part of a project funded by Coventry University, UK, Unipart Powertrain Applications ltd., UK and High Speed Sustainable Manufacturing Institute (HSSMI), UK.

References

[1] R. Muller, M. Vette, O. Mailahn, Process-oriented task assignment for assembly processes with human–robot interaction, Proc. CIRP 44 (2016) 210–215.

[2] M. Rußmann, M. Lorenz, P. Gerbert, M. Waldner, J. Justus, P. Engel, M. Harnisch, Industry 40: The future of productivity and growth in manufacturing industries, vol. 9, Boston Consulting Group, 2015.

[3] A. Bicchi, M.A. Peshkin, J.E. Colgate, Safety for physical human–robot interaction, in: B. Siciliano, O. Khatib (Eds.), Springer Handbook of Robotics, Springer Berlin Heidelberg, 2008, pp. 1335–1348.

[4] Universal Robotics, Our history, 2018, [Online]. Available: https://www.universal-robots.com/about-universal-robots/our-history/.

[5] Robotiq, Cobots EBook, 2018, [Online]. Available: https://blog.robotiq.com/collaborative-robot-ebook.

[6] Robots and robotic devices – Collaborative robots, ISO Standard ISO/TS 15066 (2016) 2016.

[7] L. Peternel, W. Kim, J. Babic, A. Ajoudani, Towards ergonomic control of human–robot co-manipulation and handover, in: 2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids), IEEE, 2017.

[8] B. Chandrasekaran, J.M. Conrad, Human–robot collaboration: A survey, in: SoutheastCon 2015, IEEE, 2015, pp. 1–8.

[9] P.A. Lasota, T. Fong, J.A. Shah, et al., A survey of methods for safe human–robot interaction, Found. Trends Robot. 5 (4) (2017) 261–349.

[10] J. Lee, A survey of robot learning from demonstrations for humanrobot collaboration, 2017, arXiv preprint arXiv:1710.08789.

[11] Z. Zhu, H. Hu, Robot learning from demonstration in robotic assembly: A survey, Robotics 7 (2) (2018) 17.

[12] H. Liu, L. Wang, Gesture recognition for human–robot collaboration: A review, Int. J. Ind. Ergon. 68 (2018) 355–367.

[13] S.A. Green, M. Billinghurst, X. Chen, J.G. Chase, Human–robot collaboration: A literature review and augmented reality approach in design, Int. J. Adv. Robot. Syst. 5 (1) (2008) 1–18.

[14] A. Bauer, D. Wollherr, M. Buss, Human–robot collaboration: A survey, Int. J. Humanoid Robot. 5 (1) (2008) 47–66.

[15] S. Haddadin, E. Croft, Physical human–robot interaction, in: B. Siciliano, O. Khatib (Eds.), Springer Handbook of Robotics, Springer International Publishing, 2016, pp. 1835–1874.

[16] N. Sylla, S. Mehta, Implementation of Collaborative Robot Applications: A Report from the Industrial Working Group, Tech. Rep., High Speed Sustainable Manufacturing Institute, 2017.

[17] A. Cesta, A. Orlandini, G. Bernardi, A. Umbrico, Towards a planning-based framework for symbiotic human–robot collaboration, in: 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), 2016, pp. 1–8.

[18] T. Munzer, M. Toussaint, M. Lopes, Efficient behavior learning in human–robot collaboration, Auton. Robots 42 (5) (2017) 1103–1115.

[19] E. Commission, Periodic reporting for period 1 - colrobot (collaborative robotics for assembly and kitting in smart manufacturing), Tech. Rep., 2018.

[20] BMW Group, Innovative human–robot cooperation in BMW group production, 2013, [Online]. Available: https://www.press.bmwgroup.com/global/article/detail/T0209722EN/innovative-human--robot-cooperation-in-bmw-group-production?language=en.

[21] N. Winkelmann, Human–robot cooperation at Audi, 2017, [Online]. Available: https://www.springerprofessional.de/en/manufacturing/production---production-technology/human--robot-cooperation-at-audi/14221870.

[22] KUKA, Many wrenches make light work: KUKA flexFELLOW will provide assistance during drive train pre-assembly, 2016, [Online]. Available: https://www.kuka.com/en-gb/press/news/2016/10/20160926vwsetztaufmensch-roboter-kollaboration.

[23] Universal Robots, UR10 Cobots offer aging workforce solution and reduce relief worker costs for global car manufacturer, 2018, [Online]. Available: https://www.universal-robots.com/case-stories/nissan-motor-company/.

[24] Robotics and Automation News, Innovative skoda factory introduces human–robot collaboration with KUKA LBR iiwa, 2017, [Online]. Available: https://roboticsandautomationnews.com/2017/02/16/innovative-skoda-factory-introduces-human--robot-collaboration-with-kuka-lbr-iiwa/11404/.

[25] S. Lichiardopol, N. van de Wouw, H. Nijmeijer, Control scheme for human–robot co-manipulation of uncertain, time-varying loads, in: 2009 American Control Conference, 2009, pp. 1485–1490.

[26] S. Nikolaidis, P. Lasota, R. Ramakrishnan, J. Shah, Improved human–robot team performance through cross-training, an approach inspired by human team training practices, Int. J. Robot. Res. 34 (14) (2015) 1711–1730.

[27] C.-M. Huang, M. Cakmak, B. Mutlu, Adaptive coordination strategies for human–robot handovers, in: Robotics: Science and Systems, 2015.

[28] L. Johannsmeier, S. Haddadin, A hierarchical human–robot interaction-planning framework for task allocation in collaborative industrial assembly processes, IEEE Robot. Autom. Lett. 2 (1) (2017) 41–48.

[29] V. Gabler, T. Stahl, G. Huber, O. Oguz, D. Wollherr, A game theoretic approach for adaptive action selection in close proximity human robot collaboration, in: IEEE International Conference on Robotics and Automation, 2017.

[30] G. Maeda, A. Maloo, M. Ewerton, R. Lioutikov, J. Peters, Anticipative interaction primitives for human–robot collaboration, in: 2016 AAAI Fall Symposium Series, 2016.

[31] M. Wongphati, H. Osawa, M. Imai, Gestures for manually controlling a helping hand robot, Int. J. Soc. Robot. 7 (5) (2015) 731–742.

[32] C. Lenz, M. Rickert, G. Panin, A. Knoll, Constraint task-based control in industrial settings, in: 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009, pp. 3058–3063.

[33] I.E. Makrini, K. Merckaert, D. Lefeber, B. Vanderborght, Design of a collaborative architecture for human–robot assembly tasks, in: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017, pp. 1624–1629.

[34] K.R. Guerin, S.D. Riedel, J. Bohren, G.D. Hager, Adjutant: A framework for flexible human-machine collaborative systems, in: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2014, pp. 1392–1399.

[35] A. Cherubini, R. Passama, P. Fraisse, A. Crosnier, A unified multimodal control framework for human–robot interaction, Robot. Auton. Syst. 70 (2015) 106–115.

[36] H. Ding, M. Schipper, B. Matthias, Collaborative behavior design of industrial robots for multiple human–robot collaboration, in: 2013 44th International Symposium on Robotics (ISR), IEEE, 2013, pp. 1–6.

[37] R. Meziane, M.J.-D. Otis, H. Ezzaidi, Human–robot collaboration while sharing production activities in dynamic environment: SPADER system, Robot. Comput.-Integr. Manuf. 48 (2017) 243–253.

[38] Robots and robotic devices Safety requirements for industrial robots Part 1: Robots, ISO Standard ISO 10 218-1 (2011) 2011.

[39] A. Realyvasquez-Vargas, K.C. Arredondo-Soto, J.L. Garcia-Alcaraz, B.Y. Marquez-Lobato, J. Cruz-Garcia, Introduction and configuration of a collaborative robot in an assembly task as a means to decrease occupational risks and increase efficiency in a manufacturing company, Robot. Comput.-Integr. Manuf. 57 (2019) 315–328.

[40] L. Wang, B. Schmidt, A.Y.C. Nee, Vision-guided active collision avoidance for human–robot collaborations, Manuf. Lett. 1 (1) (2013) 5–8.

[41] B. Schmidt, L. Wang, Depth camera based collision avoidance via active robot control, J. Manuf. Syst. 33 (4) (2014) 711–7118.

[42] Y. Wang, X. Ye, Y. Yang, W. Zhang, Collision-free trajectory planning in human–robot interaction through hand movement prediction from vision, in: 2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids), 2017, pp. 305–310.

[43] K.H. Dinh, O. Oguz, G. Huber, V. Gabler, D. Wollherr, An approach to integrate human motion prediction into local obstacle avoidance in close human–robot collaboration, in: 2015 IEEE International Workshop on Advanced Robotics and its Social Impacts (ARSO), 2015, pp. 1–6.

[44] E. Matsas, G.C. Vosniakos, D. Batras, Prototyping proactive and adaptive techniques for human–robot collaboration in manufacturing using virtual reality, Robot. Comput.-Integr. Manuf. 50 (2018) 168–180.

[45] P. Gustavsson, A. Syberfeldt, R. Brewster, L. Wang, Humanrobot collaboration demonstrator combining speech recognition and haptic control, Proc. CIRP 63 (2017) 396–401.

[46] D. Vogt, S. Stepputtis, S. Grehl, B. Jung, H.B. Amor, A system for learning continuous human–robot interactions from human-human demonstrations, in: IEEE International Conference on Robotics and Automation (ICRA), 2017, pp. 2882–2889.

[47] C. Pohlt, S. Hell, T. Schlegl, S. Wachsmuth, Impact of spontaneous human inputs during gesture based interaction on a real-world manufacturing scenario, in: Proceedings of the 5th International Conference on Human Agent Interaction, ACM, 2017, pp. 347–351.

[48] B. Gleeson, K. MacLean, A. Haddadi, E. Croft, J. Alcazar, Gestures for industry intuitive human–robot communication from human observation, in: 2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI), 2013, pp. 349–356.

[49] E. Calisgan, A. Haddadi, H.F.M.V. der Loos, J.A. Alcazar, E.A. Croft, Identifying nonverbal cues for automated human–robot turn-taking, in: 2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication, 2012, pp. 418–423.

[50] P. Barattini, C. Morand, N.M. Robertson, A proposed gesture set for the control of industrial collaborative robots, in: 2012 IEEE ROMAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication, 2012, pp. 132–137.

[51] Y. Cheng, J. Bao, Y. Jia, Z. Deng, Z. Sun, S. Bi, C. Li, N. Xi, Modelling robotic operations controlled by natural language, Control Theory Technol. 15 (4) (2017) 258–266.

[52] L. She, J. Chai, Interactive learning of grounded verb semantics towards human–robot communication, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), vol. 1, 2017, pp. 1634–1644.

[53] I. Maurtua, I. Fernndez, A. Tellaeche, J. Kildal, L. Susperregi, A. Ibarguren, B. Sierra, Natural multimodal communication for human–robot collaboration, Int. J. Adv. Robot. Syst. 14 (4) (2017) 1–12.

[54] S. Nakata, H. Kobayashi, T. Yasuda, M. Kumata, S. Suzuki, H. Igarashi, Relation between skill acquisition and task specific human speech in collaborative work, in: 2011 RO-MAN, 2011, pp. 337–342.

[55] H. Kobayashi, T. Yasuda, H. Igarashi, S. Suzuki, Language use in joint action: the means of referring expressions, Int. J. Soc. Robot. (2018) 1–9.

[56] P.A.S. Srimal, M.V.J. Muthugala, A.B.P. Jayasekara, Deictic gesture enhanced fuzzy spatial relation grounding in natural language, in: Fuzzy Systems (FUZZ-IEEE), 2017 IEEE International Conference on, IEEE, 2017, pp. 1–8.

[57] M. Giuliani, A. Knoll, Using embodied multimodal fusion to perform supportive and instructive robot roles in human–robot interaction, Int. J. Soc. Robot. 5 (3) (2013) 345–356.

[58] H. Admoni, T. Weng, B. Hayes, B. Scassellati, Robot nonverbal behavior improves task performance in difficult collaborations, in: 2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI), 2016, pp. 51–58.

[59] D. Shukla, O. Erkent, J. Piater, Proactive, incremental learning of gesture-action associations for human–robot collaboration, in: 2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), 2017, pp. 346–353.

[60] C.M. Huang, B. Mutlu, Anticipatory robot control for efficient human–robot collaboration, in: 2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI), 2016, pp. 83–90.

[61] M. Benzeghiba, R.D. Mori, O. Deroo, S. Dupont, T. Erbes, D. Jouvet, L. Fissore, P. Laface, A. Mertins, C. Ris, R. Rose, V. Tyagi, C. Wellekens, Automatic speech recognition and speech variability: A review, Speech Commun. 49 (10) (2007) 763–786.

[62] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, Y. Sheikh, OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields, 2018, arXiv preprint arXiv:1812.08008.

[63] D. Kumicakova, A. Rengevic, M. Cisar, V. Tlach, Utilisation of kinect sensors for the design of a human–robot collaborative workcell, Adv. Sci. Technol. Res. J. 11 (4) (2017) 270–278.

[64] M.R. Pedersen, D.L. Herzog, V. Kruger, Intuitive skill-level programming of industrial handling tasks on a mobile manipulator, in: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2014, pp. 4523–4530.

[65] J. de Gea Fernandez, D. Mronga, M. Gunther, T. Knobloch, M. Wirkus, M. Schroer, M. Trampler, S. Stiene, E. Kirchner, V. Bargsten, T. Banziger, J. Teiwes, T. Kruger, F. Kirchner, Multimodal sensor-based whole-body control for human–robot collaboration in industrial settings, Robot. Auton. Syst. 94 (2017) 102–119.

[66] X. Chen, X. Zhang, Z.Y. Zhao, J.H. Yang, V. Lantz, K.Q. Wang, Multiple hand gesture recognition based on surface EMG signal, in: 2007 1st International Conference on Bioinformatics and Biomedical Engineering, 2007, pp. 506–509.

[67] S. Nakata, H. Kobayashi, M. Kumata, S. Suzuki, Human speech ontology changes in virtual collaborative work, in: 2011 4th International Conference on Human System Interactions (HSI), 2011, pp. 363–368.

[68] G.F. Rossano, C. Martinez, M. Hedelind, S. Murphy, T.A. Fuhlbrigge, Easy robot programming concepts: An industrial perspective, in: 2013 IEEE International Conference on Automation Science and Engineering (CASE), 2013, pp. 1119–1126.

[69] C. Schou, J.S. Damgaard, S. Bogh, O. Madsen, Human–robot interface for instructing industrial tasks using kinesthetic teaching, in: 2013 44th International Symposium on Robotics (ISR), IEEE, 2013, pp. 1–6.

[70] C. Schou, R.S. Andersen, D. Chrysostomou, S. Bogh, O. Madsen, Skill-based instruction of collaborative robots in industrial settings, Robot. Comput.-Integr. Manuf. 53 (2018) 72–80.

[71] F. Steinmetz, R. Weitschat, Skill parametrization approaches and skill architecture for human–robot interaction, in: 2016 IEEE International Conference on Automation Science and Engineering (CASE), 2016, pp. 280–285.

[72] M.R. Pedersen, L. Nalpantidis, R.S. Andersen, C. Schou, S. Bgh, V. Kruger, O. Madsen, Robot skills for manufacturing: From concept to industrial deployment, Robot. Comput.-Integr. Manuf. 37 (2016) 282–291.

[73] F. Steinmetz, A. Wollschlager, R. Weitschat, RAZER: A HRI For visual task-level programming and intuitive skill parameterization, IEEE Robot. Autom. Lett. 3 (3) (2018) 1362–1369.

[74] P.J. Koch, P.D. Marike K. van Amstel, M.A. Thormann, A.J. Tetzlaff, S. Bogh, D. Chrysostomou, A skill-based robot co-worker for industrial maintenance tasks, Proc. Manuf. 11 (2017) 83–90.

[75] C. Paxton, A. Hundt, F. Jonathan, K. Guerin, G.D. Hager, Costar: Instructing collaborative robots with behavior trees and vision, in: 2017 IEEE International Conference on Robotics and Automation (ICRA), 2017, pp. 564–571.

[76] E. Noohi, M. Zefran, J.L. Patton, A model for human-human collaborative object manipulation and its application to human–robot interaction, IEEE Trans. Robot. 32 (4) (2016) 880–896.

[77] E. Magrini, F. Flacco, A.D. Luca, Control of generalized contact motion and force in physical human–robot interaction, in: 2015 IEEE International Conference on Robotics and Automation (ICRA), 2015, pp. 2298–2304.

[78] A. Kouris, F. Dimeas, N. Aspragathos, A frequency domain approach for contact type distinction in human–robot collaboration, IEEE Robot. Autom. Lett. 3 (2) (2018) 720–727.

[79] C. Gaz, E. Magrini, A.D. Luca, A model-based residual approach for human–robot collaboration during manual polishing operations, Mechatronics 55 (2018).

[80] T. Wojtara, M. Uchihara, H. Murayama, S. Shimoda, S. Sakai, H. Fujimoto, H. Kimura, Human–robot collaboration in precise positioning of a three-dimensional object, Automatica 45 (2) (2009) 333–342.

[81] J. Dumora, F. Geffard, C. Bidard, N.A. Aspragathos, P. Fraisse, Robot assistance selection for large object manipulation with a human, in: 2013 IEEE International Conference on Systems, Man, and Cybernetics, 2013, pp. 1828–1833.

[82] Y. Li, S.S. Ge, Human–robot collaboration based on motion intention estimation, IEEE/ASME Trans. Mechatronics 19 (3) (2014) 1007–1014.

[83] Y. Li, K.P. Tee, W.L. Chan, R. Yan, Y. Chua, D.K. Limbu, Role adaptation of human and robot in collaborative tasks, in: 2015 IEEE International Conference on Robotics and Automation (ICRA), 2015, pp. 5602–5607.

[84] L. Rozo, S. Calinon, D.G. Caldwell, P. Jimenez, C. Torras, Learning physical collaborative robot behaviors from human demonstrations, IEEE Trans. Robot. 32 (3) (2016) 513–527.

[85] D.J. Agravante, A. Cherubini, A. Bussy, P. Gergondet, A. Kheddar, Collaborative human-humanoid carrying using vision and haptic sensing, in: 2014 IEEE International Conference on Robotics and Automation (ICRA), 2014, pp. 607–612.

[86] W. Sheng, A. Thobbi, Y. Gu, An integrated framework for human–robot collaborative manipulation, IEEE Trans. Cybern. 45 (10) (2015) 2030–2041.

[87] V.V. Unhelkar, X.J. Yang, J.A. Shah, Challenges for communication decision-making in sequential human–robot collaborative tasks, in: Workshop on Mathematical Models, Algorithms, and Human-Robot Interaction at R: SS, 2017.

[88] B. Busch, G. Maeda, Y. Mollard, M. Demangeat, M. Lopes, Postural optimization for an ergonomic human–robot interaction, in: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017, pp. 2778–2785.

[89] L. Peternel, N. Tsagarakis, D. Caldwell, A. Ajoudani, Adaptation of robot physical behaviour to human fatigue in human–robot co-manipulation, in: 2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids), 2016, pp. 489–494.

[90] B. Hu, J. Chen, Optimal task allocation for human-machine collaborative manufacturing systems, IEEE Robot. Autom. Lett. 2 (4) (2017) 1933–1940.

[91] W. Kim, J. Lee, L. Peternel, N. Tsagarakis, A. Ajoudani, Anticipatory robot assistance for the prevention of human static joint overloading in human–robot collaboration, IEEE Robot. Autom. Lett. 3 (1) (2018) 68–75.

[92] G. Milliez, R. Lallement, M. Fiore, R. Alami, Using human knowledge awareness to adapt collaborative plan generation, explanation and monitoring, in: 2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI), 2016, pp. 43–50.

[93] S. Devin, R. Alami, An implemented theory of mind to improve human–robot shared plans execution, in: 2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI), 2016, pp. 319–326.

[94] H. Zhu, V. Gabler, D. Wollherr, Legible action selection in human–robot collaboration, in: 2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), 2017, pp. 354–359.

[95] J. Baraglia, M. Cakmak, Y. Nagai, R. Rao, M. Asada, Initiative in robot assistance during collaborative task execution, in: 2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI), 2016, pp. 67–74.

[96] R. Schulz, P. Kratzer, M. Toussaint, Building a bridge with a robot: A system for collaborative on-table task execution, in: Proceedings of the 5th International Conference on Human Agent Interaction, ACM, 2017, pp. 399–403.

[97] B. Sadrfaridpour, H. Saeidi, Y. Wang, An integrated framework for human–robot collaborative assembly in hybrid manufacturing cells, in: 2016 IEEE International Conference on Automation Science and Engineering (CASE), 2016, pp. 462–467.

[98] P. Maurice, M.E. Huber, N. Hogan, D. Sternad, Velocitycurvature patterns limit human–robot physical interaction, IEEE Robot. Autom. Lett. 3 (1) (2018) 249–256.

[99] C. Bodden, D. Rakita, B. Mutlu, M. Gleicher, Evaluating intentexpressive robot arm motion, in: 2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), 2016, pp. 658–663.

[100] B. Busch, J. Grizou, M. Lopes, F. Stulp, Learning legible motion from human–robot interactions, Int. J. Soc. Robot. 9 (5) (2017) 765–779.

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

[102] M. Faber, A. Mertens, C.M. Schlick, Cognition-enhanced assembly sequence planning for ergonomic and productive human–robot collaboration in self-optimizing assembly cells, Prod. Eng. 11 (2) (2017) 145–154.

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

[104] M. Gombolay, A. Bair, C. Huang, J. Shah, Computational design of mixed-initiative human–robot teaming that considers human factors: situational awareness, workload, and workflow preferences, Int. J. Robot. Res. 36 (5–7) (2017) 597–617.
[105] T. Banziger, A. Kunz, K. Wegener, Optimizing human–robot task allocation using a simulation tool based on standardized work descriptions, J. Intell. Manuf. (2018) 1–14.
[106] K. Fischer, F. Kirstein, L.C. Jensen, N. Kruger, K. Kuklinski, M.V. aus der Wieschen, T.R. Savarimuthu, A comparison of types of robot control for programming by demonstration, in: 2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI), 2016, pp. 213–220.
[107] B. Akgun, K. Subramanian, Robot learning from demonstration: Kinesthetic teaching vs. teleoperation.
[108] J.F. Lafleche, S. Saunderson, G. Nejat, Robot cooperative behavior learning using single-shot learning from demonstration and parallel hidden Markov models, IEEE Robot. Autom. Lett. 4 (2) (2019) 193–200.
[109] C. Yang, C. Zeng, P. Liang, Z. Li, R. Li, C.Y. Su, Interface design of a physical human–robot interaction system for human impedance adaptive skill transfer, IEEE Trans. Autom. Sci. Eng. 15 (1) (2018) 329–340.
[110] Y. Gu, W. Sheng, C. Crick, Y. Ou, Automated assembly skill acquisition and implementation through human demonstration, Robot. Auton. Syst. 99 (2018) 1–16.
[111] A.M. Ghalamzan, M. Ragaglia, Robot learning from demonstrations: Emulation learning in environments with moving obstacles, Robot. Auton. Syst. 101 (2018) 45–56.
[112] C. Perez-D'Arpino, J.A. Shah, C-learn: Learning geometric constraints from demonstrations for multi-step manipulation in shared autonomy, in: 2017 IEEE International Conference on Robotics and Automation (ICRA), 2017, pp. 4058–4065.
[113] S. Calinon, A tutorial on task-parameterized movement learning and retrieval, Intell. Serv. Robot. 9 (1) (2016) 1–29.
[114] D.A. Duque, F.A. Prieto, J. G.Hoyos, Trajectory generation for robotic assembly operations using learning by demonstration, Robot. Comput.-Integr. Manuf. 57 (2019) 292–302.
[115] Y.S. Liang, D. Pellier, H. Fiorino, S. Pesty, Evaluation of a robot programming framework for non-experts using symbolic planning representations, in: 2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), 2017, pp. 1121–1126.
[116] T. Hamabe, H. Goto, J. Miura, A programming by demonstration system for human–robot collaborative assembly tasks, in: 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO), 2015, pp. 1195–1201.
[117] G. Maeda, M. Ewerton, G. Neumann, R. Lioutikov, J. Peters, Phase estimation for fast action recognition and trajectory generation in human–robot collaboration, Int. J. Robot. Res. 36 (13–14) (2017) 1579–1594.
[118] M. Forbes, M.J.-Y. Chung, M. Cakmak, R.P. Rao, Robot programming by demonstration with crowdsourced action fixes, in: Second AAAI Conference on Human Computation and Crowdsourcing, 2014.
[119] R. Luo, R. Hayne, D. Berenson, Unsupervised early prediction of human reaching for human–robot collaboration in shared workspaces, Auton. Robots 42 (3) (2018) 631–648.
[120] V. Mohan, A.A. Bhat, Joint goal human robot collaboration from remembering to inferring, Procedia Comput. Sci. 123 (2018) 579–584.
[121] N. Heess, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, A. Eslami, M. Riedmiller, et al., Emergence of locomotion behaviours in rich environments, 2017, arXiv preprint arXiv:1707.02286.
[122] S. Gu, E. Holly, T. Lillicrap, S. Levine, Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates, in: 2017 IEEE International Conference on Robotics and Automation (ICRA) (2017) 3389–3396.
[123] P. Abbeel, A. Coates, M. Quigley, A.Y. Ng, An application of reinforcement learning to aerobatic helicopter flight, in: Advances in neural information processing systems, 2007, pp. 1–8.
[124] S. Levine, P. Pastor, A. Krizhevsky, D. Quillen, Learning handeye coordination for robotic grasping with deep learning and large-scale data collection, CoRR, abs/1603.02199, 2016.
[125] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, S. Levine, Learning complex dexterous manipulation with deep reinforcement learning and demonstrations, 2017, arXiv preprint arXiv:1709.10087.
[126] S. Hangl, V. Dunjko, H.J. Briegel, J. Piater, Skill learning by autonomous robotic playing using active learning and creativity, 2017, arXiv preprint arXiv:1706.08560.
[127] S. Hangl, A. Mennel, J. Piater, A novel skill-based programming paradigm based on autonomous playing and skill-centric testing, 2017, arXiv preprint arXiv:1709.06049.
[128] Y. Gu, A. Thobbi, W. Sheng, Human–robot collaborative manipulation through imitation and reinforcement learning, in: 2011 IEEE International Conference on Information and Automation (ICIA), 2011, pp. 151–156.
[129] G. Tang, P. Webb, J. Thrower, The development and evaluation of robot light skin: A novel robot signalling system to improve communication in industrial humanrobot collaboration, Robot. Comput.-Integr. Manuf. 56 (2019) 85–94.
[130] M.E. Reyes, I.V. Meza, L.A. Pineda, Robotics facial expression of anger in collaborative humanrobot interaction, Int. J. Adv. Robot. Syst. 16 (1) (2019).
[131] K. Zidek, A. Hosovsky, J. Pitel, S. Bednar, Recognition of assembly parts by convolutional neural networks, in: Advances in Manufacturing Engineering and Materials, 2019, pp. 281–289.
[132] D.J. Rude, S. Adams, P.A. Beling, Task recognition from joint tracking data in an operational manufacturing cell, J. Intell. Manuf. 29 (6) (2015) 1203–1217.
[133] G. Bejerano, G. LeMasurier, H.A. Yanco, Methods for providing indications of robot intent in collaborative human–robot tasks, in: Companion of the 2018 ACM/IEEE International Conference on Human-Robot Interaction, 2018, pp. 65–66.
[134] B. Hayes, J.A. Shah, Improving robot controller transparency through autonomous policy explanation, in: Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction (HRI), 2017, pp. 303–312.

Shirine El Zaatari is currently a PhD researcher at the Institute of Advanced Manufacturing and Engineering in Coventry University. She graduated with a Bachelor's of Engineering in Mechanical Engineering from the American University of Beirut with a focus in robotics. Shirine gained research experience in computer vision as a visiting student in King Abdallah University of Science and Technology. Her current research interests revolve around using learning from demonstration to program collaborative robots for industrial tasks.

Mohamed Marei is a PhD researcher at the Institute of Advanced Manufacturing and Engineering in Coventry University, researching real-time monitoring and optimisation of production lines and predictive plans. His main research focus is cloud real-time predictive maintenance capabilities for manufacturing processes, through developing intelligent digital twin models of manufacturing assets. He graduated from the University of Sheffield in 2016, with a Master of Engineering (MEng) in Mechatronic and Robotic Engineering. He gained research experience in Sheffield Robotics, where he researched implementing computer vision for a system of self-reconfigurable modular robots. His other research interests are in the areas of Industry 4.0, industrial automation, and robotics in manufacturing.

Weidong Li received the Ph.D. degree from the Mechanical Engineering at National University of Singapore in 2002. He is a full professor and an academic leader at Research Center for Advanced Manufacturing and Engineering, Coventry University, U.K. Before joining Coventry in 2007, he worked at Singapore Institute of Manufacturing Technology, University of Bath, and Cranfield University as a researcher and academic. His primary research interests include computer aided manufacturing and automation, sustainable manufacturing, and Big Data analytics for smart manufacturing. In the past decade, he has participated in a number of EU projects in the areas of sustainable or digital manufacturing, and cooperated with automotive, aeronautical and manufacturing industries (e.g., Airbus, Jaguar Land Rover, Sandvik, and some manufacturing SMEs). In the research areas, he has published 150 research papers in international journals and conferences, and 4 books (Springer and World Scientific Publisher).

Dr Zahid Usman, CEng, MIMechE, is currently working as Manufacturing Systems Lead Engineer in Rolls Royce plc. Before this he was working as a Lecturer in Robotics and Automation at Coventry University. He has extensive industrial and academic experience within the fields of Manufacturing Informatics, Robotics and Automation and Systems Engineering. He has lead several research and industrial projects with world leading academic institutes and industries. He has published over 20 peer reviewed research articles. Dr Usman is a reviewer for several international journals and has been on technical committees for various journals and conferences. Dr Usman continues to be involved in academic research and applied work in industry.