https://repository.uaeh.edu.mx/revistas/index.php/icbi/issue/archive
Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI

Pädi

Publicación Semestral Pädi Vol. 7 No. Especial (2019) 4–13

Robot Skin: Fully-Compliant Control Framework Using Multi-modal Tactile Events

Emmanuel Dean-Leon[*], Karinne Ramirez-Amaro[*], Florian Bergner[*], Gordon Cheng[*]

*Institute for Cognitive Systems (ICS), Technical University of Munich, Karlstraße 45/II, 80333, München, Germany.

Abstract

In this paper, we present a multi-modal control framework to provide compliant behaviors on industrial robots, even when the robots are position or velocity commanded. These robot behaviors enable the fast deployment of robots in industrial setups. This is obtained by fusing multi-modal sensor signals from robot skin with different control approaches. These compliant behaviors allow teaching robots safely. The presented framework can bridge kinesthetically demonstrated activities with low-level robot commands using state-of-the-art teaching by demonstration method based on a semantic engine. We validate our control framework in a real robot for an industrial scenario where our presented framework enables a stiff robotic system to be compliant, flexible, and adaptable to different working conditions, e.g., different end-effectors with multiple command interfaces (position/velocity and torque interfaces). The experimental validation shows that the integration of the different modules of our control framework simplifies the teaching process of robots.

Keywords: robot skin, compliant control, human-robot interaction.

1. Introduction

Fast and flexible reconfiguration control frameworks are a key component in increasing the usability of robots. Especially in industrial scenarios, such frameworks are expected to decrease the set-up time to reconfigure robotic systems and increase the safety for Human–Robot Interaction (HRI) (Andersen et al., 2014). A compelling method to increase the flexibility of robots is to combine them with human co-workers in close interaction. The fusion of the high adaptability of the human and the accuracy of a robot system can ease the automation of industrial processes. To this aim, it is required a robust control framework that supports dynamic surface-level tactile interaction, which can be used for kinesthetic teaching.

Safety in physical Human–Robot Interaction (pHRI) (Santis et al., 2008) is a fundamental aspect of developing robust control systems. Especially for the new way of teaching robot sequences using programming by demonstration methods which require physical interactions with the robot, where an expert robot programmer is not required. However, in general, standard robots are not compliant, which is an extremely important requirement for safe pHRI.

Robotic systems used in complex tasks require multiple kinematic structures of the arms, dexterous hands, or other different kinds of end-effectors. All these robot elements can also be equipped with multi-modal sensors to improve control mechanisms. For example, mounting force sensors and/or tactile sensors on a robot arm, also on the end-effector can increase the flexibility and safety of the manipulator (Luo and Chang, 2012). However, integrating all these components is not a trivial task, and in general, is customized for a specific robot. Furthermore, the integration of additional sensors impacts directly on the complexity of the control design, since we need to include different mechanisms to fuse all these different sensors. Moreover, the multi-modality is considered as a main and an unquestionable feature of human-robot interaction (Gorostiza et al., 2006). This reveals the need to develop multi-modal control frameworks that can be adaptable to different robots as well as end-effectors. Therefore, the development and integration of technologies such as control fusion, artificial skin, and multi-modal control design are needed to increase the standard capabilities of industrial robots, to improve the safe physical interaction with robots, and to simplify the robot programming.

[*]Autor en correspondencia.
Correos electrónicos: dean@tum.de (Emmanuel Dean-Leon), karinne.ramirez@tum.de (Karinne Ramirez-Amaro), florian.bergner@tum.de (Florian Bergner), gordon@tum.de (Gordon Cheng)
URL: www.ics.ei.tum.de (Emmanuel Dean-Leon), www.ics.ei.tum.de (Karinne Ramirez-Amaro), www.ics.ei.tum.de (Florian Bergner), www.ics.ei.tum.de (Gordon Cheng)

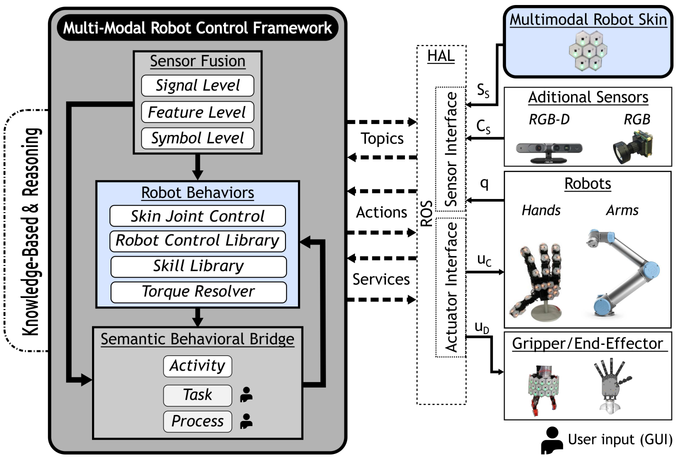

Figure 1: The system provides a general framework where different multi-modal controllers can be specified, fused and executed. Using a self-configurable artificial skin, we can re-configure the system to adapt the controllers to different robots. The framework combines semantic reasoning methods to bridge human behaviors with low-level control approaches.

One significant advantage of using multi-modal sensor fusion is to provide enhanced and complementary information during the parallel processing of data (Luo and Chang, 2012). However, many issues arise during this fusion. This includes adequate management, sensor synchronization and the necessity of *different levels of abstraction* to cope with signal uncertainty. These important aspects are addressed in this paper, using our proposed multi-modal control framework, see Fig. 1.

1.1. Related Work

The need for flexible and easily reconfigurable robots is discussed in (Pedersen et al., 2016), where the authors proposed a generic set of skills which can be combined with more complex robot tasks. This implies that the teaching process should be transformed into a high-level abstraction to allow generalization (Nicolescu and Mataric, 2003). Another important requirement for such systems is safety for collaborative industrial robots which is addressed in the new standard ISO 15066 (ISO/TS-15066, 2014).

Cyber-physical systems are used to develop novel robotic technologies to cope with larger variability on processes. For example, (Krüger et al., 2016) presented a cognitive system applied to a concrete and well-known use case from the automotive industry, which is part of a research project called STAMINA. For fast deployment of robotic technologies, the work of (Björkelund et al., 2012) proposes a "knowledge integration framework" which generates a generalized platform-independent description of a manufacturing process. The manufacturing process is modeled by abstract tasks which contain skills. The general description is robot independent such that tasks at the higher execution levels can always be executed in the same way. The work of (Hein et al., 2008) focuses on the simplification of programming industrial robots by combining on-line and off-line programming to a more intuitive and efficient assisted on-line programming technique. Instead of teach-pendants or joysticks, the authors propose an intuitive input device based on the optical tracking of a special marker tool. Moving the marker tool reflects movements of the end effector. The framework uses a modular approach in which algorithms (e.g. collision avoidance) and end-effector restrictions simplify the programming process. In a different form, Robot Programming using Augmented Reality (RPAR) has been proposed by (Pan et al., 2010) where the robot programming is more intuitive and thus more flexible for Small to Medium Enterprises (SMEs).

To consider safety, adaptability, and flexibility for robotic systems, in this paper, we present a new multi-modal control framework which considers these important requirements. The proposed framework is divided in different modules which are described in the following sections.

1.2. Framework Description

The hierarchical structure to define the workflow of our framework is divided into the following levels. The highest level is the *Process*, which is defined as the combination of sequential *Tasks*. *Tasks* are the combination of ordered *Activities*. *Activities* are semantic descriptions of *Skills*, and finally *Skills* (lowest level) represent control commands that robots need to execute, this includes multiple control fusion, see Fig. 2. This hierarchy interconnects the *Problem Space*[1] and the *Execution Space*[2].

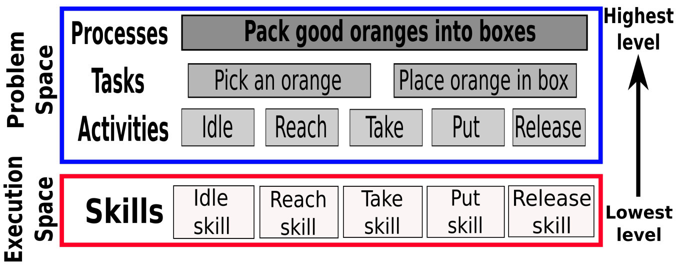

Figure 2: Hierarchical structure to define the workflow of our system. The Problem Space (marked as the blue-box) provides semantic descriptions which represents robot-agnostic knowledge. The Execution Space (red-box) is the specific information that depends on the current robot.

We introduce a new multi-modal control framework to enable fully-compliant behaviors for robots. Fig. 1 depicts our proposed framework which is defined by three principal modules. The first module handles the fusion of multiple sensors, see Sec. (2). The second module presents an adaptable robot control generator that supports dynamic surface-level tactile interaction which can be used for kinesthetics teaching, see Sec. (4). This module renders a multiple-control fusion system, a skill library which is connected directly to human activities (see Fig. 2) and a general control interface that allows driving robots with different command interfaces (i.e. Position, Velocity or Torque). The robot behavior control is adaptable to new robot systems, where the key component for this adaptation is our self

[1]Also known as the *Application Domain Space*, which is a *robot-agnostic* space.

[2]Also known as the *Solution Space*, which can be defined as *Hardware Space* and depends explicitly on the robot that executes the commands.

-organizing/-calibrating artificial skin (Mittendorfer and Cheng, 2011) that can be used to parametrize and model the target robots, see Sec. (3). The final module provides a semantic reasoning engine to integrate and translate human demonstrations to robot low-level commands, see Sec. (5), where the user can kinesthetically teach activities, which are automatically connected to robot primitives. This module is a bridge between the *Problem Space* and the *Execution Space*, see Fig. 2. This is an important link since the description of humans actions lie in the *Problem Space* while robot commands are defined in the *Execution Space*. A Hardware Abstraction Layer (HAL) provides interfaces to the sensors and actuators connected to the robot. Our complete framework has been implemented using the Robot Operating System (ROS) middleware which provides a convenient infrastructure to develop hardware interfaces easily. Furthermore, several robot manufacturers already provide interfaces for their robots in ROS[3]. This simplifies the integration of more robots to our framework[4].

1.3. Main contributions

The key components and contributions of our framework are:

1. A self-organizing and self-calibrating artificial skin which generates the required models to control the robot.
2. Integration of multi-modal skin sensor signals in the control loop of robots to enhance the safety and interaction during HRI.
3. The development of multi-modal control behaviors to allow safe physical human-robot interaction beyond the standard capabilities of standard robots.
4. A multilevel approach that is capable to automatically segment and recognize robot behaviors from kinesthetic demonstrations. This approach boosts the learning phase since the demonstrated nominal trajectories are automatically segmented and recognized, therefore no off-line annotation is needed, which is typically the case in most Programming by Demonstration (PbD) methods.

These modules are integrated into a framework, see Fig. 1, that allows the generation of new robot behaviors for standard robots[5].

The following sections are devoted to describing the different modules used in our framework.

2. Sensor Fusion Module

The first module in our framework merges the redundant information acquired by different sensors. It provides a more precise perception of the environment and better system monitoring. The multi-sensor integration and fusion are achieved in a three-level category. In each category, we define adequate algorithms to fuse information according to the processed data formats using different levels of abstraction, see Fig. 1.

2.1. Signal Level (low-level)

The first level is *Signal level (low-level)*. It represents the state of the system, where each sensor type has a different sampling rate. Therefore, an important step in this level is signal synchronization, filtering, and thresholding. Another important step is the implementation of event-based signaling to reduce the computational cost of processing the multiple sensor signals generated by our artificial robot skin. Using this event-based system the skin cells only forward tactile information whenever the local change of a sensor value exceeds a given threshold. This principle reduces the data load significantly to 20 % of the original data load and induces less computational costs which also improves the performance of the real-time control Bergner et al. (2015).

2.2. Feature Level (mid-level)

The second level *Feature Level (mid-level)* extracts features from input signals. In our case, these features are represented in two forms: image features and semantic atomic features. These last features provide the relationships between an agent and the objects in the environment, which are important to interpret the demonstrated behaviors in the robot Ramirez-Amaro et al. (2015a).

2.3. Symbol level (high-level)

The third level *Symbol level (high-level)* processes the features in symbolic representation, e.g. human descriptions. The principal outcome of the Sensor fusion model is two-folded. First, it provides meaningful information for the *Robot Behaviors Module*, see Sec. 4. Second, it extracts *symbolic properties* and connects the demonstrated *activities* with robot commands, explained in Section 5. This information is used to populate and generate the knowledge used by the *Semantic Behavioral Bridge Module*. This module transforms the low-level sensor signals into concepts (a semantic generalization of perceived events).

These three fusion levels provide additional robustness on the generated data, especially to deal with partial information and the multi-sampling nature of a complex robotic system Ramirez-Amaro et al. (2015b).

3. Robot Parametric Modeling

To generate adequate robot dynamic behaviors, we need to adapt our framework to different robot models. This includes robot Kinematic and Dynamic models. This can be a complex and tedious task. In our framework, we try to automate this process and reduce human intervention to the minimum. To this aim, we use the enhanced multi-modal signals of our artificial skin Mittendorfer and Cheng (2011), which is a modularized,

[3]http://rosindustrial.org/

[4]The only requirement is that the robot provides an external control interface, e.g. position, velocity or torque control. Currently, we are working on the deployment of our framework on the following robots: UR-10, PAL REEM-C and KUKA-LWR.

[5]The only requirement is that the robot provides an external control interface, e.g. position, velocity or torque control.

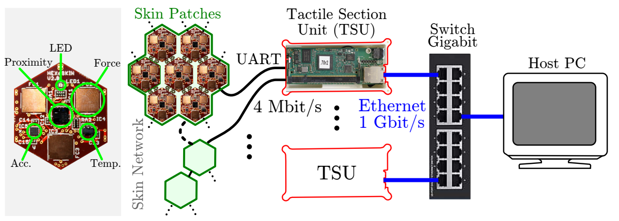

Figure 3: Artificial skin cells and its interface to the PC. Each *skin cell* has 4 different modalities: a)Proximity, b)Normal Force, c)Acceleration and d)Temperature.

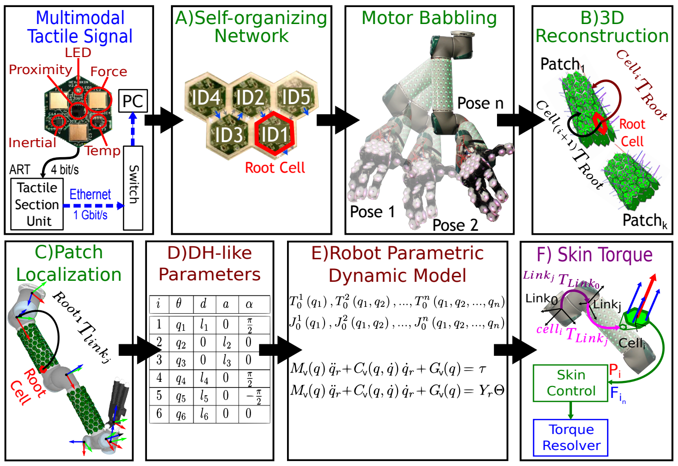

Figure 4: Block diagram of the five stages needed in the self-configuring, self-calibrating artificial skin and robot parametric modeling. A) Network exploration and configuration, B) Skin cells 3D reconstruction, C) Skin Patch calibration with respect to the robot links, D) Robot kinematic parametrization and E) Parametric modeling of the robot for control design and interface.F) Transforming tactile signals into torque information, see Sec. 4.1.

multi-modal robotic skin. The skin consists of hexagonally shaped *skin cells* (see Fig. 3). Each skin cell has the same set of sensors which transduce tactile information of different modalities, such as vibrations (acceleration sensor), pressure (capacitive force sensors), pre-touch (proximity sensor) and temperature (temperature sensor). Micro-controllers located on the back of each skin sample and filter the sensor data and pack the acquired information into skin cell data packets.

Neighboring skin cells are connected and exchange information with their neighbors. All connected skin cells realize a meshed and highly redundant skin cell network. These interconnected Skin cells shape an entity called *Skin Patch*. Skin patches are connected to interface boxes (see Fig. 3) which ease the communication between the skin cell network and the PC. We use the structural properties of our skin and its multi-modal signals to self-calibrate and model the robot system. This modeling process is divided into five stages, depicted in Fig. 4.

3.1. Self-organizing Network

The skin cell network is based on a module-to-module communication with active data system. This active system allows multiple connections with the Personal Computer (PC) increasing the bandwidth and redundancy. The first step is to assign a unique identifier (ID) to each cell in a patch. The second step is to build a skin graph to identify the neighbors of each cell, and set the *Root Cell* (see Fig. 4). The last step is to build a network based on the functional ports. All the steps are executed with an algorithm embedded in the Micro-controller on the skin cell Mittendorfer and Cheng (2011).

3.2. 3D Reconstruction

This stage uses the information from the network organization and exploits the sensor information on each cell. In particular, we use the IMU information to measure the gravity vectors. To avoid singularities in the reconstruction, we need to collect sensor data in multiple robot poses, see Fig. 4. Using this information and the structural information of the cells, we can estimate the relative pose (position and orientation) ${}^{Cell_i}T_{Root_k}$ of each cell with respect to the *Root Cell* Mittendorfer and Cheng (2012).

3.3. Patch Localization

Each skin patch is rigidly attached to a specific robot link. Therefore, the next stage is to compute the relative transformation between the *Root Cell* and its robot link, namely ${}^{Root_k}T_{Link_j}$. This stage can be done using an external sensor (RGB camera) and the Light-Emitting Diodes (LEDs) embedded in each skin cell Mittendorfer et al. (2014a), or manually with the aid of a 3D visualization in rviz and an interactive marker, see Fig. 5. From this transformation, we can easily compute the relative pose between each skin cell and its corresponding link as:

$$ {}^{Cell_i}T_{Link_j} = {}^{Root_k}\,T_{Link_j}\left({}^{Cell_i}T_{Root_k}\right). \qquad (1)$$

3.4. DH-like parametrization

The localization of each skin cell with respect to a robot body part can be used to compute kinematic parameters of the robot. In this case, we use the multi-modal sensor information of each cell to determine the axis of motion of each joint and its associated offsets (*radius*). This process is generated in 3 steps; a) Estimating the Joint Axis Unitary Vector based on a Inertial Measurement Unit (IMU) information on each cell while the robot performs joint-wise motions. b) Extracting the tangential component of the acceleration during joint motion. In this step, we subtract the gravity vector from the measurements and compute the tangential acceleration using Singular Value Decomposition (SVD). c) Computing the Radial Distance. This step is performed using the amplitude of the tangential component and fitting a least-squares linear model Mittendorfer et al. (2014b). The obtained parameters can be described in the form of DH-parameters, see Fig. 4.

3.5. Parametric Dynamic Model

The last stage is to use the Denavit-Hartenberg (DH)-like parameters to compute the robot Kinematic and Dynamic models. This process is based on iterative Euler-Lagrange modeling which is divided into the following steps: a) Compute the relative link transformations ${}^{Link_j}T_{Link_{j-1}}$ using the symbolic DH-table. b) Compute the global transformation of each link with respect to the robot base:

$$ {}^{Link_j}T_{Link_0} = {}^{Link_{j-1}}\,T_{Link_0}\left({}^{Link_j}T_{Link_{j-1}}\right) \qquad (2)$$

with $j = 2, 3, ..., n$ where n is the robot Degrees of Freedom (DOF). Using the transformations in eq. (2), we can compute the global pose of each skin cell with respect to the robot base $Link_0$ as:

$$^{Cell_i}T_{Link_0} = ^{Link_j}T_{Link_0}\left(^{Cell_i}T_{Link_j}\right). \tag{3}$$

c) Extract the $^{Link_j}z_{Link_0}$ axis and the position vector of each joint $^{Link_j}t_{Link_0} \in \mathbb{R}^3$ from the homogeneous matrices eq. (2). d) Compute the geometric Jacobian of each joint $^{Link_j}J_{Link_0} \in \mathbb{R}^{6\times n}$. Finally, use these Jacobians to compute the Mass and Inertia, Centripetal and Coriolis, and Gravitational symbolic matrices, $M(q), C(q, \dot{q}) \in \mathbb{R}^{n\times n}$ and $G(q) \in \mathbb{R}^n$, respectively. These matrices are obtained using the definitions of the Kinetic and Potential energy in the Euler-Lagrange formulation. Dean-Leon et al. (2012). The robot kinematic models together with the global calibration of the skin cell eq. (2)-(3) can be used to compute the geometric Jacobian of each skin cell, named $^{Cell_i}J_{Link_0} \in \mathbb{R}^{6\times n}$.

All the above stages are performed off-line and only once per robot. This complete process has multiple outcomes: 1) Skin cells calibration (relative pose of each cell with respect to a robot link and the base link), 2) Robot Kinematic parameters and models and 3) Parametric Dynamic model. These models and parameters are exploited to design controllers and control interfaces in the following sections.

4. Robot Behaviors Module

This module receives information from the *Sensor Fusion Module* and processes the skills requested by the *Semantic Behavioral Bridge*. This module provides a *Skin Control* to integrate the interactions with the environment perceived by the skin in the main control loop of the robot. It also provides a general *Robot Control Library* which is used to compose different skills available in the *Skill Library*. The *Robot Control Library* maps the requested skills into motor commands for the robot. Depending on the command interface specified by the robot, the *Torque Resolver* transforms these motor commands into a specific interface type. The following subsections further elaborate these components.

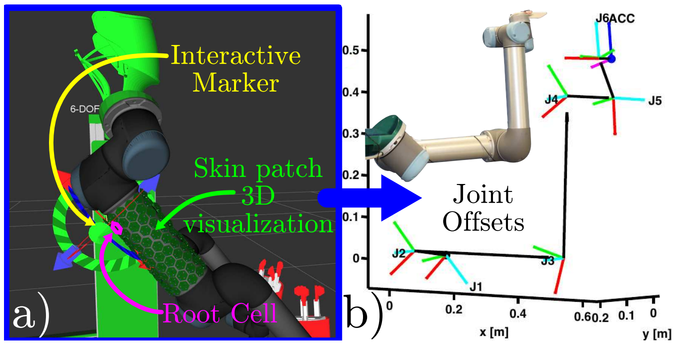

Figure 5: Interactive skin patch calibration. The user can set the relative pose of the *Root Cell* ($^{Root_k}T_{Link_j}$) using the interactive marker.

4.1. Skin Joint Control

To fuse the information from the artificial skin sensors with the different controllers available in the *Robot Control Library* (see Fig. 6), we need to transform the sensors signals (e.g. pre-touch and pressure) to generalized force commands. In this work, we use force vectors to transform tactile signals into joint torque signals. This is achieved with the following two steps.

4.1.1. Multi-modal Tactile Signals to Force Vector

Each $Cell_i$ produces a set of three *pressure signals* $f_{i_m} \in \mathbb{R}, m = 1, 2, 3$ and a single *proximity signal* $p_i \in \mathbb{R}$, see Fig. 6. The first step is to transform these signals into force vectors. By design both the pressure-signals and the proximity signal are normal to the sensor $Cell_i$, defined by its $z-axis$. Therefore the $Cell_i$ force vectors can be constructed as follows:

$$\overline{P_i} = [0, 0, w_p p_i]^T,\ \overline{F_i} = [0, 0, w_f \sum_{m=1}^{3} f_{i_m}]^T, \tag{4}$$

where $w_p, w_f \in \mathbb{R}$ are weighting gains for the proximity and pressure signals, respectively. The above equations represent the force vectors of each signal with respect to the $Cell_i$ frame, see Fig. 6. The force vector with respect to the robot base ($Link_0$) is obtained as:

$$^{Cell_i}F_{Link_0} = ^{Cell_i}R_{Link_0}\left(\overline{F_i} + \overline{P_i}\right) \tag{5}$$

where $^{Cell_i}F_{Link_0} \in \mathbb{R}^3$ represents the total virtual force vector produced by the tactile signals of the $Cell_i$. The rotation matrix $^{Cell_i}R_{Link_0} \in SO(3)$ is extracted from the homogeneous transformation $^{Cell_i}T_{Link_0}$, computed with eq. (3).

4.1.2. Force Vector to Joint Torques

In the second step, the torque $\tau_{Cell_i} \in \mathbb{R}^n$ produced by the tactile signals of each $Cell_i$ is calculated as:

$$\tau_{Cell_i} = ^{Cell_i}J^T_{Link_0}\left(^{Cell_i}W_{Link_0}\right) \in \mathbb{R}^n \tag{6}$$

where $^{Cell_i}W_{Link_0} = [^{Cell_i}F^T_{Link_0}, 0^{1\times 3}]^T \in \mathbb{R}^6$ is the wrench applied on $Cell_i$[6]. The skin joint torque $\tau_{skin} \in \mathbb{R}^n$ generated by all the *skin cells* on every patch k is computed as:

$$\tau_{skin} = \sum_{k=1}^{p}\sum_{i=1}^{s}\tau_{Cell_{k,i}} \in \mathbb{R}^n \tag{7}$$

with s as the number of skin cells in a skin patch and p is the total number of skin patches in the robot. The joint torque obtained from eq. (7) is fused with other different controls in the *Robot Control Library* to produce specific robot behaviors.

[6] We set the moment on $Cell_i = 0 \in \mathbb{R}^{3\times 1}$ since it is physically impossible to apply pure moment to an individual $Cell_i$ with respect to its own reference frame, or even measure it with the skin sensors.

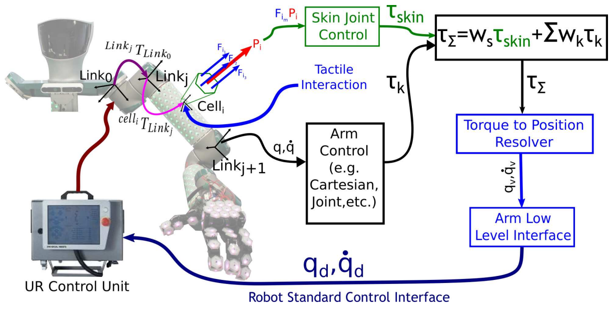

Figure 6: Transforming skin signals into control signals.

4.2. *Robot Control Library*

In this work, we define *Skill* as a composition of different controllers to achieve a specific *Activity*. This composition is obtained by fusing multiple controllers into a single control command for the robot. The parameters needed for the controllers are generated by the *Semantic Behavioral Bridge Module*, e.g. desired joint position or contact force, and they are defined on-line according to the current *Task*. In our system, these are the currently implemented controllers.

4.2.1. *Joint Control*

Based on Parra-Vega (2001), we implemented a second order sliding mode torque controller (τ) in joint space defined by the following control law:

$$\tau = -K_d S_q + Y_r \Theta \in \mathbb{R}^n \tag{8}$$

where $K_{d_+} = K_{d_+}^T \in \mathbb{R}^n$ and $Y_r\Theta \in \mathbb{R}^n$ is the robot regressor, which is used to compensate for the internal forces of the robot system. The joint error surface S_q is defined as:

$$S_q = \dot{q} - \dot{q}_r \in \mathbb{R}^n \tag{9}$$

where $\dot{q}$ is the joint velocity and the joint velocity reference $\dot{q}_r \in \mathbb{R}^n$ is given as

$$\dot{q}_r = \dot{q}_d - K_p(t)\Delta q + S_d - K_{i_1}\int_{t_0}^{t} S_\delta(\zeta)\, d\zeta - K_{i_2}\int_{t_0}^{t} \tanh(\mu S_\delta(\zeta))\, d\zeta \tag{10}$$

with $\tanh(\mu\bullet)$ as a smooth approximation for the function $sign(\bullet)$, and $\mu > 0 \in \mathbb{R}$. $K_{i_j} > 0 \in \mathbb{R}^{n\times n}$, $j = 1, 2$, and $K_p(t)$ is a time-varying proportional gain given by:

$$K_p(t) = \frac{(1+\epsilon)\dot{\xi}(t)}{(1-\xi(t))+\delta} \tag{11}$$

where ϵ and δ are positive small scalars, the function $\xi(t) \in \mathbb{C}^2$ is defined as a spline-function satisfying the following constraints; $\xi(t_0) = \dot{\xi}(t_b) = 0$, with $t_b > 0$ as the convergence time. The joint error manifold S_δ is

$$S_\delta = S - S_d = \left(\Delta\dot{q} + K_p(t)\Delta q\right) - \left(S(t_0)e^{-\kappa t}\right) \tag{12}$$

with the joint position error defined as:

$$\Delta q = q - q_d \in \mathbb{R}^n \tag{13}$$

where $\Delta\dot{q} = \dot{q} - \dot{q}_d$ is the joint velocity error. Using the above definition of S_δ, the joint error surface S_q (eq. 9) is finally defined as:

$$S_q = S_\delta + K_{i_1}\int_{t_0}^{t} S_\delta(\zeta)\, d\zeta + K_{i_2}\int_{t_0}^{t} sign(S_\delta(\zeta))\, d\zeta \tag{14}$$

which is a prototypical function of a second-order sliding mode function[7]. Furthermore, S_d and $K_p(t)$ play an important role to guarantee the stability of the system, especially when two antagonistic controllers are acting, e.g. Joint/Operational Control and Skin Joint Control. The control framework detects this condition and resets the bias function (S_d) and the proportional gain (K_p) to avoid overshooting behaviors.

4.2.2. *Operational Control*

For the operational space, we extended the Joint Control from eq. (8), defining the joint velocity reference (eq. 10) as:

$$\dot{q}_r = J_a(q)^{-1}\dot{x}_{ef_r} \tag{15}$$

where $J_a(q)$ is the analytical Jacobian defined as:

$$J_a(q) = \begin{bmatrix} I^{3\times3} & 0^{3\times3} \\ 0^{3\times3} & B(\varphi,\phi)^{-1} \end{bmatrix} J(q). \tag{16}$$

$B(\varphi,\theta)$ is a transformation matrix to represent angular velocities as function of Euler angles (φ,θ,ψ) $[Z-Y-Z]$, given by:

$$B(\varphi,\theta) = \begin{bmatrix} cos(\varphi)\,sin(\theta) & -sin(\varphi) & 0 \\ sin(\varphi)\,sin(\theta) & cos(\varphi) & 0 \\ cos(\theta) & 0 & 1 \end{bmatrix}. \tag{17}$$

In eq. (15), we have defined a Cartesian velocity reference as:

$$\dot{x}_{ef_r} = \dot{x}_{ef_d} - K_p(t)\Delta x + S_{x_d} - K_{i_1}\int_{t_0}^{t} S_{x_\delta}(\zeta)\, d\zeta - K_{i_2}\int_{t_0}^{t} \tanh\left(\mu S_{x_d}(\zeta)\right) d\zeta \in \mathbb{R}^6 \tag{18}$$

which generates the following Cartesian error manifold:

$$S_{x_\delta} = S_x - S_{x_d} = \left(\Delta\dot{x} + K_p(t)\Delta x\right) - \left(S_x(t_0)e^{-\kappa t}\right) \tag{19}$$

where the Cartesian position and velocity errors are defined with:

$$\Delta x = x_{ef} - x_{ef_d} \in \mathbb{R}^6 \tag{20}$$

$$\Delta\dot{x} = \dot{x}_{ef} - \dot{x}_{ef_d} \in \mathbb{R}^6 \tag{21}$$

where x_{ef}, x_{ef_d} are the end-effector position and desired position defined in the robot base.

4.2.3. *G Control*

This control is a *Gravity Compensation* control, defined as $\tau = G(q) \in \mathbb{R}^n$, where $G(q)$ is the robot's gravitational torques vector.

[7] $\tanh(\mu\bullet) \approx sign(\bullet)$

4.2.4. Spline Joint Control

For this controller, we use the same definition for the joint velocity reference defined in eq. (10), where the desired joint position and velocity vectors $q_d, \dot{q}_d$ on eq. (13) are defined as time-varying trajectories which can be generated on-line. However, for this controller, the desired joint position is a static goal $q_g \in \mathbb{R}^n$. In this case, the controller generates a smooth trajectory from the initial joint position $q_i \in \mathbb{R}^n$ to the target one q_g. The trajectory generator is defined as:

$$q_d = a_1 \left(q_g - q_i\right) + q_i, \; \dot{q}_d = a_2 \left(q_g - q_i\right) \tag{22}$$

The coefficients a_i are defined as: $a_1 = 10r^3 - 15r^4 + 6r^5$, $a_2 = (30r^2 - 60r^3 + 30r^4)/(t_f)$, where $r = t/t_f$ is a time ratio between the current time t and the desired total time t_f. This function guarantees a smooth trajectory that satisfies the constraints $q_d(0) = q_i$, $q_d\left(t_f\right) = q_g$ and $\dot{q}(0) = \dot{q}\left(t_f\right) = \ddot{q}(0) = \ddot{q}\left(t_f\right) = 0 \in \mathbb{R}^n$.

4.2.5. Spline Operational Control

This controller uses the Cartesian velocity reference in eq. (18) where the desired end-effector position and velocity $x_{ef_d}, \dot{x}_{ef_d}$ are generated using eq. (22), with x_{ef_i}, x_{ef_g} instead of q_i, q_g. The controllers *Joint Control* and *Operational Control* can be used to generate reactive trajectories, where the desired Joint/Cartesian position and velocity are defined using the input of external sensors. The last two controllers are used to define static goal positions.

The current *Robot Control Library* contains standard controllers. However, the choice of joint torque as the common control output allows the implementation of more sophisticated controllers, e.g. Constrained Cartesian Control Garcia-Valdovinos et al. (2005) or Image-Based Visual Servoing Dean-León and Cheng (2014).

4.3. Control Fusion

The mapping from tactile signals into joint torques (eq. (6)-(7)) allows the fusion of multiple controls that use the same generalized force representation, see Section 4.2. Thus, a simple normalized weighted-sum approach to adding the contribution of each controller to a total joint torque output τ_Σ can be used:

$$\tau_\Sigma = w_s \tau_{skin} + \sum\nolimits_{k=1}^{u} w_k \tau_k, \tag{23}$$

where u is the number of controllers and $w_s, w_k \in \mathbb{R}$ are weighting values. τ_k is the control output of a controller defined by the user. We selected this fusion method to guarantee a deterministic behavior, even when local minima are present. Nevertheless, a more sophisticated approach can be used to select an optimal combination of controls, e.g. Aertbeliën and Schutter (2014). The weight selection for the controllers depends on the specific robot behavior that we need to generate. Some examples of the different robot behaviors are depicted in Fig. 7. These robot behaviors are triggered by the *Skill Library*. The controller Skin Joint Control has special importance since can induce compliance in a standard robot.

	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						✓	✓

Figure 7: Robot Behaviors and their relation with the controllers.

4.4. Skill Library

This library provides a compendium of skills, where the user can specify the type of controllers that a skill requires for its execution. This library is the interface between the *Semantic Behavioral Bridge Module* (Sec. 5), and the low-level control (Sec. 4). Figure 7 shows some examples of skills currently available in our framework and their corresponding controllers. The Skill library provides a recipe for the management of low-level controllers. It provides a list of controllers that are loaded during execution time and which controllers need to be unloaded when a transition from one skill to another is requested. For example, the Skill *Reach Cartesian Goal* loads *Spline Operational Control*, *Skin Joint Control* and *G Control*. These controllers are fused using eq. (23).

In the *Robot Behaviors Module*, we provide two control classifications: *Continuous Controls* U_C and *Discrete Controls* U_D, see Fig. 1. The former are the type of controllers that generates a closed-loop continuous signal. The controllers introduced in Sec. 4.1 and Sec. 4.2 fall in this category. On the other hand, the U_D controllers trigger binary states, e.g. a gripper state *open/close*.

4.5. Torque Resolver

The Robot Control Framework depicted in Fig. 6, is designed to provide two low-level control interfaces, either *Position/Velocity* interface, available in most of the modern industrial robots, or *Torque* interface. In the case of *Torque* interface, we command directly τ_Σ to the control unit of the robot. To control robots with *Position/Velocity* interface, we need to transform the total commanded joint control τ_Σ into desired joint positions/velocities. To this aim, we have implemented a torque resolver which uses the dynamic state of a nonlinear observer to generate the desired joint commands. We obtain the full dynamic model to design the observer using the kinematic models of the robot in combination with the parametric dynamic model obtained in Sec. 3. This parametric model allows to specify user-defined dynamic behaviors, e.g. it can increase the viscous friction, thus generating a slower step response to an external input (e.g. tactile interaction). The desired joint positions/velocities $(q_d, \dot{q}_d)$ generated by the torque resolver are sent to the robot using its standard control interface, see Fig. 6.

5. Semantic Behavioral Bridge

The fusion of controls from the *Robot Behaviors Module* allows a safe physical interaction with any robot, even when they are not explicitly compliant by design. This represents a major advantage to teach robots new skills using kinesthetic demonstrations, especially when the demonstrator is not a robotics expert. We have developed this module to make a bridge between the demonstrations and the low-level robot control module. Thus, making this complex and not-intuitive connection transparent to the user. With this module, the user only needs to specify the desired *Activities* (via Kinesthetic Teaching) and this module automatically will segment and recognize the taught activities and transform them into robot skills. The *Skill Library* (Sec. 4.4) directly transforms the commanded skill into low-level controls and fuse them with the execution parameters obtained from the information generated by the multi-modal sensor fusion.

We employ a state-of-the-art hierarchical reasoning approach which consists of two levels Ramirez-Amaro et al. (2015a). In the first level, we determine the current state of the system by extracting the most relevant information from different sensors, such as cameras, multi-modal skin, and robot joint information. Then in the second level, we extract symbolic relationships to infer the demonstrated activities. This module segments and recognizes the demonstrated activities by physically interacting with the robot. These activities are directly mapped to specific skills from the *Skill Library*. For example, when the system infers the demonstrated *Reaching* an *Object* activity, then the Skill *Reach Cart Goal* (see Fig. 7) is executed. This will trigger the controllers: *Spline Operational Control*, *Skin Joint Control*, and *G Control*. The target position for the controllers will be defined on-line by the position of the detected *Object*. More details on the activity mapping can be found in Dean et al. (2016).

6. Demonstration Scenario – Packing Fruits

The main purpose of our framework is to enhance the safety, adaptability, and flexibility of industrial robots. Therefore, we selected a demonstration scenario where the different modules of our framework simplify the teaching and execution processes. The selected scenario is the sorting fruits process. In this scenario, we show the benefits of using the tactile and proximity sensors of the artificial skin to sense the quality of the products, in this case, oranges. Moreover, in this scenario, the human is teaching the robot the intermediate activities required to sort and pack oranges into boxes when the oranges are good (rigid), or the oranges will be thrown to the trash container when the oranges are bad (soft).

We evaluate our Multi-level control framework in our robotic platform Tactile Omni-directional Mobile Manipulator (TOMM). TOMM is composed of two industrial robot arms (UR-5[8]) covered with our artificial skin. The UR-5 robots are controlled using Position/Velocity command interface. As end-effector we tested two different options a) two Lacquey grippers[9] and b) two Allegro Hands[10], both covered with our artificial skin. These two types of end-effector differ both in its construction and its control interface. On one hand, the Laquey grippers are controlled using a *Discrete Control* where we can only command the binary states open/close. On the other hand, the Allegro Hands are more complex robotic systems which use a *Continuous Control* and their command interface is via joint torques. Therefore, we tested our framework with a hybrid system with Position/Velocity interface and Torque interface at the same time. This makes the robot more flexible and adaptable. The robot has as additional sensors two RGB cameras and one RGB-D camera used to obtain the 3D position of the objects.

Fig. 8 depicts the execution of different activities by the robot. Column *a* illustrates the kinesthetic teaching exploiting the full-compliance behavior generated by our framework. Notice that the robot UR-5 is not compliant by design, nevertheless with our framework, we can change the robot dynamic behavior. The activities in these experiments are: *Reaching* (b_1-c_1), *Putting* (b_2-c_2) and *Squeezing* (b_3-c_3). The loaded controllers are:

Reaching= (Skill:ReachCartGoal)[SplineOpCtrl,SkinJointCtrl, GCtrl]

Putting= (Skill:ReachCartGoal)[SplineOpCtrl,SkinJointCtrl, GCtrl]

Squeezing= (Skill:Close=>ForceCart)[CloseCtrl(D)] =>[OpCartCtrl, SkinJointCtrl].

Column *d* shows the fully-compliance behavior of the robot. During the execution of the activities, the user can always interact with the robot and the robot will react to these tactile interactions (contact and pre-contact). In this case, the robot will avoid collision with the user and self-collisions with the environment. The pre-contact information enhances the sensitivity of the robot, then the robot can react even when a feather is in close proximity with the skin on the robot, this increases the safety of the robot, see Fig. 8 d_2. Finally, column *e* shows different grasping types using the skin on the Allegro Hands controlled by our framework. Our framework can control robots with different kinematics, dynamics and control interfaces.

7. Conclusions

In this paper, we presented a multi-modal control framework which allows the generation of compliant robot behaviors for standard robots, even when the robot is not explicitly designed for compliant tasks. The only requirement to include new robots in our framework is that the robot provides any external control interface, e.g. position, velocity or torque control. The designed controllers take advantage of the self-organization and self-calibration features of our artificial skin to generate a parametric model of the controlled robots. These models are used to design specific controllers for those robots. The presented framework integrates multi-modal skin sensor signals in the control loop of robots to enhance the safety and

[8]http://www.universal-robots.com/products/ur5-robot

[9]http://lacquey.nl

[10]http://www.simlab.co.kr/Allegro-Hand.htm

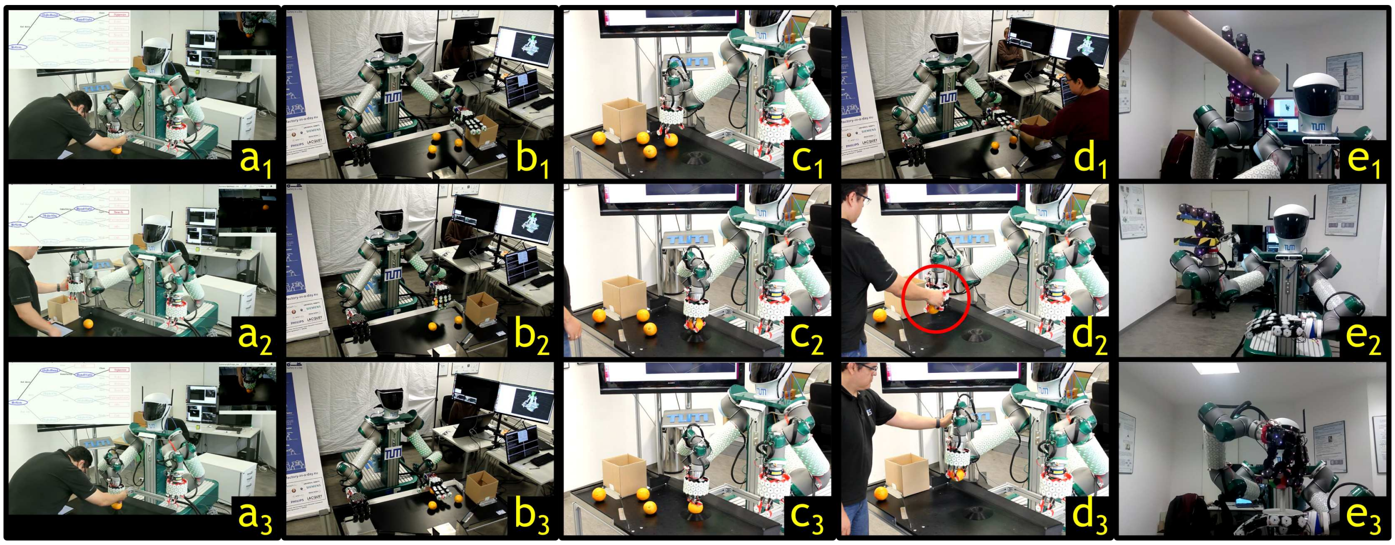

Figure 8: Teaching and execution of *Tasks* for the scenario of packing oranges. a_1)-a_3) Show the kinesthetic demonstration using the generated full-compliance on the robot arm and end-effector. b_1)-b_3) Execution of the learned Activities with the Allegro Hand as end-effector. c_1)-c_3) Similar to b, in this case, the end-effector is the Laquey gripper. d_1)-d_3) Full-compliance behavior generated by the control framework. e_1)-e_3) Allegro Hand executing different grasping tasks with different objects. The parameters for the controllers are the position of the *oranges*, *box* and *squeeze area*.

interaction during HRI. This allows teaching robots new activities via kinesthetic demonstrations, enhancing the intuitiveness of robot systems. We tested our system in a complex robot, demonstrating the adaptability of our framework to robots with different kinematics, dynamics and control interfaces.

Resumen en Español

Aspecto del robot: marco de control totalmente compatible con eventos táctiles multimodales.

Resumen

En este documento, presentamos un marco de control multimodal para proporcionar comportamientos compatibles con los robots industriales, incluso cuando se les ordena la posición o la velocidad. Estos comportamientos de los robots permiten el despliegue rápido de robots en configuraciones industriales. Esto se obtiene fusionando las sñales del sensor multimodal de la piel del robot con diferentes enfoques de control. Estos comportamientos compatibles permiten enseñar a los robots de forma segura. El marco presentado puede unir las actividades demostradas kinestésicamente con comandos de robot de bajo nivel utilizando la enseñanza más avanzada mediante un método de demostración basado en un motor semántico. Validamos nuestro marco de control en un robot real para un escenario industrial donde nuestro marco presentado permite que un sistema robótico rígido sea compatible, flexible y adaptable a diferentes condiciones de trabajo, por ejemplo, diferentes efectores finales con múltiples interfaces de comando (posición/velocidad e interfaces de par). La validación experimental muestra que la integración de los diferentes módulos en el marco de control simplifica el proceso de enseñanza de los robots.

Palabras Claves:

Aspecto del robot, control conforme, interacción humano-robot.

References

Aertbeliën, E., Schutter, J. D., Sept 2014. eTaSL/eTC: A constraint-based task specification language and robot controller using expression graphs. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 1540–1546.
DOI: `10.1109/IROS.2014.6942760`

Andersen, R. H., Solund, T., Hallam, J., June 2014. Definition and Initial Case-Based Evaluation of Hardware-Independent Robot Skills for Industrial Robotic Co-Workers. In: ISR/Robotik 2014; 41st International Symposium on Robotics. pp. 1–7.

Bergner, F., Mittendorfer, P., Dean-Leon, E. C., Cheng, G., 2015. Event-based signaling for reducing required data rates and processing power in a large-scale artificial robotic skin. In: IROS. IEEE, pp. 2124–2129.

Björkelund, A., Bruyninckx, H., Malec, J., Nilsson, K., Nugues, P., 2012. Knowledge for Intelligent Industrial Robots. In: AAAI Spring Symposium: Designing Intelligent Robots. Vol. SS-12-02 of AAAI Technical Report. AAAI.

Dean, E., Ramirez-Amaro, K., Bergner, F., Dianov, I., Lanillos, P., Cheng, G., October 2016. Robotic technologies for fast deployment of industrial robot systems. In: 42nd IEEE Industrial Electronics Conference (IEEE IECON2016). [Accepted]. IEEE.

Dean-León, E., Cheng, G., Nov 2014. A new method for solving 6D Image-Based Visual Servoing with virtual composite camera model. In: 2014 IEEE-RAS Int. Conf. on Humanoid Robots. pp. 519–525.

Dean-Leon, E., Nair, S., Knoll, A., Dec 2012. User friendly Matlab-toolbox for symbolic robot dynamic modeling used for control design. In: Robotics and Biomimetics (ROBIO), 2012 IEEE International Conference on. pp. 2181–2188.

Garcia-Valdovinos, L. G., Parra-Vega, V., Mendez-Iglesias, J. A., Arteaga, M. A., Nov 2005. Cartesian sliding PID force/position control for transparent bilateral teleoperation. In: 31st Annual Conference of IEEE Industrial Electronics Society, 2005. IECON 2005.

Gorostiza, J. F., Barber, R., Khamis, A. M., Malfaz, M., Pacheco, R., Rivas, R., Corrales, A., Delgado, E., Salichs, M. A., Sept 2006. Multimodal human-robot interaction framework for a personal robot. In: ROMAN 2006 - The

15th IEEE International Symposium on Robot and Human Interactive Communication. pp. 39–44.
DOI: 10.1109/ROMAN.2006.314392

Hein, B., Hensel, M., Wörn, H., 2008. Intuitive and model-based on-line programming of industrial robots: A modular on-line programming environment. In: ICRA. IEEE, pp. 3952–3957.

ISO/TS-15066, 2014. ISO/TS15066, Safety for collaborative industrial robots. Technical Standard ISO 15066.

Krüger, V., Chazoule, A., Crosby, M., Lasnier, A., Pedersen, M. R., Rovida, F., Nalpantidis, L., Petrick, R. P. A., Toscano, C., Veiga, G., 2016. A Vertical and Cyber-Physical Integration of Cognitive Robots in Manufacturing. Proceedings of the IEEE 104 (5), 1114–1127.

Luo, R. C., Chang, C.-C., 2012. Multisensor Fusion and Integration: A Review on Approaches and Its Applications in Mechatronics. IEEE Trans. Industrial Informatics 8 (1), 49–60.

Mittendorfer, P., Cheng, G., 2011. Humanoid Multimodal Tactile-Sensing Modules. IEEE Trans. Robotics 27 (3), 401–410.

Mittendorfer, P., Cheng, G., 2012. 3D surface reconstruction for robotic body parts with artificial skins. In: IROS. IEEE, pp. 4505–4510.

Mittendorfer, P., Dean, E., Cheng, G., Sept 2014a. 3D spatial self-organization of a modular artificial skin. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 3969–3974.
DOI: 10.1109/IROS.2014.6943120

Mittendorfer, P., Dean, E., Cheng, G., Nov 2014b. Automatic robot kinematic modeling with a modular artificial skin. In: 2014 IEEE-RAS International Conference on Humanoid Robots. pp. 749–754.
DOI: 10.1109/HUMANOIDS.2014.7041447

Nicolescu, M. N., Mataric, M. J., 2003. Natural methods for robot task learning: Instructive demonstrations, generalization and practice. In: Proceedings of the second international joint conference on Autonomous agents and multiagent systems. ACM, pp. 241–248.

Pan, Z., Polden, J., Larkin, N., van Duin, S., Norrish, J., 2010. Recent Progress on Programming Methods for Industrial Robots. In: ISR/ROBOTIK. VDE Verlag, pp. 1–8.

Parra-Vega, V., 2001. Chattering-free dynamical tbg adaptive sliding mode control of robot arms with dynamic friction for tracking in finite-time. In: Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on. Vol. 4. pp. 3471–3476 vol.4.
DOI: 10.1109/ROBOT.2001.933155

Pedersen, M. R., Nalpantidis, L., Andersen, R. S., Schou, C., Bøgh, S., Krüger, V., Madsen, O., 2016. Robot skills for manufacturing: From concept to industrial deployment. Robotics and Computer-Integrated Manufacturing 37, 282–291.

Ramirez-Amaro, K., Beetz, M., Cheng, G., 2015a. Transferring skills to humanoid robots by extracting semantic representations from observations of human activities. Artificial Intelligence.

Ramirez-Amaro, K., Dean-Leon, E. C., Cheng, G., 2015b. Robust semantic representations for inferring human co-manipulation activities even with different demonstration styles. In: Humanoids. IEEE, pp. 1141–1146.

Santis, A. D., Siciliano, B., Luca, A. D., Bicchi, A., 2008. An atlas of physical human–robot interaction. Mechanism and Machine Theory 43 (3), 253–270.