Integration of Robotic Technologies for Rapidly Deployable Robots

Emmanuel Dean-Leon, *Member, IEEE*, Karinne Ramirez-Amaro, *Member, IEEE*, Florian Bergner, *Member, IEEE*, Ilya Dianov, *Member, IEEE*, and Gordon Cheng, *Fellow, IEEE*

***Abstract*—The automation of production lines in industrial scenarios implies solving different problems, such as the *flexibility* to deploy robotic solutions to different production lines, *usability* to allow nonrobotics expert users to teach robots different tasks, and *safety* to enable operators to physically interact with robots without the need of fences. In this paper, we present a system that integrates three novel technologies to address the above mentioned problems. We use an autocalibrated multimodal *robot skin*, a general *robot control framework* to generate dynamic behaviors fusing multiple sensor signals, and an intuitive and fast *teaching by demonstration* method based on semantic reasoning. We validate the proposed technologies with a wheeled humanoid robot in an industrial set-up. The benefits of our system are the transferability of the learned tasks to different robots, the reusability of the models when new objects are introduced in the production line, the capability of detecting and recovering from errors, and the reliable detection of *collisions* and *precollisions* to provide a fast reactive robot that improves the physical human-robot interaction.**

***Index Terms*—Multimodal control, physical human-robot interaction (pHRI), robot skin, semantic reasoning, teaching by demonstration.**

I. Introduction

THE demand for an increasingly high productivity level in industrial scenarios requires both, shorter task execution times and faster/easier robotic programming methods, which reduce the production costs. An automated process using robots needs to be programmed to perform as efficient as a human worker in various domains, for example in packing and quality checking of products. However, setting up a robotic system takes, in general, at least three months [1] implying the need of robot expert programmers with higher costs. These factors are more prominent for small and medium enterprises (SMEs) since they usually have small production batches and have to cope with more frequent changes in production processes. This problem is not only limited to SMEs but also affects major enterprises (MEs), which in general undergo a shift from *mass production* to *mass customization* increasing the overall need for more flexible production lines and fast effortless reconfigurations [2]. The successful automation of these production processes demands *flexible*, *usable*, and *safer* robotic solutions [3], [4]. *Flexibility* implicates that robotic systems have to be quickly deployable with short installation times, to be easy to move to different production sites and to allow quick and easy adjustments to current production needs. *Usability* implicates simple and intuitive programming methods, enabling nonexperts, and untrained personnel to effortlessly reconfigure the system in a natural way. *Safety* entails that systems incorporate new principles to provide the necessary safety[1] for human operators during physical interactions in shared workspaces. Combining all these requirements leads to *Robot Transparency*. Ideally, a robot is considered fully *transparent* when the deployment of the robot does not produce any changes (disruptions) in the production line. *Robot Transparency* can be measured by the effort needed to deploy the robot, such as safety mechanisms, personnel training, and changes in the production process. *Transparent Robots* allow human-robot collaborations, just as if they were human-human collaborations since the robot will have ideally the same set of skills and requirements as a human co-worker – in the context of a *specific* production process. The high adaptability and accuracy of human-robot collaborations facilitate the automation of industrial processes for both SMEs and MEs. Physical human-robot interaction (pHRI) [5] is a fundamental aspect of *Robot Transparency* as well as simple and intuitive teaching methods, for example, programming by demonstration techniques (PbD). This sort of teaching methods allow the operator to teach the robot tasks in an easy and natural way [6], hence, an expert robot programmer is not required, see Fig. 1. Therefore, the development and integration of technologies, such as *robot skin*, reactive control schemes, and robust teaching methods are needed to simplify the robot programming, to improve the physical interaction with robots, and to decrease the deployment time of robotic systems in shop floors, i.e., to increase the *Robot Transparency*.

A. Related Work

Programming robots can be done by manually guiding the robot to the desired position through physical (direct) or

[1] More concretely, with *safety*, we mean avoiding dangerous collisions during a physical human-robot interaction.

Manuscript received May 30, 2017; revised August 23, 2017; accepted September 29, 2017. Paper no. TII-17-1139. *(Corresponding author: Emmanuel Dean-Leon).*

The authors are with the Institute for Cognitive Systems, Technische Universität München, Munich 80333 Germany (e-mail: dean@tum.de; karinne.ramirez@tum.de; florian.bergner@tum.de; dianoviv@gmail.com; gordon@tum.de).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2017.2766096

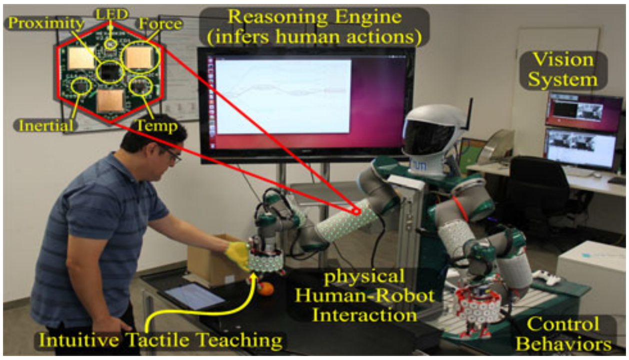


Fig. 1. Demonstration scenario: The user can intuitively teach a complete process to a robot. The setup consists of a perception system (fusing *robot skin* and vision), multimodal robot behaviors, and a reasoning system.

cyber-physical (indirect) interaction. Indirect guidance has been realized by using a six-dimensional (6-D) marker which is tracked by a visual tracking system [7], or by using gestures and posture recognition through an accelerometer-based input device [8]. Similarly, the Leap Motion sensor was used to develop a contactless and markerless human-robot interface to control dual-arms with the hands [9]. Offline programming in virtual reality, online programming in augmented reality and a combination of both is considered in [10] and further discussed in [7], where guidance with collision avoidance and end-effector restrictions is proposed. The major drawback of indirect guidance methods is that the user is physically separated from the robot. In the context of safety, this is advantageous, however, it impacts the intuitiveness of the teaching process. Direct manual guidance is often provided by robot specific teach pendants or can be realized by using force/torque sensors [11], [12] or inherently by low system inertia and high joint compliance [3]. Tactile sensors have also been used for manual guidance applications [13]. These approaches are based on sensors located either in the joints or scattered in some parts of the robot, or they rely on current measuring sensors, which require complete dynamic models to estimate the applied force. However, when dealing with physical interactions (with humans or the environment), the location, the direction, and the areas of contact are extremely important. Unfortunately, force/torque sensors can not deal with multiple contact points (they can only estimate the resultant force/torque applied to a single point). In some situations, this could lead to unsafe conditions, since the real pressure that the robot applies to a surface can not be determined. Manual guidance enables the untrained personnel to easily show the robot which paths (trajectories) it has to follow. The demonstrated trajectories and end-effector positions can then easily be assembled to the desired task. However, the user needs to define trajectories in the coordinate space, which leads to the classical *frame of reference problem.*[2] Furthermore, th e lack of precision and adaptability, when representing tasks with trajectories and positions, limits the usability and flexibility of the system [4].

[2]Thinking in the coordinate space is less natural than in the object space and needs expert knowledge.

PbD systems learn new skills by extracting redundancies across multiple demonstrations of the same movement and build time-independent models to reproduce the dynamics of the demonstrated motion [14]. An extension to learn also force profiles in combination with position profiles is introduced in [15]. The PbD system introduced in [6] builds generalized representations of dynamic motion primitives (DMP). The system separates demonstrated motions into a sequence of DMPs and maps them to predefined motion primitives (grasp, move, etc.) thus finding a symbolic representation of the demonstrated motion. The work of [16] derives tasks specified by parameters, where the parameters are invariant across demonstration. A change in these parameters defines a task transition, thus demonstrations can be segmented to subtasks with specific constraints (force, position, etc.).

The intuitiveness and naturalness of robot task programming can be increased by shifting coordinate-based programming (positions and trajectories) to object-based programming [2], [17], [18]. Object-based programming assumes that a general task can be subdivided into skills which are object-centered. Object-centered skills are configured with the parameters that make reference to objects instead of coordinates, e.g., pick, rotate, place, etc. The abstraction of object-centered skills hides low-level implementations which are system specific. Thus, the tasks composed of these skills are more precise and can be transferred and reused on different robot platforms. For example, the "little helper" is an autonomous industrial mobile manipulator [19], which implements task-level programming and enables users to compose tasks by manually selecting skills in a GUI. The skill parameters are obtained through kinesthetic teaching. *Cognitive robots* [17] increase the flexibility and usability of robots in manufacturing, validated with the example of a knitting task in the project STAMINA. *Cognitive automation* considers automatic and flexible decision making in complex environments with an intelligent adaptation of skills [20].

However, flexibility is not sufficient for real world applications since reusability of knowledge is required to handle multiple unmodeled conditions. Reusing the knowledge that has already been acquired can help to realize fast reconfiguration in manufacturing processes. The work of [21] introduces a knowledge integration framework for combining different knowledge representations in robotics. In [22], a method to generate plan descriptions for the automation of manufacturing processes is proposed. This paper uses a knowledge base in combination with ontologies to infer knowledge through reasoning for a given process specification. However, this approach was not tested in a physical system and the reusability of the obtained plans is limited to certain initial conditions.

Safety plays an important role in successfully deploying industrial robots. Safety requirements specified by the Organization for Standardization, such as ISO 10218-1/2 [23], [24], and IEC 61508 [25], have to be fulfilled. A new specification (ISO 15066) [26] specifically addresses safety requirements for physical interactions with robots in fence-less workspaces. Robot system solutions can follow different principles for enabling safe human-robot interactions in shared workspaces. Different scenarios for collaborative operations and their implications for safety are discussed in [27]. The robot introduced in [3] is

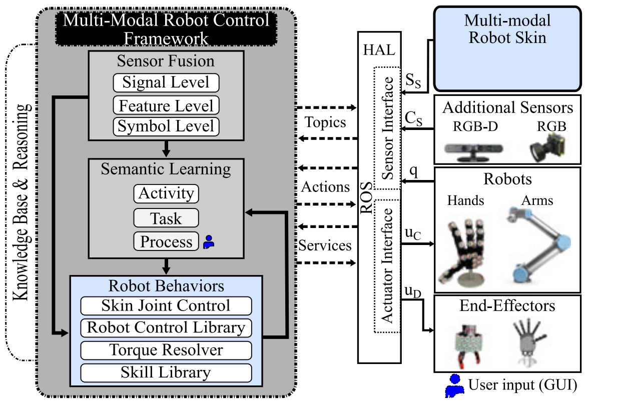


Fig. 2. Overview of the robot framework integrating the technologies: Multimodal robot skin, robot behavior generator, and semantic reasoning engine.

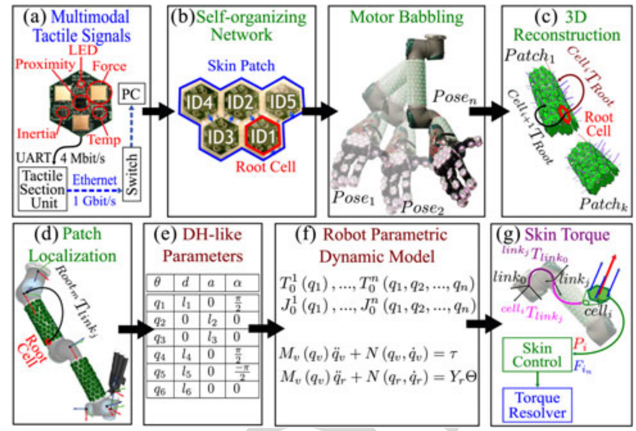


Fig. 3. Block diagram of our *end-to-end* self-configuring and self-calibrating *robot skin* approach.

inherently safe because of its low inertia and high passive compliance. On the other hand, the Kuka LWR minimizes injury risks by lowering the load-to-weight ratio, enabling fast reactions to collisions, and providing active compliance through force/torque sensing [28]. The ROSETTA project introduces a flexible, collaborative robot for the automatic assembly of small parts [29]. Safe interaction in shared workspaces is enhanced through low payload and inertia, a mechanical design without sharp or pointed edges, cushioning (passive compliance), power and speed limitations, and software based collision detection. However, these principles add constraints to the robot design, making them more expensive, and require a complete redesign of industrial robots.

Q1

B. Main Contributions

In this paper, we extend our previous work [30] to improve our proposed method, which integrates three robotic technologies to allow the fast deployment of industrial robot systems, namely the *Multimodal robot skin*, the *Robot behavior generator*, and the *Semantic reasoning engine*, see Fig. 2. More concretely, our contributions are:

1) An overview on how the proposed technologies can be integrated in an *end-to-end* framework;
2) A multimodal control approach, providing fast reactions to reliably detected *contacts,* and *precontacts* to improve *pHRI*;
3) The enhancement of our semantic reasoning method to kinesthetically teach new activities to robots without the need of an expert robot-programmer;
4) The demonstration of the flexibility and reusability of the framework in different situations, such as the adaptation of the learned processes to new objects, and their transferability to different robots (with different frame of reference), without human intervention.
5) The extension of the reasoning method to detect and handle errors at execution time.

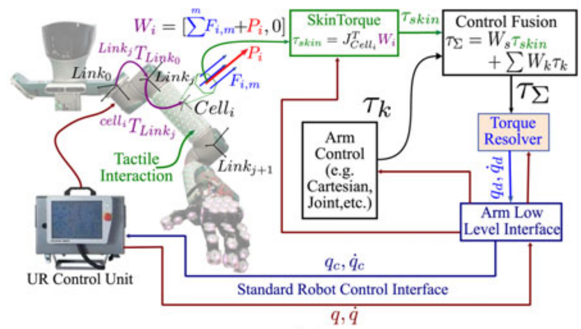
II. Multimodal Robot Skin

Fast, configurable multimodal *robot skin* can transform a standard industrial robot arm into a reactive robot system, enabling pHRI. A robot covered with *robot skin* can detect *contacts* and *precontacts* with high confidence and without occlusion. In combination with appropriate reactive low-level controllers, *robot skin* enables robots to actively mitigate or avoid potentially dangerous situations generated by unexpected changes in the environment, see Section III.

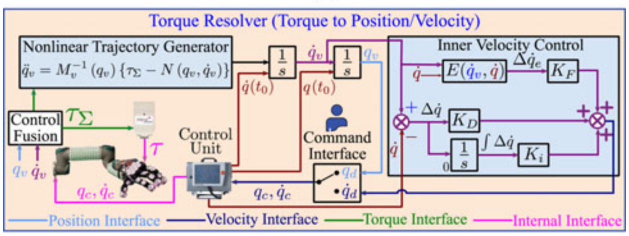
Our *robot skin* [31] is composed of modularized, hexagonally shaped skin cells [see Fig. 3(a)]. Each of these cells utilizes the same set of sensors, which transduce tactile information of different modalities, such as *vibrations* (3-D acceleration sensor), *pressure* (three capacitive force sensors), *pretouch* (optical proximity sensor), and *temperature* (two temperature sensors). The skin cells communicate and exchange information with their neighbors, and build up a self-organized and redundant skin cell network, which enables bidirectional communication with a central processing system, see Fig. 3(a) and (b). The *robot skin* uses the standard Gigabit Ethernet protocol, thus specific drivers are not required, making the *robot skin* easy to deploy. A group of connected skin cells forms a *skin patch*, and each patch has a *root cell*, see Fig. 3(b). These patches are used to cover the robot limbs, see Fig. 3(c) and (d). The *root cell* of a patch is used as a common reference frame for all the skin cells in a patch to define their spatial location on a robot link. This spatial information (homogeneous transformations) is essential to map tactile information to meaningful control commands, and is obtained through a robot skin calibration process.

A. Robot Skin Calibration

Performing manual skin calibration of hundreds of skin cells is prone to errors and totally infeasible. We tackle this challenge by developing a complete *end-to-end robot skin* system, see Fig. 3. The first stage of the system is to explore the cell network using a self-organizing network algorithm, resulting in optimal communication paths and neighbor information of the skin cells [see Fig. 3(b)]. Next, we use motor babbling, a 3-D surface reconstruction algorithm, and an extrinsic calibration algorithm to obtain both, the relative poses of the skin cells with respect to the *root cell*, see Fig. 3(c), and the *root cell* with respect to the *robot link*, Fig. 3(d). From these homogeneous transformations, we obtain a set of Denavit–Hartenberg-like parameters, Fig. 3(e),



(a)



(b)

Fig. 4. (a) General control pipeline to fuse *robot skin* signals in multiple controllers. $F_{i,m}$ and P_i are the forces and proximity signals of a $Cell_i$. $W_i \in \mathbb{R}^{6\times 1}$ is a virtual tactile wrench, where $J_{Cell_i} \in \mathbb{R}^{6\times n}$, with n as the robot's DoF, represents the Jacobian of each cell [32]. W_s and W_k are weight matrices to control the influence of each low-level control and depend on the specific *Robot Behavior*. (b) Torque resolver defined by two principal modules: first, the nonlinear trajectory generator that produces desired trajectories based on user-defined dynamic behaviors, and second, the inner velocity control that generates a desired joint velocity to compensate uncertainties in the robot parameters. $q_v, \dot{q}_v$ represent the joint position/velocity of the virtual robot (desired position/velocity), $q, \dot{q}$ are the joint position/velocity of the real robot, $q_c, \dot{q}_c$ are the commanded joint position/velocities. $E(\dot{q}_v, \dot{q})$ is a joint velocity estimator.

which can be used to generate kinematic models [forward kinematics and robot Jacobians, Fig. 3(f)]. For further details about this calibration process, see [32]. These kinematic models are used to obtain the dynamic model of the robot. This model is defined in a parametric algebraic form, and its dynamic parameters are defined by the designer, see Fig. 3(f). In this manner, the designer can define how the robot should react to the *robot skin* information, i.e., the *desired dynamic behavior of the robot*. Notice that the dynamic parameters do not need to be exact, but they should produce a suitable desired behavior that the real robot is able to generate. A close approximation of these parameters can be obtained using the *Robot Regressor* technique [33]. These models are exploited by our Robot control framework (see Fig. 4(b) in Section III).

B. Event-Driven Robot Skin

The deployment of large-scale *robot skin*[3] introduces new challenges [34]. To tackle these challenges, we investigate and apply biologically inspired principles. The pivotal principle that we use is the novelty driven tactile information transduction, transmission, and processing, i.e., an *event-driven* system. In contrast to synchronous sensors, which continuously transmit information, the sensors of an *event-driven* system only transmit information when there is an event. Events represent *novel information* and are usually triggered by sufficiently large changes in the sensed information. The skin cells of our *event-driven robot skin* asynchronously transmit information in form of *complete sensor values*, this means that this information is not based on differences in the form of deltas [34]. As a result, *event-driven robot skin* improves the performance of real-time control, and allows fast controller responses. For example, this *event-driven* system reduced the network usage of a skin network with 253 skin cells from 1 MB/s to 81 kB/s, and the CPU usage of the controller from around 104% to 42% [35].

[3]We estimate that a completely covered humanoid robot will need at least 3000 skin cells.

Robot Behaviors	Low-Level Controllers: Joint Control	Spline Joint Control	Cartesian Control	Spline Cartesian Control	Skin Joint Control	Skin Cartesian Control	G Control
Reach Joint Compliant	✓				✓		✓
Reach Joint Goal Compliant		✓			✓		✓
Reach Cartesian Compliant			✓		✓		✓
Reach Cartesian Goal Compliant				✓		✓	✓
Kinesthetic Joint					✓		✓
Kinesthetic Cartesian						✓	✓

Fig. 5. Figure shows the arm behaviors composed of low-level controllers running in parallel. The gripper behavior is a state machine (open/close).

III. Robot Behaviors

The *Robot Behavior* module provides a library of low-level controllers that can be combined to produce different robot behaviors. This library contains control approaches with low-level control methodologies that produce continuous signals for the robot arms, the hands, and the grippers. Examples of these controllers are depicted in Fig. 5. The *Skin Joint controller* is of particular interest since it transforms the multimodal signals obtained from the *robot skin* into a coherent control signal that can be fused with the other low-level controllers, see Fig. 4(a). The generation of robot behaviors has three steps. The first step is to transform the tactile signals (force and proximity) of the *robot skin* into torque signals *Skin Torque* (τ_{skin}). This is achieved by representing these *robot skin* signals as forces.[4] This is possible since we know the spatial information (pose) of each skin cell, see Fig. 3(c) and (d) in Section II-A. The second step is to fuse the *Skin Torque* signals with the other controllers to obtain different behaviors. In this case, we used a simple weighted sum approach. The third step is to transform this fused control signal into an appropriate command signal. This command depends on the target robot.[5] If the robot uses *torque* commands, the commanded signal will be the fused torque signal. However, if the robot is controlled using *joint positions/velocities*, we use a *Torque Resolver*, see Fig. 4(b). This resolver uses the kinematic and dynamic models obtained in the self-calibration process, see Fig. 3(e)–(g) in Section II-A [32].

[4]We selected force (wrench) as the representation of the signals due to its convenient relation with joint torques $\tau = J^T F$.

[5]In general, we can find three types of command interfaces for robots: joint position, joint velocity, and joint torque commands. The first two are the most common interfaces for industrial robots.

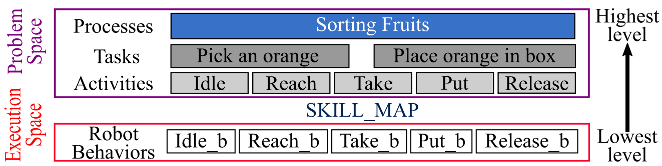


Fig. 6. Hierarchical structure of our reasoning system. The Problem Space (purple box) provides semantic descriptions which represent robot-agnostic knowledge. The Execution Space (red box) is the information needed to execute robot motions. This information is robot-dependent.

The fusion of the different low-level controllers produces various behaviors. These combinations are defined in the *Robot Behavior* module, e.g., the *Reach Cartesian Goal Compliant* behavior fuses the *Cartesian*, the *Skin Torque*, and the *Gravity compensation* controllers, see Fig. 5. The previously discussed controller allows the robot to follow a Cartesian trajectory with the end-effector while at the same time can react to the tactile stimuli. This *Robot Behavior* module is the interface between the *Semantic Reasoning Engine* module, see Section IV, and the robot's low-level controllers.

IV. Semantic Reasoning Engine

We developed a semantic reasoning engine that allows nonexpert users to teach robots new tasks simply by guiding the robot's end-effector. To this aim, we use a hierarchical learning approach that is able to extract and interpret *low-level* features from the robot's end-effector as well as from its sensors to automatically generate compact semantic rules. The reasoning engine uses the obtained semantic rules to infer the robot activities as well as tasks from human Kinesthetic demonstrations (*high-level*). Fig. 6 exemplifies the transition between the lowest level to the highest level used in our system. The lowest level represents robot behaviors, which are defined in the *execution space*. These behaviors represent the primitives that the robots can execute. The robot motions are automatically interpreted by our reasoning system as *activities*, e.g., "Reach" and "Take." Our reasoning system is also able to combine a set of different activities into a *task* (see Fig. 6). Finally, the user defines a new *process* using the tasks provided by our system. *Processes*, *Tasks*, and *Activities* are described in the *Problem Space*, and they are robot agnostic descriptions.

In order to automatically interpret the kinesthetic demonstrations, our learning system transforms the continuous signals obtained from the demonstrations to symbolic representations [36]. For example, the motions (m) of the robot's end-effector (ef) are interpreted as either *Move* or *Not Move* symbols. Where *Move*: the end-effector is moving, i.e., $\dot{x} > \varepsilon$ and *Not Move*: the end-effector stops its motion, i.e., $\dot{x} \to 0$, where $\dot{x}$ is the end-effector velocity and ε is a heuristically defined threshold. In addition, the information about the perceived environment is also transformed into symbolic representations. For the demonstration scenario described in Section V, the robot tactile omnidirectional mobile manipulator (TOMM) can perceive its environment through the following sensors: *robot skin*, RGB-D camera, and joint sensors. From these sensors the following abstract properties can be defined:

1) *ObjectActedOn*[6] (o_a): The end-effector is moving towards an object, $d(x_{ef}, x_{o_i}) \to 0$;
2) *ObjectInHand* (o_h): The object is in the end-effector, i.e., $d(x_{ef}, x_{o_i}) \approx 0$, where $d(\cdot,\cdot)$ is the Euclidean distance between the end-effector (x_{ef}) and the detected object (x_{o_i});
3) *GripperState* (g_s): The current state of the gripper (open/closed).

After transforming the perceived environment and the robot motions into symbolic representations, $s_s = \{m, o_a, o_h, g_s\}$, we train a decision tree (T) using only one kinesthetic demonstration. We follow a similar pipeline as the one presented in [36], where the C4.5 algorithm is employed to compute T. This tree contains semantic descriptions of the robot motions represented by *if-then rules*, which are human readable:

$$\begin{aligned} &if\ ef(Move)\ \&\ ObjActOn(Fruit)\ \&\ Gripper(Open) \\ &\quad \to Act(\mathbf{Reach}) \end{aligned} \tag{1}$$

The rules obtained from T are enhanced with our knowledge and reasoning engine. The knowledge base is defined by an ontology representation, expressed in the Web Ontology Language. The reasoning is based on description logics, such as Prolog queries. Note that (1) contains the class "Fruit" rather than the trained object "Orange" which means that the extracted semantic representation for the *activity* "Reach" can be reused for all the objects that belong to the general class "Fruit". Consequently, the more general we create the semantic representations, the more demonstrations the robot can interpret without the need to retrain, thus making the reasoning system reusable in different situations [37], [38]. When a new activity is inferred by the reasoning system, a *Robot Behavior* is associated with this activity and stored in an abstract *Robot Behavior* representation (*Skill Map*), see Fig. 6. The *Skill Map* contains the necessary parameters to execute the demonstrated activity by the robot, thus allowing the transition between the *execution space* and the *problem space*. For example, when the reasoning system infers the activity "PutSomethingSomewhere," the parameters generated in the *skill map* are *something*, *somewhere,* and *Robot Behavior*, where *something* is instantiated when a new object is detected (orange), *somewhere* identifies the final position of the activity (box) and the executed *Robot Behavior* is *Reach Cartesian*. Then, a directed *task graph* is obtained where vertices represent the inferred activities and edges represent transitions between activities. The *pre-* and *post-* conditions of each activity are also obtained [39]. Furthermore, the reasoning system can provide the sequence of activities that compose a task. The user can also define a new *process* by selecting the desired tasks and a stopping criterion,[7] without the specification of additional parameters.

[6]The information from the object can be obtained either from the vision system or the proximity sensor of the skin. The same is valid for the property *ObjectInHand*.

[7]The stop criterion indicates when a process should stop, e.g., duration, weight, or number of objects.

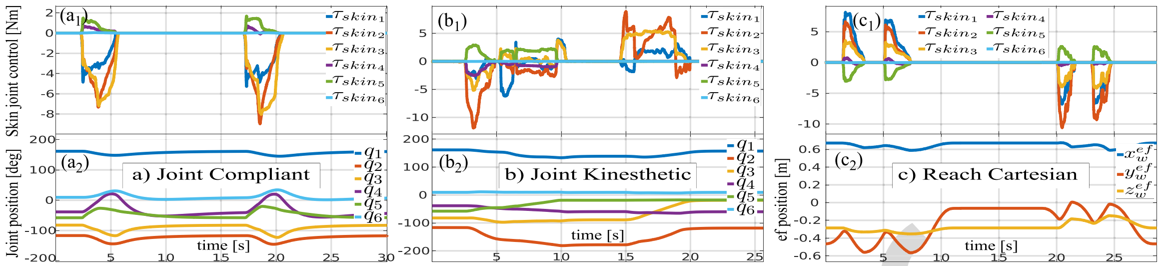
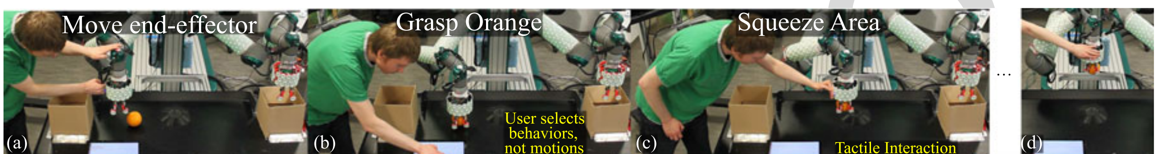


Fig. 7. Different robot behaviors obtained with the Multi-modal Robot Control Framework. The Skin joint control uses the *robot skin* to generate the skin torque. This torque reflects tactile interactions with the robot. (a) The robot is transformed into a compliant system. Each time there is a physical interaction, reflected in the skin joint control torque (a_1), the joint position changes (a_2). (b) For the kinesthetic teaching mode, the robot changes its position (b_2) when the user physically interacts with it (b_1). In this case, the robot does not return to its original position after the interaction. (c) When a user interferes with the robot's motion (c_1), the robot reacts and smoothly changes its trajectory (c_2). As soon as the obstacle is no longer present, the robot restarts the task until it reaches the goal (c_2).



Input (User)	Output (Sem.Eng.)	Input (User)	Output (Sem.Eng.)	Input (User)	Output (Sem.Eng.)	Input (U)	Output (SE)
Resource: Right Arm		Resource: Right Gripper		Resource: Right Arm		…	…
Behavior: Select Kinesthetic Cartesian	Skill Map: Reach Cartesian Behavior	Behavior: Select Close Gripper	Skill Map: Close Gripper Behavior	Behavior: Select Kinesthetic Cartesian	Skill Map: Reach Cartesian Behavior	…	…
Demonstration: Move end-effector to orange	Inferred Activity: Reach	Demonstration: Grasp the orange with gripper	Inferred Activity: Take	Demonstration: Move end effector to squeeze area	Inferred Activity: Put something somewhere	…	…
Task 1: Pick Fruit				Task 2: Identify Good Fruit			
Process: Sorting Fruits							

Fig. 8. Semantic reasoning engine interprets user demonstrations. The input of the reasoning engine (white boxes) comes from the human, i.e., he/she selects the *Robot Behavior* and demonstrates the activities. The output (gray boxes) is defined with two abstraction levels. The first level, (light gray) represents the inferred activities and their associated *Robot Behavior*. The second level (dark gray) represents the tasks generated from the inferred activities, using the sequence followed by the user during the demonstration. The user can build different processes as needed (lower dark gray box).

V. Demonstration Scenarios & Validation

A. Robot Platform & Evaluation of Control Behaviors

To evaluate our multilevel control framework, our tactile omni-directional mobile manipulator with more than 600 skin cells was used [40]. These skin cells cover the arms and the grippers. The grippers have three different skin patches: hand patches to detect obstacles, external finger patches to detect the texture of the fruits (stiff or soft), and internal finger patches to detect when the grippers are grasping an object. TOMM has three-fingered grippers which are inherently compliant and suitable to handle fruits without damaging them. We consider the following three different dynamic behaviors:

1) *Joint Compliant* behavior,
2) *Kinesthetic Joint* behavior, and
3) *Reach Cartesian Goal* behavior.

In the *Joint Compliant* behavior [see Fig. 7(a)], the robot reacts to tactile events and changes its position, e.g., when the user applies forces to the arm. When there are no external perturbations, the robot *returns to its original position* smoothly. The *Kinesthetic Joint* behavior [see Fig. 7(b)] is similar to the Joint Compliant behavior, but instead of returning to its original position, the robot will stop and *will remain in this position* as long as no further tactile events are detected. In the *Reach Cartesian Goal* behavior [see Fig. 7(c)], the goal position of the robot's end-effector is defined by the user. Then, a trajectory to reach this goal is computed using a spline function. The robot arm follows this trajectory and when the user interferes with the robot (detected by the skin sensors), this event produces a compliant reactive behavior which forces a *change in the robot trajectory*.

B. Sorting Fruits Scenario

As a demonstration scenario, we consider the task of sorting fruits. With this scenario, we can highlight the benefits of using the tactile and proximity sensors on the *robot skin* to sense the quality of the fruits.[8] The user teaches the robot the activities and the intermediate tasks required to sort oranges: 1) Good oranges (with stiff texture) will be placed in a box, and 2) Bad oranges (with soft texture) will be thrown into the trash bin, see Fig. 8. The texture of the oranges is evaluated using the force sensors from the *robot skin* placed in the external finger patches of the grippers. The stiffness threshold to discriminate the texture of the fruits is defined during the demonstration. Our approach consists of two phases: *Teaching* and *Execution*.

C. Kinesthetic Teaching With Semantic Inference

In our previous work [41], we obtained semantic models of human activities for "making a pancake" using the iCub robot. From these models, we obtained common descriptions such as "Reach," "Take," etc., along with common tasks such as "Picking an object." These semantic descriptions were reused and extended to teach a Humanoid robot H1 (REEM-C) how to "make

[8]This scenario was inspired by the standard process of orange sorting where humans use their tactile sensation to discriminate good and bad oranges.

Perceived Object	Task 0 Reset		Task 1 Pick Fruit		Task 2 Identify Good Fruit		Task 3 Put Fruit into Box		Task 4 Identify Bad Fruit		Task 5 Put Fruit into Bin	
Orange	Pre-cond	Post-cond	Pre-cond	Post-cond	Pre-cond	Post-cond	Pre-cond	Post-cond	Pre-cond	Post-cond	Pre-cond	Post-cond
	OIH=none Hand=any	OIH=none Hand=open	OIH=none Hand=open	OIH=orange Hand=close	OIH=orange Hand=close Squeeze=0.0	OIH=orange Hand=close Squeeze ≥0.3 Dest=s_a	OIH=orange Hand=close Dest=s_a	OIH=none Hand=open Arm=any Dest=box	OIH=orange Hand=close Squeeze=0.0	OIH=orange Hand=close Squeeze<0.3 Dest=s_a	OIH=orange Hand=close Dest=s_a	OIH=none Hand=open Arm=any Dest=bin
Apple	Pre-cond	Post-cond	Pre-cond	Post-cond	Pre-cond	Post-cond	Pre-cond	Post-cond	Pre-cond	Post-cond	Pre-cond	Post-cond
	OIH=none Hand=any	OIH=none Hand=open	OIH=none Hand=open	OIH=apple Hand=close	OIH=apple Hand=close Squeeze=null	OIH=apple Hand=close Squeeze ≥0.3 Dest=s_a	OIH=apple Hand=close Dest=s_a	OIH=none Hand=open Arm=any Dest=box	OIH=apple Hand=close Squeeze=null	OIH=apple Hand=close Squeeze<0.3 Dest=s_a	OIH=apple Hand=close Dest=s_a	OIH=none Hand=open Arm=any Dest=bin

Stop Criteria

Fig. 9. Learned process of "Sorting Fruits". The semantic system verifies the pre-conditions of each task before starting its execution. The first row depicts this process when the perception system detected *oranges*. The second row shows the automatic adaptation of the same process for *apples*. Task 2 and Task 3 have been rejected since the class "Apple" does not have the *Squeeze* property. When there is an anomaly, for example, dropping an orange, the system can detect the error and the following tasks are rejected.

a sandwich" [42]. In this case, the semantic descriptions were extended to include new activities, such as "Cut" and new tasks, such as "Cutting an object". These semantic descriptions are the initial knowledge-base for our robot TOMM (*robot experience*) to learn how to "sort fruits". This is possible since the semantic models are robot-agnostic and defined in the *Problem Space*, see Fig. 6. During the *teaching* phase, our reasoning system is extended to generate and populate new semantic descriptions for the "sorting fruit" scenario through human demonstrations. For these demonstrations, the user kinesthetically guides the robot by selecting a specific *Robot Behavior*, see Fig. 5.

Fig. 8 depicts the pipeline of the reasoning engine which uses human demonstrations as input (low-level sensor information) and produces the inferred activities as output (high-level interpretations). These activities will be connected by the reasoning engine to generate the demonstrated tasks, following the sequence taught by the user. Each inferred activity will be associated with a specific *Robot Behavior* through the *Skill Map*. For example, in Fig. 8(a), the user selects the right arm as the desired resource for teaching. Then, the user selects the *Kinesthetic Cartesian* behavior to move the robot's end-effector towards the orange. This demonstration is inferred by the semantic engine, as the activity "Reach", and the *Skill Map* connects this activity with the robot behavior *Reach Cartesian*. Similarly, Fig. 8(b) depicts the user selecting the gripper as the desired resource and *Close Gripper* as its behavior. This new demonstration is interpreted by the semantic engine, as the activity "Take". Then, these two activities ("Reach" and "Take") are automatically connected and defined as the task "Pick Fruit" by the reasoning system. As part of the task definition, the reasoning system defines the *pre-* and *post-* conditions required for each task, see Fig. 9. All these generated tasks will be stored in the knowledge base and can be retrieved by the user to define new processes. The processes and the tasks are specified using abstract representations (semantic level), which make the system highly flexible. Note that the reasoning system abstracts the meaning of the demonstrated tasks, instead of storing motion patterns, or low-level information, such as velocities, trajectory patterns, end-effector positions, etc. For example, instead of saving the trajectory from the box to the orange while moving the arm, the learning system defines this activity as "Reaching".

Fig. 9 depicts the automatically generated tasks, for the process "Sorting Fruits". In this case, five tasks were generated: 1) *Pick fruit*, 2) *Identify good fruit*, 3) *Put fruit into a box*, 4) *Identify bad fruit*, and 5) *Put fruit to trash*.[9] Our reasoning method also provides the option to create tasks by manually selecting and connecting the available activities from the acquired knowledge base (using a GUI).

The main feature of this learning system is that a nonexpert user can teach the robot new tasks. Since the user does not need to program the robot directly (using a teach-pendant and a specific robot language), but rather the user guides the robot and the system generates the proper sequences in a human readable form. In this context, the robot's program is replaced by a sequence of tasks where their parameters are defined at runtime.

We tested the integration of the presented three technologies, using the demonstrations from two different participants.[10] Four demonstrations were considered with random positions for the oranges [43]. The robot is able to recognize the demonstrated activities using our reasoning technology with an average accuracy of around 83%[11] when the participants kinesthetically showed the desired activities for the *sorting fruits* process. The sequence of recognized activities is consecutively stored to automatically define task structures to accelerate the learning of new tasks [39]. Ongoing research on a formal validation, including multiple participants with different backgrounds, is currently being pursued.

To teach a new process to a robot, a normal user takes approximately two minutes and 44 s. Two minutes and 6 s to kinesthetically teach the activities and create the tasks, and 38 s to build a new process and launch it [39]. These times depend on the complexity of the new process, in this case, the process consists of five tasks. After that, no more user intervention is required, even when new objects appear in the scene.

D. Execution of the Demonstrated Tasks

First, the user indicates the process to be executed, see Fig. 10. Then, the framework verifies if the process can be executed in the current environment. For this, the semantic engine loads the tasks of the selected process, see Fig. 9. Prior to executing a task, the system verifies if all the task preconditions are satisfied. If a precondition is not satisfied, the task fails (we exploit

[9]The system connects the activities to compose the tasks, and the user defines the labels for these tasks.

[10]One participant was a robotic expert and the other nonexpert.

[11]This accuracy is obtained by comparing the recognized activities between the reasoning system and the ground truth (manual annotations).

Process: Sorting Fruits									
Task 1: Pick Fruit				Task 2: Identify Good Fruit					
	Res: Both Arms		Res: Both Grippers		Res: Both Arms	…	…		
Skill: Reach Cartesian Behavior	Skill: Cart. Control Spline(origin,goal)	Skill: Close Gripper Behavior	Skill: Gripper Control (state)	Skill: Reach Cartesian Behavior	Skill: Cart. Control Spline(origin,goal)	…	…		
Inferred Activity: Reach	Par: Fruit=**orange,** origin=E.E. goal=3D **orange**	Inferred Activity: Take	Parameter: state=**close**	Inferred Activity: Put something somewhere	Parameter: somewhere = **squeeze_area (s_a).** ori=E.E. goal=3D **s_a**	…	…		
Input (Sem.Eng.)	Output (Robot)	Input (Sem.Eng.)	Output (Robot)	Input (Sem.Eng.)	Output (Robot)	In(SE)	Out (R)		

Fig. 10. Execution of the learned process. The semantic engine analyzes all the tasks and their activities of the process and executes the associated robot behaviors. The input of the execution phase is the semantic process, and the output is the execution of the robot behaviors with the required parameters.

this verification process to detect errors, see Section V-F). Then, for each task, the semantic engine infers the sequence of activities at runtime. Each activity has an associated *Robot Behavior*, which requires specific parameters for its execution. These parameters are obtained at runtime, using the perception system, which identifies and labels the objects in the scene. The semantic engine uses these labels to create instances of the proper class of the identified object. The instances are single data containers (semantic abstractions) that provide multiple information from the perceived object, e.g., parent class, type, color, size, position, orientation, squeeze ratio, etc. For the resources (arms and grippers) we use the following information joint/end-effector positions, velocities. For example, the perception system identifies an orange and obtains its 3-D-position. Then, an instance of the class "Orange" is created and its property named "Position" is populated with the orange's position.

The semantic engine starts to execute each activity with its associated *Robot Behavior* and its targeted object. Each executed behavior triggers a set of low-level controllers, which requires different parameters. For example, in Fig. 10(a), the Task 1 "Pick Fruit" executes two sequential activities: "Reach" and "Take". The unbounded variable *Fruit* is instantiated with the perceived object (an instance of class "Orange").[12]

The activity "Reach" is associated with the *Reach Cartesian* behavior, see Fig. 8(a). This behavior executes in parallel three low-level controllers *Spline Cartesian*, *Skin Cartesian*, and *Gravity compensation*, see Fig. 5. The *Spline Cartesian* control requires two parameters the *origin* and the *goal*. The *origin* is set, by default, as the *current position of the end-effector*, and the *goal* is defined, by the semantic engine, as the *orange's position*. When the low-level controllers are finished, the next activity will be executed. For the activity "Squeeze," the robot places the end-effector over the fruit, and moves down slowly until the fruit is detected by the *robot skin* (soft contact). Then, the end-effector moves down again a few milliliters and measures the contact force. This force is correlated to the stiffness of the fruit. The rest of the tasks and their activities are executed until the process is finished. Note that during the teaching phase, the user demonstrated the process using only one arm,

[12]The perception system can detect different objects from different classes, e.g., oranges and apples (class "Fruit"), box and trash bin (class "Container"), etc. The semantic reasoning discriminates and uses these objects according to their class.

Process: Sorting Fruits . . .					
Task 1: Pick Fruit				Task n …	
	Res: Both Arms		Res:2 Grippers	…	…
Skill: Reach Cart. Behavior	Skill: Cart.Control Spline(origin,goal)	Skill: Close Gripper Behavior	Skill: Gripper Control (state)	…	…
Inferred Activity:Reach	Par: Fruit=**apple** origin=E.E goal =3D **apple**	Inferred Activity: Take	Parameter: state=**close**	…	…
Input(Sem.Eng)	Output (Robot)	Input (Sem.Eng.)	Output (Robot)	In(SE)	Out(R)

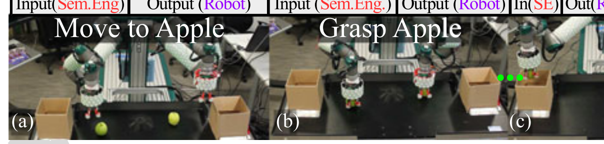


Fig. 11. Executing the inferred process for apples. The same process "Sorting Fruits" can be used without user intervention or reprogramming, even when the process was generated using *oranges*. The inferred execution is the robot grasping the apples and putting them into the boxes without squeezing them. This is a correct execution since the apples should not be squeezed.

i.e., the right arm. However, since the activities and their parameters are defined with abstract representations, the same process can be used with multiple resources (robots). This makes the descriptions general, transferable, and reusable.

The semantic system verifies the available resources of the robot for the execution of the desired task. In this case, the robot has the right and the left arms enabled, and there is more than one orange on the table, see Fig. 10. Therefore, exactly, the same process ("Sorting Fruits") is executed with both arms (the only difference is the resource assigned by the semantic engine).

E. Handling Variations in the Process: Apple

The obtained general descriptions of tasks and activities allow our generated semantic process to work also with objects different from the ones used during the teaching phase. In Fig. 11, the scenario presents a different object (apples). In this case, the variable *Fruit* is an instance of the class "Apple" and the corresponding semantic properties of this class are loaded. However, the class "Apple" does not have the property "squeezable". Therefore, Task 2 and Task 4 can not be executed, see Fig. 9 (Squeeze = null). In this case, only the tasks that can be executed for *apples* are Task 1, Task 3, and Task 5 (tasks that do not depend on "squeeze" property). These tasks are executed sequentially. As a result, the robot takes the apple and place it into the box. Our reasoning system only takes approximately 0.124 s to make this new execution plan compared to 38 s required to generate the orange sorting plan.

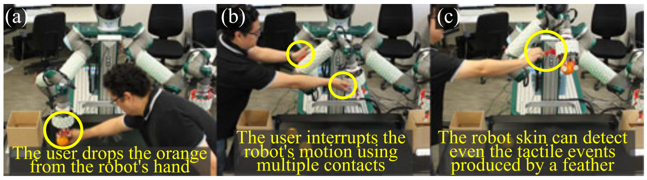


Fig. 12. Error detection and user perturbations. (a) Using the internal finger patches, the semantic reasoning detects this error and infers which task can be executed. (b) The robot safely reacts to the human interaction and avoids collisions. (c) Exploiting both the force and the proximity sensors, the *robot skin* can detect even a feather.

F. Error Detection

The structure of the process description allows the semantic engine to detect errors through the verification of the preconditions on each task. Fig. 12(a) shows the case when the user removes the orange from the robot's gripper at the end of Task 1. This anomaly is detected by the reasoning engine, i.e., *Error Detection*. As can be seen in Fig. 9, Task 2 requires as precondition that the gripper has an object in the hand (OIH = orange) since this precondition is not satisfied, then all the following tasks will fail. Hence, the sequence Task 0–Task 1 will be repeated until the gripper has an object in hand. In this case, the system also provides our first approach to *Error Handling*. The time that our system takes to detect this error and to search for a new strategy takes around 0.38 s.

G. Physical Human-Robot Interaction

One important aspect considered in our system is the pHRI, where safe interaction is paramount. In Fig. 5 can be seen that all the behaviors contain either *Skin Joint Control* or *Skin Cartesian Control*. These two controllers make the robot reactive to tactile events (*precollisions* and *pressure*) allowing physical interactions. Fig. 12(b) and (c) show two examples of these reactive interactions. Other safety mechanisms, we have adopted in our robot skin are redundancy of communication paths in the skin cell network, redundancy in skin sensor modality (e.g., contact detection through proximity and force sensors), and real-time user feedback through RGB LEDs of the skin cells.[13] In order to validate our rapidly deployable robot system, we successfully installed the *robot skin* on two different robot arms in two different laboratories. First, we fully covered a UR5 robot with 410 skin cells using 13 patches. In addition, we also covered the forelimb of a UR10 arm with 373 skin cells. The deployment from installing the skin patches in the robots to a fully calibrated and ready to use *robot skin* took in both cases about 5 h.

We provide a video[14] to illustrate the robot behaviors in our robot TOMM using the proposed approach, and to show the teaching and execution phases for the process "Sorting Fruits."

[13]The authors consider these mechanisms only as starting points for functional safety and standardization, and we only highlight the system's potential regarding safety. Nevertheless, the process for productization of these systems is still in a preliminary state.

[14]https://youtu.be/_X255OyzGs0

VI. Conclusion

The overview of the integration of three main robotic technologies was presented in this paper. These technologies enable fast deployment of industrial robot systems and consists of a fast self-configurable artificial skin, a multimodal control framework to extend the dynamic behaviors of standard robots, and a robust and intuitive teaching method based on semantic reasoning. The presented results demonstrate that these technologies enhance the *usability*, *flexibility*, and introduce our first approach to handle *safety* for industrial robots, especially when a nonexpert user teaches the robot new processes using pHRI. The *usability* is demonstrated with the following aspects:

1) Novel technologies to teach robots new tasks using pHRI;
2) The extension of a semantic reasoning engine to automatically infer activities and tasks from human demonstrations;
3) The generation of processes in human-readable form via semantic descriptions, and;
4) Error detection and handling during process execution without human intervention.

The *flexibility* is validated as follows: 1) *end-to-end* robot skin framework for fast deployment on different robots, and 2) a knowledge-base system that allows the re-usability and transferability of learned skills. The *safety* is realized through a reactive control framework based on multimodal *robot skin* avoiding dangerous collisions during HRI. The presented framework can be implemented in any standard industrial robot as long as it provides an external control interface.

References

[1] A. Jäger, C. Moll, O. Som, C. Zanker, S. Kinkel, and R. Lichtner, "Analysis of the impact of robotic systems on employment in the European Union. Final report," Publications Office of the European Union, Luxembourg, 2015.

[2] M. R. Pedersen *et al.*, "Robot skills for manufacturing: From concept to industrial deployment," *Robot. Comput.-Integr. Manuf.*, vol. 37, pp. 282–291, 2016.

[3] T. Lens, J. Kunz, O. Von Stryk, C. Trommer, and A. Karguth, "Biorob-arm: A quickly deployable and intrinsically safe, light-weight robot arm for service robotics applications," in *Proc. 6th German Conf. Robot.*, 2010. Q2

[4] R. D. Schraft and C. Meyer, "The need for an intuitive teaching method for small and medium enterprises," *37th Int. Symp. Robot., 4th German Conf. Robot.*, 2006, vol. 1956.

[5] A. D. Santis, B. Siciliano, A. D. Luca, and A. Bicchi, "An atlas of physical human–robot interaction," *Mechanism Mach. Theory*, vol. 43, no. 3, pp. 253–270, 2008.

[6] R. Dillmann *et al.*, "Advances in robot programming by demonstration," *KI-Künstliche Intelligenz*, vol. 24, no. 4, pp. 295–303, 2010.

[7] B. Hein, M. Hensel, and H. Wörn, "Intuitive and model-based on-line programming of industrial robots: A modular on-line programming environment," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2008, pp. 3952–3957.

[8] P. Neto, J. Norberto Pires, and A. Paulo Moreira, "High-level programming and control for industrial robotics: Using a hand-held accelerometer-based input device for gesture and posture recognition," *Ind. Robot, Int. J.*, vol. 37, no. 2, pp. 137–147, 2010.

[9] G. Du, P. Zhang, and X. Liu, "Markerless human-manipulator interface using leap motion with interval kalman filter and improved particle filter," *IEEE Trans. Ind. Informat.*, vol. 12, no. 2, pp. 694–704, Apr. 2016.

[10] Z. Pan, J. Polden, N. Larkin, S. Van Duin, and J. Norrish, "Recent progress on programming methods for industrial robots," *Robot. Comput.-Integr. Manuf.*, vol. 28, no. 2, pp. 87–94, 2012.

[11] G. Grunwald, G. Schreiber, A. Albu-Schäffer, and G. Hirzinger, "Programming by touch: The different way of human–robot interaction," *IEEE Trans. Ind. Electron.*, vol. 50, no. 4, pp. 659–666, Aug. 2003.

[12] D. Massa, M. Callegari, and C. Cristalli, "Manual guidance for industrial robot programming," *Ind. Robot*, vol. 42, no. 5, pp. 457–465, 2015.

[13] T. Wösch and W. Feiten, "Reactive motion control for human-robot tactile interaction," in *Proc. Int. Conf. Robot. Autom.*, 2002, vol. 4, pp. 3807–3812.

[14] S. Calinon, F. D'halluin, E. L. Sauser, D. G. Caldwell, and A. G. Billard, "A probabilistic approach based on dynamical systems to learn and reproduce gestures by imitation," *IEEE Robot. Autom. Mag.*, vol. 17, no. 2, pp. 44–54, 2010. Q3

[15] P. Kormushev, S. Calinon, and D. G. Caldwell, "Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input," *Adv. Robot.*, vol. 25, no. 5, pp. 581–603, 2011.

[16] A. L. P. Ureche, K. Umezawa, Y. Nakamura, and A. Billard, "Task parameterization using continuous constraints extracted from human demonstrations," *IEEE Trans. Robot.*, vol. 31, no. 6, pp. 1458–1471, Dec. 2015.

[17] V. Krueger *et al.*, "A vertical and cyber-physical integration of cognitive robots in manufacturing," *Proc. IEEE*, vol. 104, no. 5, pp. 1114–1127, May 2016.

[18] M. N. Nicolescu and M. J. Mataric, "Natural methods for robot task learning: Instructive demonstrations, generalization and practice," in *Proc. Second Int. Joint Conf. Auton. Agents Multiagent Syst.*, 2003, pp. 241–248.

[19] C. Schou, J. S. Damgaard, S. Bogh, and O. Madsen, "Human-robot interface for instructing industrial tasks using kinesthetic teaching," in *Proc. Int. Symp. Robot.*, 2013.

[20] D. Bruckner, H. Zeilinger, and D. Dietrich, "Cognitive automation—Survey of novel artificial general intelligence methods for the automation of human technical environments," *IEEE Trans. Ind. Informat.*, vol. 8, no. 2, pp. 206–215, May 2012.

[21] A. Björkelund, H. Bruyninckx, J. Malec, K. Nilsson, and P. Nugues, "Knowledge for intelligent industrial robots," in *Proc. AAAI Spring Symp., Des. Intell. Robots*, vol. 12, no. 2, 2012.

[22] J. Puttonen, A. Lobov, and J. L. M. Lastra, "Semantics-based composition of factory automation processes encapsulated by web services," *IEEE Trans. Ind. Informat.*, 2013, vol. 9, no. 4, pp. 2349–2359, Nov. 2013.

[23] *Robots and Robotic Devices – Safety Requirements for Industrial Robots—Part 1: Robots*, ISO 10218-1, 2010.

[24] *Robots and Robotic Devices – Safety Requirements for Industrial Robots–Part 2: Robot Systems and Integration*, ISO 10218-2, 2011.

[25] *Functional Safety of Electrical-Electronic-Programmable Electronic Safety-Related Systems. International electrotechnical commission*, IEC 61508, 1998.

[26] *Safety for Collaborative Industrial Robots*. Technical Standard ISO 15066, 2014.

[27] J. Fryman and B. Matthias, "Safety of industrial robots: From conventional to collaborative applications," in *Proc. 7th German Conf. Robot.*, 2012.

[28] A. Albu-Schäffer, S. Haddadin, C. Ott, A. Stemmer, T. Wimböck, and G. Hirzinger, "The DLR lightweight robot: design and control concepts for robots in human environments," *Ind. Robot, Int. J.*, vol. 34, no. 5, pp. 376–385, 2007.

[29] R. Patel, M. Hedelind, and P. Lozan-Villegas, "Enabling robots in small-part assembly lines: The "ROSETTA approach" - an industrial perspective," in *Proc. 7th German Conf. Robot.*, 2012.

[30] E. Dean-Leon, K. Ramirez-Amaro, F. Bergner, I. Dianov, P. Lanillos, and G. Cheng, "Robotic technologies for fast deployment of industrial robot systems," in *Proc. IECON 2016–42nd Annual Conf. IEEE Ind. Electron. Soc.*, Oct. 2016, pp. 6900–6907.

[31] P. Mittendorfer and G. Cheng, "Humanoid multimodal tactile-sensing modules.," *IEEE Trans. Robot.*, vol. 27, no. 3, pp. 401–410, Jun. 2011.

[32] E. Dean-Leon, F. Bergner, K. Ramirez-Amaro, and G. Cheng, "From multi-modal tactile signals to a compliant control," in *Proc. 2016 IEEE-RAS 16th Int. Conf. Humanoid Robots*, Nov. 2016, pp. 892–898.

[33] E. Dean-Leon, S. Nair, and A. Knoll, "User friendly Matlab-toolbox for symbolic robot dynamic modeling used for control design," in *Proc. 2012 IEEE Int. Conf., Robot. Biomimetics*, Dec. 2012, pp. 2181–2188.

[34] F. Bergner, E. Dean-Leon, and G. Cheng, "Event-based signaling for large-scale artificial robotic skin - Realization and performance evaluation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 4918–4924.

[35] F. Bergner, E. Dean-Leon, and G. Cheng, "Efficient event-driven reactive control for large scale robot skin," in *Proc. IEEE Int. Conf. Robot. Autom.*, to be published. Q4

[36] K. Ramirez-Amaro, M. Beetz, and G. Cheng, "Understanding the intention of human activities through semantic perception: Observation, understanding and execution on a humanoid robot." *Adv. Robot.*, vol. 29, no. 5, pp. 345–362, 2015.

[37] K. Ramirez-Amaro, M. Beetz, and G. Cheng, "Transferring skills to humanoid robots by extracting semantic representations from observations of human activities," *Artif. Intell.*, vol. 247, pp. 95–118, 2017.

[38] K. Ramirez-Amaro, H. N. Minhas, M. Zehetleitner, M. Beetz, and G. Cheng, "Added value of gaze-exploiting semantic representation to allow robots inferring human behaviors.," *ACM Trans. Interactive Intell. Syst.*, vol. 7, no. 1, pp. 5:1–5:30, 2017.

[39] I. Dianov, K. Ramirez-Amaro, P. Lanillos, E. Dean-Leon, F. Bergner, and G. Cheng, "Extracting general task structures to accelerate the learning of new tasks," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, 2016, pp. 802–807.

[40] E. Dean-Leon *et al.*, "TOMM: Tactile omnidirectional mobile manipulator," in *Proc. 2017 IEEE Int. Conf. Robot. Autom.,* May 2017, pp. 2441–2447.

[41] K. Ramirez-Amaro, M. Beetz, and G. Cheng, "Automatic segmentation and recognition of human activities from observation based on semantic reasoning," in *IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2014, pp. 5043–5048.

[42] K. Ramirez-Amaro, E. C. Dean-Leon, and G. Cheng, "Robust semantic representations for inferring human co-manipulation activities even with different demonstration styles." in *Proc. 15th Int. Conf. Humanoids Robots.*, 2015, pp. 1141–1146.

[43] K. Ramirez-Amaro, E. C. Dean-Leon, I. Dianov, F. Bergner, and G. Cheng, "General recognition models capable of integrating multiple sensors for different domains." in *Proc. 16th Int. Conf. Humanoids Robots*, 2016, pp. 306–311.

Emmanuel Dean-Leon (M'XX) received the M.Sc. and Ph.D. degrees in mechatronics from the Center for Research and Advanced Studies, Mexico City, Mexico, in 2003 and 2006, respectively.

He is currently a Senior Researcher at the Chair for Cognitive Systems, Technical University of Munich, Munich, Germany. His research interests include robotics, low-level control, and physical human-robot-interaction. Q5

Karinne Ramirez-Amaro (M'XX) received the M.Sc. degree in computer science at the Center for Computing Research, Mexico city, Mexico, in 2007 and 2015, respectively, and the Ph.D. degree (*summa cum laude*) in electrical engineering from the Technical University of Munich (TUM), Munich, Germany.

Since 2015, she has been a Postdoctoral Researcher at the Chair for Cognitive Systems, TUM. Her research interests include semantics representations, expert systems, human activity recognition, and programming by demonstration. Q6

Florian Bergner (M'XX) received the B.Sc. and M.Sc. degree in electrical engineering from the Technical University of Munich (TUM), Munich, Germany, in 2011 and 2014, respectively. Since 2014, he has been working toward the Ph.D. degree at the Chair for Cognitive Systems, TUM. Q7

His research interests include multimodal tactile sensing, large-scale robot skin, sensor networks, sensor fusion, neuromorphic engineering, and event-driven signaling and processing. Q8

Ilya Dianov (M'XX) received the M.S. and B.S. degree in computer science from Moscow Power Engineering University, Moscow, Russia, in 2011, and the M.Sc. degree in computer science from the Technical University of Munich (TUM), Munich, Germany, in 2015. He is currently working toward the Ph.D. degree at the Chair of Cognitive Systems, TUM. Q9

His research interest include semantic reasoning and task planning. Q10

Gordon Cheng (F'XX) received the Ph.D. degree in systems engineering from the Australian National University, Canberra, ACT, Australia, in 2001.

He is currently a Professor of the Chair for Cognitive Systems the Technical University of Munich, Munich, Germany. From 2002 to 2008, he was the Head of the Department of Humanoid Robotics and Computational Neuroscience, ATR Computational Neuroscience Laboratories, Kyoto, Japan. His research interests include humanoid robotics, cognitive systems, brain machine interfaces, biomimetic of human vision, human-robot interaction, active vision, and mobile robot navigation. Q11

Queries

Q1. Author: Please provide the expansion of acronyms “STAMINA” and “ROSETTA.”
Q2. Author: Please provide page range in Refs. [3], [4], [19], [21], [27], and [29].
Q3. Author: Please provide month in Ref. [14].
Q4. Author: Please update Ref. [35].
Q5. Author: Please provide the year in which Emmanuel Dean-Leon became “Member” of the IEEE.
Q6. Author: Please provide the year in which Karinne Ramirez-Amaro became “Member” of the IEEE.
Q7. Author: Please provide the subject in which Florian Bergner working toward the Ph.D. degree.
Q8. Author: Please provide the year in which Florian Bergner became “Member” of the IEEE.
Q9. Author: Please provide the subject in which Ilya Dianov working toward the Ph.D. degree.
Q10. Author: Please provide the year in which Ilya Dianov became “Member” of the IEEE.
Q11. Author: Please provide the year in which Gordon Cheng became “Fellow” of the IEEE.

Integration of Robotic Technologies for Rapidly Deployable Robots

Emmanuel Dean-Leon, *Member, IEEE*, Karinne Ramirez-Amaro, *Member, IEEE*, Florian Bergner, *Member, IEEE*, Ilya Dianov, *Member, IEEE*, and Gordon Cheng, *Fellow, IEEE*

***Abstract*—The automation of production lines in industrial scenarios implies solving different problems, such as the *flexibility* to deploy robotic solutions to different production lines, *usability* to allow nonrobotics expert users to teach robots different tasks, and *safety* to enable operators to physically interact with robots without the need of fences. In this paper, we present a system that integrates three novel technologies to address the above mentioned problems. We use an autocalibrated multimodal *robot skin*, a general *robot control framework* to generate dynamic behaviors fusing multiple sensor signals, and an intuitive and fast *teaching by demonstration* method based on semantic reasoning. We validate the proposed technologies with a wheeled humanoid robot in an industrial set-up. The benefits of our system are the transferability of the learned tasks to different robots, the reusability of the models when new objects are introduced in the production line, the capability of detecting and recovering from errors, and the reliable detection of *collisions* and *precollisions* to provide a fast reactive robot that improves the physical human-robot interaction.**

***Index Terms*—Multimodal control, physical human-robot interaction (pHRI), robot skin, semantic reasoning, teaching by demonstration.**

I. Introduction

THE demand for an increasingly high productivity level in industrial scenarios requires both, shorter task execution times and faster/easier robotic programming methods, which reduce the production costs. An automated process using robots needs to be programmed to perform as efficient as a human worker in various domains, for example in packing and quality checking of products. However, setting up a robotic system takes, in general, at least three months [1] implying the need of robot expert programmers with higher costs. These factors are more prominent for small and medium enterprises (SMEs) since they usually have small production batches and have to cope with more frequent changes in production processes. This problem is not only limited to SMEs but also affects major enterprises (MEs), which in general undergo a shift from *mass production* to *mass customization* increasing the overall need for more flexible production lines and fast effortless reconfigurations [2]. The successful automation of these production processes demands *flexible*, *usable*, and *safer* robotic solutions [3], [4]. *Flexibility* implicates that robotic systems have to be quickly deployable with short installation times, to be easy to move to different production sites and to allow quick and easy adjustments to current production needs. *Usability* implicates simple and intuitive programming methods, enabling nonexperts, and untrained personnel to effortlessly reconfigure the system in a natural way. *Safety* entails that systems incorporate new principles to provide the necessary safety[1] for human operators during physical interactions in shared workspaces. Combining all these requirements leads to *Robot Transparency*. Ideally, a robot is considered fully *transparent* when the deployment of the robot does not produce any changes (disruptions) in the production line. *Robot Transparency* can be measured by the effort needed to deploy the robot, such as safety mechanisms, personnel training, and changes in the production process. *Transparent Robots* allow human-robot collaborations, just as if they were human-human collaborations since the robot will have ideally the same set of skills and requirements as a human co-worker – in the context of a *specific* production process. The high adaptability and accuracy of human-robot collaborations facilitate the automation of industrial processes for both SMEs and MEs. Physical human-robot interaction (pHRI) [5] is a fundamental aspect of *Robot Transparency* as well as simple and intuitive teaching methods, for example, programming by demonstration techniques (PbD). This sort of teaching methods allow the operator to teach the robot tasks in an easy and natural way [6], hence, an expert robot programmer is not required, see Fig. 1. Therefore, the development and integration of technologies, such as *robot skin*, reactive control schemes, and robust teaching methods are needed to simplify the robot programming, to improve the physical interaction with robots, and to decrease the deployment time of robotic systems in shop floors, i.e., to increase the *Robot Transparency*.

A. Related Work

Programming robots can be done by manually guiding the robot to the desired position through physical (direct) or

Manuscript received May 30, 2017; revised August 23, 2017; accepted September 29, 2017. Paper no. TII-17-1139. *(Corresponding author: Emmanuel Dean-Leon).*

The authors are with the Institute for Cognitive Systems, Technische Universität München, Munich 80333 Germany (e-mail: dean@tum.de; karinne.ramirez@tum.de; florian.bergner@tum.de; dianoviv@gmail.com; gordon@tum.de).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2017.2766096

[1] More concretely, with *safety*, we mean avoiding dangerous collisions during a physical human-robot interaction.

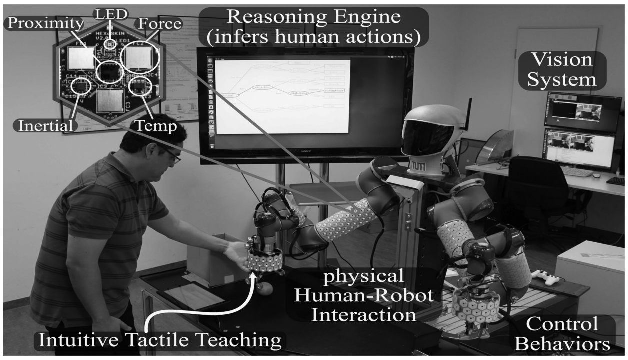


Fig. 1. Demonstration scenario: The user can intuitively teach a complete process to a robot. The setup consists of a perception system (fusing *robot skin* and vision), multimodal robot behaviors, and a reasoning system.

cyber-physical (indirect) interaction. Indirect guidance has been realized by using a six-dimensional (6-D) marker which is tracked by a visual tracking system [7], or by using gestures and posture recognition through an accelerometer-based input device [8]. Similarly, the Leap Motion sensor was used to develop a contactless and markerless human-robot interface to control dual-arms with the hands [9]. Offline programming in virtual reality, online programming in augmented reality and a combination of both is considered in [10] and further discussed in [7], where guidance with collision avoidance and end-effector restrictions is proposed. The major drawback of indirect guidance methods is that the user is physically separated from the robot. In the context of safety, this is advantageous, however, it impacts the intuitiveness of the teaching process. Direct manual guidance is often provided by robot specific teach pendants or can be realized by using force/torque sensors [11], [12] or inherently by low system inertia and high joint compliance [3]. Tactile sensors have also been used for manual guidance applications [13]. These approaches are based on sensors located either in the joints or scattered in some parts of the robot, or they rely on current measuring sensors, which require complete dynamic models to estimate the applied force. However, when dealing with physical interactions (with humans or the environment), the location, the direction, and the areas of contact are extremely important. Unfortunately, force/torque sensors can not deal with multiple contact points (they can only estimate the resultant force/torque applied to a single point). In some situations, this could lead to unsafe conditions, since the real pressure that the robot applies to a surface can not be determined. Manual guidance enables the untrained personnel to easily show the robot which paths (trajectories) it has to follow. The demonstrated trajectories and end-effector positions can then easily be assembled to the desired task. However, the user needs to define trajectories in the coordinate space, which leads to the classical *frame of reference problem.*[2] Furthermore, th e lack of precision and adaptability, when representing tasks with trajectories and positions, limits the usability and flexibility of the system [4].

[2]Thinking in the coordinate space is less natural than in the object space and needs expert knowledge.

PbD systems learn new skills by extracting redundancies across multiple demonstrations of the same movement and build time-independent models to reproduce the dynamics of the demonstrated motion [14]. An extension to learn also force profiles in combination with position profiles is introduced in [15]. The PbD system introduced in [6] builds generalized representations of dynamic motion primitives (DMP). The system separates demonstrated motions into a sequence of DMPs and maps them to predefined motion primitives (grasp, move, etc.) thus finding a symbolic representation of the demonstrated motion. The work of [16] derives tasks specified by parameters, where the parameters are invariant across demonstration. A change in these parameters defines a task transition, thus demonstrations can be segmented to subtasks with specific constraints (force, position, etc.).

The intuitiveness and naturalness of robot task programming can be increased by shifting coordinate-based programming (positions and trajectories) to object-based programming [2], [17], [18]. Object-based programming assumes that a general task can be subdivided into skills which are object-centered. Object-centered skills are configured with the parameters that make reference to objects instead of coordinates, e.g., pick, rotate, place, etc. The abstraction of object-centered skills hides low-level implementations which are system specific. Thus, the tasks composed of these skills are more precise and can be transferred and reused on different robot platforms. For example, the "little helper" is an autonomous industrial mobile manipulator [19], which implements task-level programming and enables users to compose tasks by manually selecting skills in a GUI. The skill parameters are obtained through kinesthetic teaching. *Cognitive robots* [17] increase the flexibility and usability of robots in manufacturing, validated with the example of a knitting task in the project STAMINA. *Cognitive automation* considers automatic and flexible decision making in complex environments with an intelligent adaptation of skills [20].

However, flexibility is not sufficient for real world applications since reusability of knowledge is required to handle multiple unmodeled conditions. Reusing the knowledge that has already been acquired can help to realize fast reconfiguration in manufacturing processes. The work of [21] introduces a knowledge integration framework for combining different knowledge representations in robotics. In [22], a method to generate plan descriptions for the automation of manufacturing processes is proposed. This paper uses a knowledge base in combination with ontologies to infer knowledge through reasoning for a given process specification. However, this approach was not tested in a physical system and the reusability of the obtained plans is limited to certain initial conditions.

Safety plays an important role in successfully deploying industrial robots. Safety requirements specified by the Organization for Standardization, such as ISO 10218-1/2 [23], [24], and IEC 61508 [25], have to be fulfilled. A new specification (ISO 15066) [26] specifically addresses safety requirements for physical interactions with robots in fence-less workspaces. Robot system solutions can follow different principles for enabling safe human-robot interactions in shared workspaces. Different scenarios for collaborative operations and their implications for safety are discussed in [27]. The robot introduced in [3] is

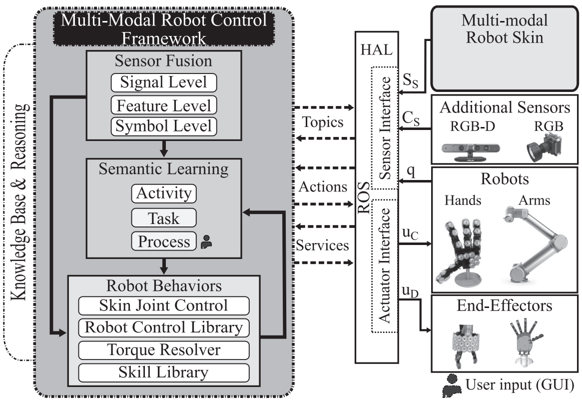
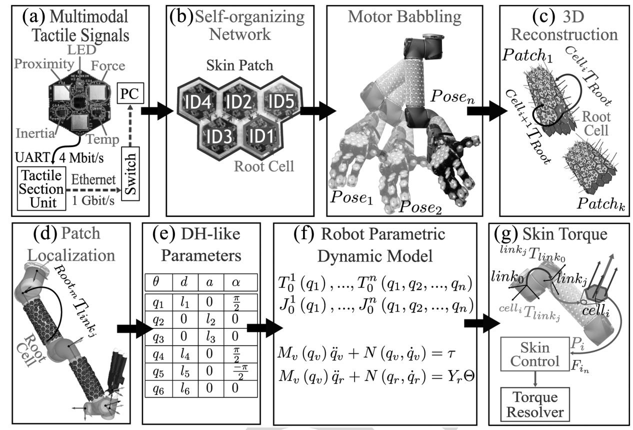


Fig. 2. Overview of the robot framework integrating the technologies: Multimodal robot skin, robot behavior generator, and semantic reasoning engine.



θ	d	a	α
q_1	l_1	0	$\frac{\pi}{2}$
q_2	0	l_2	0
q_3	0	l_3	0
q_4	l_4	0	$\frac{\pi}{2}$
q_5	l_5	0	$\frac{-\pi}{2}$
q_6	l_6	0	0

Fig. 3. Block diagram of our *end-to-end* self-configuring and self-calibrating *robot skin* approach.

inherently safe because of its low inertia and high passive compliance. On the other hand, the Kuka LWR minimizes injury risks by lowering the load-to-weight ratio, enabling fast reactions to collisions, and providing active compliance through force/torque sensing [28]. The ROSETTA project introduces a flexible, collaborative robot for the automatic assembly of small parts [29]. Safe interaction in shared workspaces is enhanced through low payload and inertia, a mechanical design without sharp or pointed edges, cushioning (passive compliance), power and speed limitations, and software based collision detection. However, these principles add constraints to the robot design, making them more expensive, and require a complete redesign of industrial robots.

Q1

B. Main Contributions

In this paper, we extend our previous work [30] to improve our proposed method, which integrates three robotic technologies to allow the fast deployment of industrial robot systems, namely the *Multimodal robot skin*, the *Robot behavior generator*, and the *Semantic reasoning engine*, see Fig. 2. More concretely, our contributions are:

1) An overview on how the proposed technologies can be integrated in an *end-to-end* framework;
2) A multimodal control approach, providing fast reactions to reliably detected *contacts,* and *precontacts* to improve *pHRI*;
3) The enhancement of our semantic reasoning method to kinesthetically teach new activities to robots without the need of an expert robot-programmer;
4) The demonstration of the flexibility and reusability of the framework in different situations, such as the adaptation of the learned processes to new objects, and their transferability to different robots (with different frame of reference), without human intervention.
5) The extension of the reasoning method to detect and handle errors at execution time.

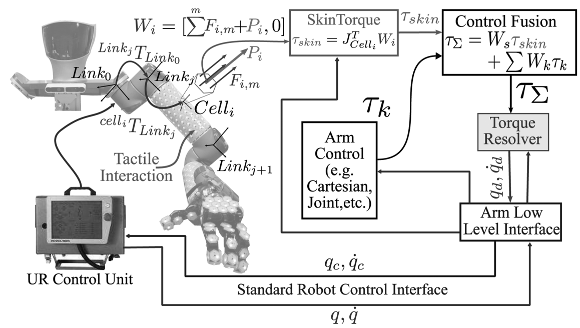
II. Multimodal Robot Skin

Fast, configurable multimodal *robot skin* can transform a standard industrial robot arm into a reactive robot system, enabling pHRI. A robot covered with *robot skin* can detect *contacts* and *precontacts* with high confidence and without occlusion. In combination with appropriate reactive low-level controllers, *robot skin* enables robots to actively mitigate or avoid potentially dangerous situations generated by unexpected changes in the environment, see Section III.

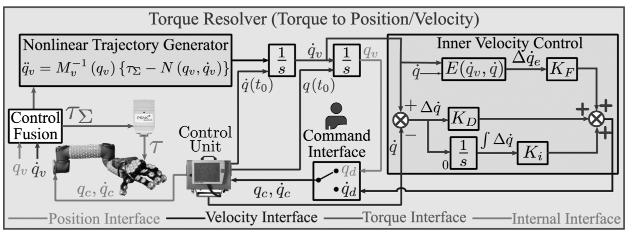
Our *robot skin* [31] is composed of modularized, hexagonally shaped skin cells [see Fig. 3(a)]. Each of these cells utilizes the same set of sensors, which transduce tactile information of different modalities, such as *vibrations* (3-D acceleration sensor), *pressure* (three capacitive force sensors), *pretouch* (optical proximity sensor), and *temperature* (two temperature sensors). The skin cells communicate and exchange information with their neighbors, and build up a self-organized and redundant skin cell network, which enables bidirectional communication with a central processing system, see Fig. 3(a) and (b). The *robot skin* uses the standard Gigabit Ethernet protocol, thus specific drivers are not required, making the *robot skin* easy to deploy. A group of connected skin cells forms a *skin patch*, and each patch has a *root cell*, see Fig. 3(b). These patches are used to cover the robot limbs, see Fig. 3(c) and (d). The *root cell* of a patch is used as a common reference frame for all the skin cells in a patch to define their spatial location on a robot link. This spatial information (homogeneous transformations) is essential to map tactile information to meaningful control commands, and is obtained through a robot skin calibration process.

A. Robot Skin Calibration

Performing manual skin calibration of hundreds of skin cells is prone to errors and totally infeasible. We tackle this challenge by developing a complete *end-to-end robot skin* system, see Fig. 3. The first stage of the system is to explore the cell network using a self-organizing network algorithm, resulting in optimal communication paths and neighbor information of the skin cells [see Fig. 3(b)]. Next, we use motor babbling, a 3-D surface reconstruction algorithm, and an extrinsic calibration algorithm to obtain both, the relative poses of the skin cells with respect to the *root cell*, see Fig. 3(c), and the *root cell* with respect to the *robot link*, Fig. 3(d). From these homogeneous transformations, we obtain a set of Denavit–Hartenberg-like parameters, Fig. 3(e),



(a)



(b)

Fig. 4. (a) General control pipeline to fuse *robot skin* signals in multiple controllers. $F_{i,m}$ and P_i are the forces and proximity signals of a $Cell_i$. $W_i \in \mathbb{R}^{6\times 1}$ is a virtual tactile wrench, where $J_{Cell_i} \in \mathbb{R}^{6\times n}$, with n as the robot's DoF, represents the Jacobian of each cell [32]. W_s and W_k are weight matrices to control the influence of each low-level control and depend on the specific *Robot Behavior.* (b) Torque resolver defined by two principal modules: first, the nonlinear trajectory generator that produces desired trajectories based on user-defined dynamic behaviors, and second, the inner velocity control that generates a desired joint velocity to compensate uncertainties in the robot parameters. $q_v, \dot{q}_v$ represent the joint position/velocity of the virtual robot (desired position/velocity), $q, \dot{q}$ are the joint position/velocity of the real robot, $q_c, \dot{q}_c$ are the commanded joint position/velocities. $E(\dot{q}_v, \dot{q})$ is a joint velocity estimator.

which can be used to generate kinematic models [forward kinematics and robot Jacobians, Fig. 3(f)]. For further details about this calibration process, see [32]. These kinematic models are used to obtain the dynamic model of the robot. This model is defined in a parametric algebraic form, and its dynamic parameters are defined by the designer, see Fig. 3(f). In this manner, the designer can define how the robot should react to the *robot skin* information, i.e., the *desired dynamic behavior of the robot*. Notice that the dynamic parameters do not need to be exact, but they should produce a suitable desired behavior that the real robot is able to generate. A close approximation of these parameters can be obtained using the *Robot Regressor* technique [33]. These models are exploited by our Robot control framework (see Fig. 4(b) in Section III).

B. Event-Driven Robot Skin

The deployment of large-scale *robot skin*[3] introduces new challenges [34]. To tackle these challenges, we investigate and apply biologically inspired principles. The pivotal principle that we use is the novelty driven tactile information transduction, transmission, and processing, i.e., an *event-driven* system. In contrast to synchronous sensors, which continuously transmit information, the sensors of an *event-driven* system only transmit information when there is an event. Events represent *novel information* and are usually triggered by sufficiently large changes in the sensed information. The skin cells of our *event-driven robot skin* asynchronously transmit information in form of *complete sensor values*, this means that this information is not based on differences in the form of deltas [34]. As a result, *event-driven robot skin* improves the performance of real-time control, and allows fast controller responses. For example, this *event-driven* system reduced the network usage of a skin network with 253 skin cells from 1 MB/s to 81 kB/s, and the CPU usage of the controller from around 104% to 42% [35].

[3]We estimate that a completely covered humanoid robot will need at least 3000 skin cells.

	Low-Level Controllers						
Robot Behaviors	Joint Control	Spline Joint Control	Cartesian Control	Spline Cartesian Control	Skin Joint Control	Skin Cartesian Control	G Control
Reach Joint Compliant	✓				✓		✓
Reach Joint Goal Compliant		✓			✓		✓
Reach Cartesian Compliant			✓		✓		✓
Reach Cartesian Goal Compliant				✓		✓	✓
Kinesthetic Joint					✓		✓
Kinesthetic Cartesian						✓	✓

Fig. 5. Figure shows the arm behaviors composed of low-level controllers running in parallel. The gripper behavior is a state machine (open/close).

III. Robot Behaviors

The *Robot Behavior* module provides a library of low-level controllers that can be combined to produce different robot behaviors. This library contains control approaches with low-level control methodologies that produce continuous signals for the robot arms, the hands, and the grippers. Examples of these controllers are depicted in Fig. 5. The *Skin Joint controller* is of particular interest since it transforms the multimodal signals obtained from the *robot skin* into a coherent control signal that can be fused with the other low-level controllers, see Fig. 4(a). The generation of robot behaviors has three steps. The first step is to transform the tactile signals (force and proximity) of the *robot skin* into torque signals *Skin Torque* (τ_{skin}). This is achieved by representing these *robot skin* signals as forces.[4] This is possible since we know the spatial information (pose) of each skin cell, see Fig. 3(c) and (d) in Section II-A. The second step is to fuse the *Skin Torque* signals with the other controllers to obtain different behaviors. In this case, we used a simple weighted sum approach. The third step is to transform this fused control signal into an appropriate command signal. This command depends on the target robot.[5] If the robot uses *torque* commands, the commanded signal will be the fused torque signal. However, if the robot is controlled using *joint positions/velocities*, we use a *Torque Resolver*, see Fig. 4(b). This resolver uses the kinematic and dynamic models obtained in the self-calibration process, see Fig. 3(e)–(g) in Section II-A [32].

[4]We selected force (wrench) as the representation of the signals due to its convenient relation with joint torques $\tau = J^T F$.

[5]In general, we can find three types of command interfaces for robots: joint position, joint velocity, and joint torque commands. The first two are the most common interfaces for industrial robots.

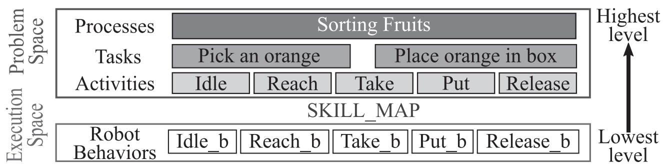


Fig. 6. Hierarchical structure of our reasoning system. The Problem Space (purple box) provides semantic descriptions which represent robot-agnostic knowledge. The Execution Space (red box) is the information needed to execute robot motions. This information is robot-dependent.

The fusion of the different low-level controllers produces various behaviors. These combinations are defined in the *Robot Behavior* module, e.g., the *Reach Cartesian Goal Compliant* behavior fuses the *Cartesian*, the *Skin Torque*, and the *Gravity compensation* controllers, see Fig. 5. The previously discussed controller allows the robot to follow a Cartesian trajectory with the end-effector while at the same time can react to the tactile stimuli. This *Robot Behavior* module is the interface between the *Semantic Reasoning Engine* module, see Section IV, and the robot's low-level controllers.

IV. Semantic Reasoning Engine

We developed a semantic reasoning engine that allows nonexpert users to teach robots new tasks simply by guiding the robot's end-effector. To this aim, we use a hierarchical learning approach that is able to extract and interpret *low-level* features from the robot's end-effector as well as from its sensors to automatically generate compact semantic rules. The reasoning engine uses the obtained semantic rules to infer the robot activities as well as tasks from human Kinesthetic demonstrations (*high-level*). Fig. 6 exemplifies the transition between the lowest level to the highest level used in our system. The lowest level represents robot behaviors, which are defined in the *execution space*. These behaviors represent the primitives that the robots can execute. The robot motions are automatically interpreted by our reasoning system as *activities*, e.g., "Reach" and "Take." Our reasoning system is also able to combine a set of different activities into a *task* (see Fig. 6). Finally, the user defines a new *process* using the tasks provided by our system. *Processes*, *Tasks*, and *Activities* are described in the *Problem Space*, and they are robot agnostic descriptions.

In order to automatically interpret the kinesthetic demonstrations, our learning system transforms the continuous signals obtained from the demonstrations to symbolic representations [36]. For example, the motions (m) of the robot's end-effector (ef) are interpreted as either *Move* or *Not Move* symbols. Where *Move*: the end-effector is moving, i.e., $\dot{x} > \varepsilon$ and *Not Move*: the end-effector stops its motion, i.e., $\dot{x} \to 0$, where $\dot{x}$ is the end-effector velocity and ε is a heuristically defined threshold. In addition, the information about the perceived environment is also transformed into symbolic representations. For the demonstration scenario described in Section V, the robot tactile omnidirectional mobile manipulator (TOMM) can perceive its environment through the following sensors: *robot skin*, RGB-D camera, and joint sensors. From these sensors the following abstract properties can be defined:

1) *ObjectActedOn*[6] (o_a): The end-effector is moving towards an object, $d(x_{ef}, x_{o_i}) \to 0$;
2) *ObjectInHand* (o_h): The object is in the end-effector, i.e., $d(x_{ef}, x_{o_i}) \approx 0$, where $d(\cdot,\cdot)$ is the Euclidean distance between the end-effector (x_{ef}) and the detected object (x_{o_i});
3) *GripperState* (g_s): The current state of the gripper (open/closed).

After transforming the perceived environment and the robot motions into symbolic representations, $s_s = \{m, o_a, o_h, g_s\}$, we train a decision tree (T) using only one kinesthetic demonstration. We follow a similar pipeline as the one presented in [36], where the C4.5 algorithm is employed to compute T. This tree contains semantic descriptions of the robot motions represented by *if-then rules*, which are human readable:

$$if\ ef(Move)\ \&\ ObjActOn(Fruit)\ \&\ Gripper(Open) \rightarrow Act(\mathbf{Reach}) \quad (1)$$

The rules obtained from T are enhanced with our knowledge and reasoning engine. The knowledge base is defined by an ontology representation, expressed in the Web Ontology Language. The reasoning is based on description logics, such as Prolog queries. Note that (1) contains the class "Fruit" rather than the trained object "Orange" which means that the extracted semantic representation for the *activity* "Reach" can be reused for all the objects that belong to the general class "Fruit". Consequently, the more general we create the semantic representations, the more demonstrations the robot can interpret without the need to retrain, thus making the reasoning system reusable in different situations [37], [38]. When a new activity is inferred by the reasoning system, a *Robot Behavior* is associated with this activity and stored in an abstract *Robot Behavior* representation (*Skill Map*), see Fig. 6. The *Skill Map* contains the necessary parameters to execute the demonstrated activity by the robot, thus allowing the transition between the *execution space* and the *problem space*. For example, when the reasoning system infers the activity "PutSomethingSomewhere," the parameters generated in the *skill map* are *something*, *somewhere,* and *Robot Behavior*, where *something* is instantiated when a new object is detected (orange), *somewhere* identifies the final position of the activity (box) and the executed *Robot Behavior* is *Reach Cartesian*. Then, a directed *task graph* is obtained where vertices represent the inferred activities and edges represent transitions between activities. The *pre-* and *post-* conditions of each activity are also obtained [39]. Furthermore, the reasoning system can provide the sequence of activities that compose a task. The user can also define a new *process* by selecting the desired tasks and a stopping criterion,[7] without the specification of additional parameters.

[6]The information from the object can be obtained either from the vision system or the proximity sensor of the skin. The same is valid for the property *ObjectInHand*.

[7]The stop criterion indicates when a process should stop, e.g., duration, weight, or number of objects.

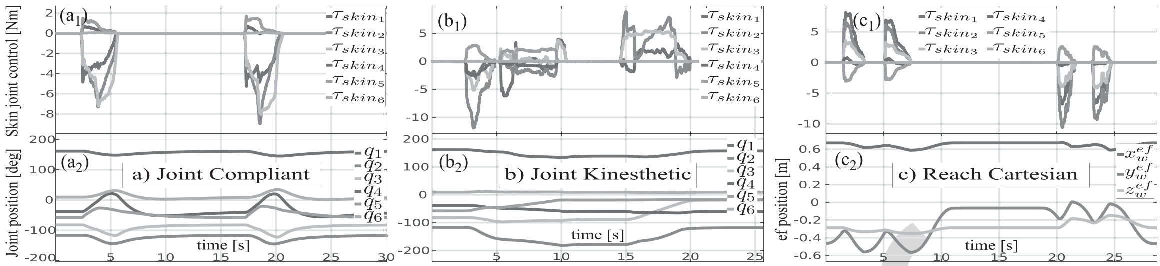
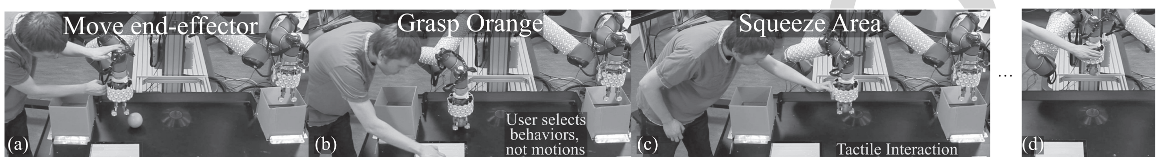


Fig. 7. Different robot behaviors obtained with the Multi-modal Robot Control Framework. The Skin joint control uses the *robot skin* to generate the skin torque. This torque reflects tactile interactions with the robot. (a) The robot is transformed into a compliant system. Each time there is a physical interaction, reflected in the skin joint control torque (a_1), the joint position changes (a_2). (b) For the kinesthetic teaching mode, the robot changes its position (b_2) when the user physically interacts with it (b_1). In this case, the robot does not return to its original position after the interaction. (c) When a user interferes with the robot's motion (c_1), the robot reacts and smoothly changes its trajectory (c_2). As soon as the obstacle is no longer present, the robot restarts the task until it reaches the goal (c_2).



Input (User)	Output (Sem.Eng.)	Input (User)	Output (Sem.Eng.)	Input (User)	Output (Sem.Eng.)	Input (U)	Output (SE)
Resource: Right Arm		Resource: Right Gripper		Resource: Right Arm		...	...
Behavior: Select Kinesthetic Cartesian	Skill Map: Reach Cartesian Behavior	Behavior: Select Close Gripper	Skill Map: Close Gripper Behavior	Behavior: Select Kinesthetic Cartesian	Skill Map: Reach Cartesian Behavior	...	...
Demonstration: Move end-effector to orange	Inferred Activity: Reach	Demonstration: Grasp the orange with gripper	Inferred Activity: Take	Demonstration: Move end effector to squeeze area	Inferred Activity: Put something somewhere	...	...
Task 1: Pick Fruit				Task 2: Identify Good Fruit			
Process: Sorting Fruits							

Fig. 8. Semantic reasoning engine interprets user demonstrations. The input of the reasoning engine (white boxes) comes from the human, i.e., he/she selects the *Robot Behavior* and demonstrates the activities. The output (gray boxes) is defined with two abstraction levels. The first level, (light gray) represents the inferred activities and their associated *Robot Behavior*. The second level (dark gray) represents the tasks generated from the inferred activities, using the sequence followed by the user during the demonstration. The user can build different processes as needed (lower dark gray box).

V. Demonstration Scenarios & Validation

A. Robot Platform & Evaluation of Control Behaviors

To evaluate our multilevel control framework, our tactile omni-directional mobile manipulator with more than 600 skin cells was used [40]. These skin cells cover the arms and the grippers. The grippers have three different skin patches: hand patches to detect obstacles, external finger patches to detect the texture of the fruits (stiff or soft), and internal finger patches to detect when the grippers are grasping an object. TOMM has three-fingered grippers which are inherently compliant and suitable to handle fruits without damaging them. We consider the following three different dynamic behaviors:

1) *Joint Compliant* behavior,
2) *Kinesthetic Joint* behavior, and
3) *Reach Cartesian Goal* behavior.

In the *Joint Compliant* behavior [see Fig. 7(a)], the robot reacts to tactile events and changes its position, e.g., when the user applies forces to the arm. When there are no external perturbations, the robot *returns to its original position* smoothly. The *Kinesthetic Joint* behavior [see Fig. 7(b)] is similar to the Joint Compliant behavior, but instead of returning to its original position, the robot will stop and *will remain in this position* as long as no further tactile events are detected. In the *Reach Cartesian Goal* behavior [see Fig. 7(c)], the goal position of the robot's end-effector is defined by the user. Then, a trajectory to reach this goal is computed using a spline function. The robot arm follows this trajectory and when the user interferes with the robot (detected by the skin sensors), this event produces a compliant reactive behavior which forces a *change in the robot trajectory*.

B. Sorting Fruits Scenario

As a demonstration scenario, we consider the task of sorting fruits. With this scenario, we can highlight the benefits of using the tactile and proximity sensors on the *robot skin* to sense the quality of the fruits.[8] The user teaches the robot the activities and the intermediate tasks required to sort oranges: 1) Good oranges (with stiff texture) will be placed in a box, and 2) Bad oranges (with soft texture) will be thrown into the trash bin, see Fig. 8. The texture of the oranges is evaluated using the force sensors from the *robot skin* placed in the external finger patches of the grippers. The stiffness threshold to discriminate the texture of the fruits is defined during the demonstration. Our approach consists of two phases: *Teaching* and *Execution*.

C. Kinesthetic Teaching With Semantic Inference

In our previous work [41], we obtained semantic models of human activities for "making a pancake" using the iCub robot. From these models, we obtained common descriptions such as "Reach," "Take," etc., along with common tasks such as "Picking an object." These semantic descriptions were reused and extended to teach a Humanoid robot H1 (REEM-C) how to "make

[8]This scenario was inspired by the standard process of orange sorting where humans use their tactile sensation to discriminate good and bad oranges.

Perceived Object	Task 0 Reset		Task 1 Pick Fruit		Task 2 Identify Good Fruit		Task 3 Put Fruit into Box		Task 4 Identify Bad Fruit		Task 5 Put Fruit into Bin	
Orange	Pre-cond	Post-cond	Pre-cond	Post-cond	Pre-cond	Post-cond	Pre-cond	Post-cond	Pre-cond	Post-cond	Pre-cond	Post-cond
	OIH=none Hand=any	OIH=none Hand=open	OIH=none Hand=open	OIH=orange Hand=close	OIH=orange Hand=close Squeeze=0.0	OIH=orange Hand=close Squeeze ≥0.3 Dest=s_a	OIH=orange Hand=close Dest=s_a	OIH=none Hand=open Arm=any Dest=box	OIH=orange Hand=close Squeeze=0.0	OIH=orange Hand=close Squeeze<0.3 Dest=s_a	OIH=orange Hand=close Dest=s_a	OIH=none Hand=open Arm=any Dest=bin
Apple	Pre-cond	Post-cond	Pre-cond	Post-cond	Pre-cond	Post-cond	Pre-cond	Post-cond	Pre-cond	Post-cond	Pre-cond	Post-cond
	OIH=none Hand=any	OIH=none Hand=open	OIH=none Hand=open	OIH=apple Hand=close	OIH=apple Hand=close Squeeze=null	OIH=apple Hand=close Squeeze ≥0.3 Dest=s_a	OIH=apple Hand=close Dest=s_a	OIH=none Hand=open Arm=any Dest=box	OIH=apple Hand=close Squeeze=null	OIH=apple Hand=close Squeeze<0.3 Dest=s_a	OIH=apple Hand=close Dest=s_a	OIH=none Hand=open Arm=any Dest=bin

Stop Criteria

Fig. 9. Learned process of "Sorting Fruits". The semantic system verifies the pre-conditions of each task before starting its execution. The first row depicts this process when the perception system detected *oranges*. The second row shows the automatic adaptation of the same process for *apples*. Task 2 and Task 3 have been rejected since the class "Apple" does not have the *Squeeze* property. When there is an anomaly, for example, dropping an orange, the system can detect the error and the following tasks are rejected.

a sandwich" [42]. In this case, the semantic descriptions were extended to include new activities, such as "Cut" and new tasks, such as "Cutting an object". These semantic descriptions are the initial knowledge-base for our robot TOMM (*robot experience*) to learn how to "sort fruits". This is possible since the semantic models are robot-agnostic and defined in the *Problem Space*, see Fig. 6. During the *teaching* phase, our reasoning system is extended to generate and populate new semantic descriptions for the "sorting fruit" scenario through human demonstrations. For these demonstrations, the user kinesthetically guides the robot by selecting a specific *Robot Behavior*, see Fig. 5.

Fig. 8 depicts the pipeline of the reasoning engine which uses human demonstrations as input (low-level sensor information) and produces the inferred activities as output (high-level interpretations). These activities will be connected by the reasoning engine to generate the demonstrated tasks, following the sequence taught by the user. Each inferred activity will be associated with a specific *Robot Behavior* through the *Skill Map*. For example, in Fig. 8(a), the user selects the right arm as the desired resource for teaching. Then, the user selects the *Kinesthetic Cartesian* behavior to move the robot's end-effector towards the orange. This demonstration is inferred by the semantic engine, as the activity "Reach", and the *Skill Map* connects this activity with the robot behavior *Reach Cartesian*. Similarly, Fig. 8(b) depicts the user selecting the gripper as the desired resource and *Close Gripper* as its behavior. This new demonstration is interpreted by the semantic engine, as the activity "Take". Then, these two activities ("Reach" and "Take") are automatically connected and defined as the task "Pick Fruit" by the reasoning system. As part of the task definition, the reasoning system defines the *pre-* and *post-* conditions required for each task, see Fig. 9. All these generated tasks will be stored in the knowledge base and can be retrieved by the user to define new processes. The processes and the tasks are specified using abstract representations (semantic level), which make the system highly flexible. Note that the reasoning system abstracts the meaning of the demonstrated tasks, instead of storing motion patterns, or low-level information, such as velocities, trajectory patterns, end-effector positions, etc. For example, instead of saving the trajectory from the box to the orange while moving the arm, the learning system defines this activity as "Reaching".

Fig. 9 depicts the automatically generated tasks, for the process "Sorting Fruits". In this case, five tasks were generated: 1) *Pick fruit*, 2) *Identify good fruit*, 3) *Put fruit into a box*, 4) *Identify bad fruit*, and 5) *Put fruit to trash*.[9] Our reasoning method also provides the option to create tasks by manually selecting and connecting the available activities from the acquired knowledge base (using a GUI).

The main feature of this learning system is that a nonexpert user can teach the robot new tasks. Since the user does not need to program the robot directly (using a teach-pendant and a specific robot language), but rather the user guides the robot and the system generates the proper sequences in a human readable form. In this context, the robot's program is replaced by a sequence of tasks where their parameters are defined at runtime.

We tested the integration of the presented three technologies, using the demonstrations from two different participants.[10] Four demonstrations were considered with random positions for the oranges [43]. The robot is able to recognize the demonstrated activities using our reasoning technology with an average accuracy of around 83%[11] when the participants kinesthetically showed the desired activities for the *sorting fruits* process. The sequence of recognized activities is consecutively stored to automatically define task structures to accelerate the learning of new tasks [39]. Ongoing research on a formal validation, including multiple participants with different backgrounds, is currently being pursued.

To teach a new process to a robot, a normal user takes approximately two minutes and 44 s. Two minutes and 6 s to kinesthetically teach the activities and create the tasks, and 38 s to build a new process and launch it [39]. These times depend on the complexity of the new process, in this case, the process consists of five tasks. After that, no more user intervention is required, even when new objects appear in the scene.

D. Execution of the Demonstrated Tasks

First, the user indicates the process to be executed, see Fig. 10. Then, the framework verifies if the process can be executed in the current environment. For this, the semantic engine loads the tasks of the selected process, see Fig. 9. Prior to executing a task, the system verifies if all the task preconditions are satisfied. If a precondition is not satisfied, the task fails (we exploit

[9] The system connects the activities to compose the tasks, and the user defines the labels for these tasks.

[10] One participant was a robotic expert and the other nonexpert.

[11] This accuracy is obtained by comparing the recognized activities between the reasoning system and the ground truth (manual annotations).

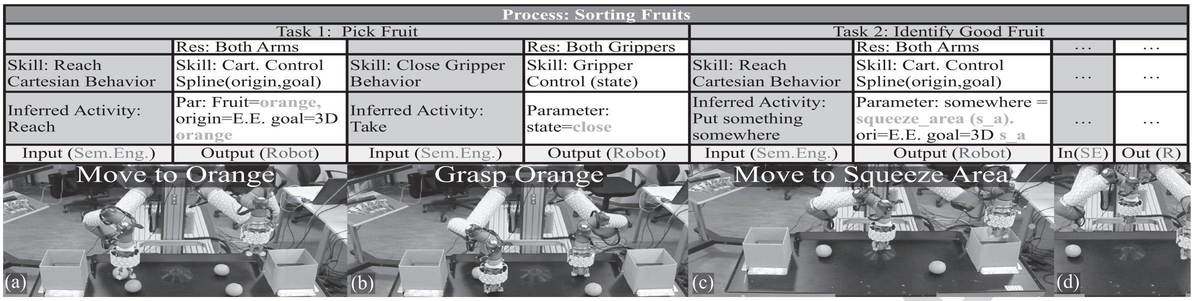


Fig. 10. Execution of the learned process. The semantic engine analyzes all the tasks and their activities of the process and executes the associated robot behaviors. The input of the execution phase is the semantic process, and the output is the execution of the robot behaviors with the required parameters.

this verification process to detect errors, see Section V-F). Then, for each task, the semantic engine infers the sequence of activities at runtime. Each activity has an associated *Robot Behavior*, which requires specific parameters for its execution. These parameters are obtained at runtime, using the perception system, which identifies and labels the objects in the scene. The semantic engine uses these labels to create instances of the proper class of the identified object. The instances are single data containers (semantic abstractions) that provide multiple information from the perceived object, e.g., parent class, type, color, size, position, orientation, squeeze ratio, etc. For the resources (arms and grippers) we use the following information joint/end-effector positions, velocities. For example, the perception system identifies an orange and obtains its 3-D-position. Then, an instance of the class "Orange" is created and its property named "Position" is populated with the orange's position.

The semantic engine starts to execute each activity with its associated *Robot Behavior* and its targeted object. Each executed behavior triggers a set of low-level controllers, which requires different parameters. For example, in Fig. 10(a), the Task 1 "Pick Fruit" executes two sequential activities: "Reach" and "Take". The unbounded variable *Fruit* is instantiated with the perceived object (an instance of class "Orange").[12]

The activity "Reach" is associated with the *Reach Cartesian* behavior, see Fig. 8(a). This behavior executes in parallel three low-level controllers *Spline Cartesian*, *Skin Cartesian*, and *Gravity compensation*, see Fig. 5. The *Spline Cartesian* control requires two parameters the *origin* and the *goal*. The *origin* is set, by default, as the *current position of the end-effector*, and the *goal* is defined, by the semantic engine, as the *orange's position*. When the low-level controllers are finished, the next activity will be executed. For the activity "Squeeze," the robot places the end-effector over the fruit, and moves down slowly until the fruit is detected by the *robot skin* (soft contact). Then, the end-effector moves down again a few milliliters and measures the contact force. This force is correlated to the stiffness of the fruit. The rest of the tasks and their activities are executed until the process is finished. Note that during the teaching phase, the user demonstrated the process using only one arm,

[12]The perception system can detect different objects from different classes, e.g., oranges and apples (class "Fruit"), box and trash bin (class "Container"), etc. The semantic reasoning discriminates and uses these objects according to their class.

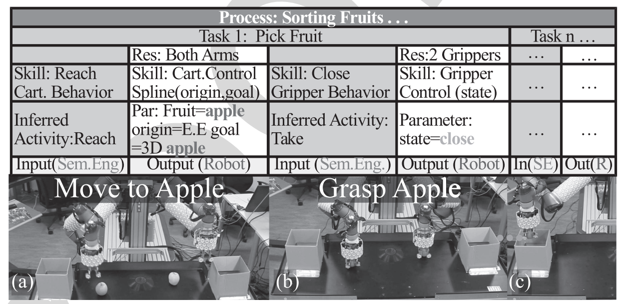


Fig. 11. Executing the inferred process for apples. The same process "Sorting Fruits" can be used without user intervention or reprogramming, even when the process was generated using *oranges*. The inferred execution is the robot grasping the apples and putting them into the boxes without squeezing them. This is a correct execution since the apples should not be squeezed.

i.e., the right arm. However, since the activities and their parameters are defined with abstract representations, the same process can be used with multiple resources (robots). This makes the descriptions general, transferable, and reusable.

The semantic system verifies the available resources of the robot for the execution of the desired task. In this case, the robot has the right and the left arms enabled, and there is more than one orange on the table, see Fig. 10. Therefore, exactly, the same process ("Sorting Fruits") is executed with both arms (the only difference is the resource assigned by the semantic engine).

E. Handling Variations in the Process: Apple

The obtained general descriptions of tasks and activities allow our generated semantic process to work also with objects different from the ones used during the teaching phase. In Fig. 11, the scenario presents a different object (apples). In this case, the variable *Fruit* is an instance of the class "Apple" and the corresponding semantic properties of this class are loaded. However, the class "Apple" does not have the property "squeezable". Therefore, Task 2 and Task 4 can not be executed, see Fig. 9 (Squeeze = null). In this case, only the tasks that can be executed for *apples* are Task 1, Task 3, and Task 5 (tasks that do not depend on "squeeze" property). These tasks are executed sequentially. As a result, the robot takes the apple and place it into the box. Our reasoning system only takes approximately 0.124 s to make this new execution plan compared to 38 s required to generate the orange sorting plan.

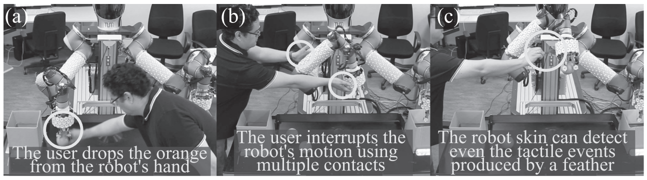


Fig. 12. Error detection and user perturbations. (a) Using the internal finger patches, the semantic reasoning detects this error and infers which task can be executed. (b) The robot safely reacts to the human interaction and avoids collisions. (c) Exploiting both the force and the proximity sensors, the *robot skin* can detect even a feather.

F. Error Detection

The structure of the process description allows the semantic engine to detect errors through the verification of the preconditions on each task. Fig. 12(a) shows the case when the user removes the orange from the robot's gripper at the end of Task 1. This anomaly is detected by the reasoning engine, i.e., *Error Detection*. As can be seen in Fig. 9, Task 2 requires as precondition that the gripper has an object in the hand (OIH = orange) since this precondition is not satisfied, then all the following tasks will fail. Hence, the sequence Task 0–Task 1 will be repeated until the gripper has an object in hand. In this case, the system also provides our first approach to *Error Handling*. The time that our system takes to detect this error and to search for a new strategy takes around 0.38 s.

G. Physical Human-Robot Interaction

One important aspect considered in our system is the pHRI, where safe interaction is paramount. In Fig. 5 can be seen that all the behaviors contain either *Skin Joint Control* or *Skin Cartesian Control*. These two controllers make the robot reactive to tactile events (*precollisions* and *pressure*) allowing physical interactions. Fig. 12(b) and (c) show two examples of these reactive interactions. Other safety mechanisms, we have adopted in our robot skin are redundancy of communication paths in the skin cell network, redundancy in skin sensor modality (e.g., contact detection through proximity and force sensors), and real-time user feedback through RGB LEDs of the skin cells.[13] In order to validate our rapidly deployable robot system, we successfully installed the *robot skin* on two different robot arms in two different laboratories. First, we fully covered a UR5 robot with 410 skin cells using 13 patches. In addition, we also covered the forelimb of a UR10 arm with 373 skin cells. The deployment from installing the skin patches in the robots to a fully calibrated and ready to use *robot skin* took in both cases about 5 h.

We provide a video[14] to illustrate the robot behaviors in our robot TOMM using the proposed approach, and to show the teaching and execution phases for the process "Sorting Fruits."

[13]The authors consider these mechanisms only as starting points for functional safety and standardization, and we only highlight the system's potential regarding safety. Nevertheless, the process for productization of these systems is still in a preliminary state.

[14]https://youtu.be/_X255OyzGs0

VI. Conclusion

The overview of the integration of three main robotic technologies was presented in this paper. These technologies enable fast deployment of industrial robot systems and consists of a fast self-configurable artificial skin, a multimodal control framework to extend the dynamic behaviors of standard robots, and a robust and intuitive teaching method based on semantic reasoning. The presented results demonstrate that these technologies enhance the *usability*, *flexibility*, and introduce our first approach to handle *safety* for industrial robots, especially when a nonexpert user teaches the robot new processes using pHRI. The *usability* is demonstrated with the following aspects:

1) Novel technologies to teach robots new tasks using pHRI;
2) The extension of a semantic reasoning engine to automatically infer activities and tasks from human demonstrations;
3) The generation of processes in human-readable form via semantic descriptions, and;
4) Error detection and handling during process execution without human intervention.

The *flexibility* is validated as follows: 1) *end-to-end* robot skin framework for fast deployment on different robots, and 2) a knowledge-base system that allows the re-usability and transferability of learned skills. The *safety* is realized through a reactive control framework based on multimodal *robot skin* avoiding dangerous collisions during HRI. The presented framework can be implemented in any standard industrial robot as long as it provides an external control interface.

References

[1] A. Jäger, C. Moll, O. Som, C. Zanker, S. Kinkel, and R. Lichtner, "Analysis of the impact of robotic systems on employment in the European Union. Final report," Publications Office of the European Union, Luxembourg, 2015.

[2] M. R. Pedersen *et al.*, "Robot skills for manufacturing: From concept to industrial deployment," *Robot. Comput.-Integr. Manuf.*, vol. 37, pp. 282–291, 2016.

[3] T. Lens, J. Kunz, O. Von Stryk, C. Trommer, and A. Karguth, "Biorob-arm: A quickly deployable and intrinsically safe, light-weight robot arm for service robotics applications," in *Proc. 6th German Conf. Robot.*, 2010. Q2

[4] R. D. Schraft and C. Meyer, "The need for an intuitive teaching method for small and medium enterprises," *37th Int. Symp. Robot., 4th German Conf. Robot.*, 2006, vol. 1956.

[5] A. D. Santis, B. Siciliano, A. D. Luca, and A. Bicchi, "An atlas of physical human–robot interaction," *Mechanism Mach. Theory*, vol. 43, no. 3, pp. 253–270, 2008.

[6] R. Dillmann *et al.*, "Advances in robot programming by demonstration," *KI-Künstliche Intelligenz*, vol. 24, no. 4, pp. 295–303, 2010.

[7] B. Hein, M. Hensel, and H. Wörn, "Intuitive and model-based on-line programming of industrial robots: A modular on-line programming environment," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2008, pp. 3952–3957.

[8] P. Neto, J. Norberto Pires, and A. Paulo Moreira, "High-level programming and control for industrial robotics: Using a hand-held accelerometer-based input device for gesture and posture recognition," *Ind. Robot, Int. J.*, vol. 37, no. 2, pp. 137–147, 2010.

[9] G. Du, P. Zhang, and X. Liu, "Markerless human-manipulator interface using leap motion with interval kalman filter and improved particle filter," *IEEE Trans. Ind. Informat.*, vol. 12, no. 2, pp. 694–704, Apr. 2016.

[10] Z. Pan, J. Polden, N. Larkin, S. Van Duin, and J. Norrish, "Recent progress on programming methods for industrial robots," *Robot. Comput.-Integr. Manuf.*, vol. 28, no. 2, pp. 87–94, 2012.

[11] G. Grunwald, G. Schreiber, A. Albu-Schäffer, and G. Hirzinger, "Programming by touch: The different way of human–robot interaction," *IEEE Trans. Ind. Electron.*, vol. 50, no. 4, pp. 659–666, Aug. 2003.

[12] D. Massa, M. Callegari, and C. Cristalli, "Manual guidance for industrial robot programming," *Ind. Robot*, vol. 42, no. 5, pp. 457–465, 2015.

[13] T. Wösch and W. Feiten, "Reactive motion control for human-robot tactile interaction," in *Proc. Int. Conf. Robot. Autom.*, 2002, vol. 4, pp. 3807–3812.

[14] S. Calinon, F. D'halluin, E. L. Sauser, D. G. Caldwell, and A. G. Billard, "A probabilistic approach based on dynamical systems to learn and reproduce gestures by imitation," *IEEE Robot. Autom. Mag.*, vol. 17, no. 2, pp. 44–54, 2010.

[15] P. Kormushev, S. Calinon, and D. G. Caldwell, "Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input," *Adv. Robot.*, vol. 25, no. 5, pp. 581–603, 2011.

[16] A. L. P. Ureche, K. Umezawa, Y. Nakamura, and A. Billard, "Task parameterization using continuous constraints extracted from human demonstrations," *IEEE Trans. Robot.*, vol. 31, no. 6, pp. 1458–1471, Dec. 2015.

[17] V. Krueger *et al.*, "A vertical and cyber-physical integration of cognitive robots in manufacturing," *Proc. IEEE*, vol. 104, no. 5, pp. 1114–1127, May 2016.

[18] M. N. Nicolescu and M. J. Mataric, "Natural methods for robot task learning: Instructive demonstrations, generalization and practice," in *Proc. Second Int. Joint Conf. Auton. Agents Multiagent Syst.*, 2003, pp. 241–248.

[19] C. Schou, J. S. Damgaard, S. Bogh, and O. Madsen, "Human-robot interface for instructing industrial tasks using kinesthetic teaching," in *Proc. Int. Symp. Robot.*, 2013.

[20] D. Bruckner, H. Zeilinger, and D. Dietrich, "Cognitive automation—Survey of novel artificial general intelligence methods for the automation of human technical environments," *IEEE Trans. Ind. Informat.*, vol. 8, no. 2, pp. 206–215, May 2012.

[21] A. Björkelund, H. Bruyninckx, J. Malec, K. Nilsson, and P. Nugues, "Knowledge for intelligent industrial robots," in *Proc. AAAI Spring Symp., Des. Intell. Robots*, vol. 12, no. 2, 2012.

[22] J. Puttonen, A. Lobov, and J. L. M. Lastra, "Semantics-based composition of factory automation processes encapsulated by web services," *IEEE Trans. Ind. Informat.*, 2013, vol. 9, no. 4, pp. 2349–2359, Nov. 2013.

[23] *Robots and Robotic Devices – Safety Requirements for Industrial Robots—Part 1: Robots*, ISO 10218-1, 2010.

[24] *Robots and Robotic Devices – Safety Requirements for Industrial Robots–Part 2: Robot Systems and Integration*, ISO 10218-2, 2011.

[25] *Functional Safety of Electrical-Electronic-Programmable Electronic Safety-Related Systems. International electrotechnical commission*, IEC 61508, 1998.

[26] *Safety for Collaborative Industrial Robots*. Technical Standard ISO 15066, 2014.

[27] J. Fryman and B. Matthias, "Safety of industrial robots: From conventional to collaborative applications," in *Proc. 7th German Conf. Robot.*, 2012.

[28] A. Albu-Schäffer, S. Haddadin, C. Ott, A. Stemmer, T. Wimböck, and G. Hirzinger, "The DLR lightweight robot: design and control concepts for robots in human environments," *Ind. Robot, Int. J.*, vol. 34, no. 5, pp. 376–385, 2007.

[29] R. Patel, M. Hedelind, and P. Lozan-Villegas, "Enabling robots in small-part assembly lines: The "ROSETTA approach" - an industrial perspective," in *Proc. 7th German Conf. Robot.*, 2012.

[30] E. Dean-Leon, K. Ramirez-Amaro, F. Bergner, I. Dianov, P. Lanillos, and G. Cheng, "Robotic technologies for fast deployment of industrial robot systems," in *Proc. IECON 2016–42nd Annual Conf. IEEE Ind. Electron. Soc.*, Oct. 2016, pp. 6900–6907.

[31] P. Mittendorfer and G. Cheng, "Humanoid multimodal tactile-sensing modules.," *IEEE Trans. Robot.*, vol. 27, no. 3, pp. 401–410, Jun. 2011.

[32] E. Dean-Leon, F. Bergner, K. Ramirez-Amaro, and G. Cheng, "From multi-modal tactile signals to a compliant control," in *Proc. 2016 IEEE-RAS 16th Int. Conf. Humanoid Robots*, Nov. 2016, pp. 892–898.

[33] E. Dean-Leon, S. Nair, and A. Knoll, "User friendly Matlab-toolbox for symbolic robot dynamic modeling used for control design," in *Proc. 2012 IEEE Int. Conf., Robot. Biomimetics*, Dec. 2012, pp. 2181–2188.

[34] F. Bergner, E. Dean-Leon, and G. Cheng, "Event-based signaling for large-scale artificial robotic skin - Realization and performance evaluation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 4918–4924.

[35] F. Bergner, E. Dean-Leon, and G. Cheng, "Efficient event-driven reactive control for large scale robot skin," in *Proc. IEEE Int. Conf. Robot. Autom.*, to be published.

[36] K. Ramirez-Amaro, M. Beetz, and G. Cheng, "Understanding the intention of human activities through semantic perception: Observation, understanding and execution on a humanoid robot." *Adv. Robot.*, vol. 29, no. 5, pp. 345–362, 2015.

[37] K. Ramirez-Amaro, M. Beetz, and G. Cheng, "Transferring skills to humanoid robots by extracting semantic representations from observations of human activities," *Artif. Intell.*, vol. 247, pp. 95–118, 2017.

[38] K. Ramirez-Amaro, H. N. Minhas, M. Zehetleitner, M. Beetz, and G. Cheng, "Added value of gaze-exploiting semantic representation to allow robots inferring human behaviors.," *ACM Trans. Interactive Intell. Syst.*, vol. 7, no. 1, pp. 5:1–5:30, 2017.

[39] I. Dianov, K. Ramirez-Amaro, P. Lanillos, E. Dean-Leon, F. Bergner, and G. Cheng, "Extracting general task structures to accelerate the learning of new tasks," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, 2016, pp. 802–807.

[40] E. Dean-Leon *et al.*, "TOMM: Tactile omnidirectional mobile manipulator," in *Proc. 2017 IEEE Int. Conf. Robot. Autom.,* May 2017, pp. 2441–2447.

[41] K. Ramirez-Amaro, M. Beetz, and G. Cheng, "Automatic segmentation and recognition of human activities from observation based on semantic reasoning," in *IEEE/RSJ Int. Conf. Intell. Robots Syst*., Sep. 2014, pp. 5043–5048.

[42] K. Ramirez-Amaro, E. C. Dean-Leon, and G. Cheng, "Robust semantic representations for inferring human co-manipulation activities even with different demonstration styles." in *Proc. 15th Int. Conf. Humanoids Robots.*, 2015, pp. 1141–1146.

[43] K. Ramirez-Amaro, E. C. Dean-Leon, I. Dianov, F. Bergner, and G. Cheng, "General recognition models capable of integrating multiple sensors for different domains." in *Proc. 16th Int. Conf. Humanoids Robots*, 2016, pp. 306–311.

Emmanuel Dean-Leon (M'XX) received the M.Sc. and Ph.D. degrees in mechatronics from the Center for Research and Advanced Studies, Mexico City, Mexico, in 2003 and 2006, respectively.

He is currently a Senior Researcher at the Chair for Cognitive Systems, Technical University of Munich, Munich, Germany. His research interests include robotics, low-level control, and physical human-robot-interaction.

Karinne Ramirez-Amaro (M'XX) received the M.Sc. degree in computer science at the Center for Computing Research, Mexico city, Mexico, in 2007 and 2015, respectively, and the Ph.D. degree (*summa cum laude*) in electrical engineering from the Technical University of Munich (TUM), Munich, Germany.

Since 2015, she has been a Postdoctoral Researcher at the Chair for Cognitive Systems, TUM. Her research interests include semantics representations, expert systems, human activity recognition, and programming by demonstration.

Florian Bergner (M'XX) received the B.Sc. and M.Sc. degree in electrical engineering from the Technical University of Munich (TUM), Munich, Germany, in 2011 and 2014, respectively. Since 2014, he has been working toward the Ph.D. degree at the Chair for Cognitive Systems, TUM.

His research interests include multimodal tactile sensing, large-scale robot skin, sensor networks, sensor fusion, neuromorphic engineering, and event-driven signaling and processing.

Ilya Dianov (M'XX) received the M.S. and B.S. degree in computer science from Moscow Power Engineering University, Moscow, Russia, in 2011, and the M.Sc. degree in computer science from the Technical University of Munich (TUM), Munich, Germany, in 2015. He is currently working toward the Ph.D. degree at the Chair of Cognitive Systems, TUM.

His research interest include semantic reasoning and task planning.

Gordon Cheng (F'XX) received the Ph.D. degree in systems engineering from the Australian National University, Canberra, ACT, Australia, in 2001.

He is currently a Professor of the Chair for Cognitive Systems the Technical University of Munich, Munich, Germany. From 2002 to 2008, he was the Head of the Department of Humanoid Robotics and Computational Neuroscience, ATR Computational Neuroscience Laboratories, Kyoto, Japan. His research interests include humanoid robotics, cognitive systems, brain machine interfaces, biomimetic of human vision, human-robot interaction, active vision, and mobile robot navigation.

Queries

Q1. Author: Please provide the expansion of acronyms “STAMINA” and “ROSETTA.”
Q2. Author: Please provide page range in Refs. [3], [4], [19], [21], [27], and [29].
Q3. Author: Please provide month in Ref. [14].
Q4. Author: Please update Ref. [35].
Q5. Author: Please provide the year in which Emmanuel Dean-Leon became “Member” of the IEEE.
Q6. Author: Please provide the year in which Karinne Ramirez-Amaro became “Member” of the IEEE.
Q7. Author: Please provide the subject in which Florian Bergner working toward the Ph.D. degree.
Q8. Author: Please provide the year in which Florian Bergner became “Member” of the IEEE.
Q9. Author: Please provide the subject in which Ilya Dianov working toward the Ph.D. degree.
Q10. Author: Please provide the year in which Ilya Dianov became “Member” of the IEEE.
Q11. Author: Please provide the year in which Gordon Cheng became “Fellow” of the IEEE.