2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Model-Based Specification of Control Architectures for Compliant Interaction with the Environment

Dennis Leroy Wigand[1], Niels Dehio[2] and Sebastian Wrede[1]

***Abstract*— In recent years the need for manipulation tasks in the industrial as well as in the service robotics domain that require compliant interaction with the environment rose. Since then, an increased number of publications use a model-driven approach to describe these tasks. High-level tasks and sequences of skills are coordinated to achieve a desired motion for e.g., screwing, polishing, or snap mounting [1], [2]. Even though the awareness of the environment, especially in terms of contact situations, is essential for successful task execution, it is too often neglected or considered insufficiently.**

In this paper, we present a model-based approach, using domain-specific languages (DSL), that enables the explicit modeling of the environment in terms of contact situations. Decoupling the environment model from the skills, fosters exchangeability and allows the adaptation to different environmental situations. This way, an explicit but non-invasive link is established to the skills, enabling the environment model to provide a context to constrain the execution of the skills. Further, we present a synthesis from the modeled contact situations to a real-time component-based control architecture, which executes the skills subject to the active environmental context. A dual arm yoga mat rolling task is used to show the impact of the environment model on the skill execution.

I. INTRODUCTION

Already quite early in the evolution of robotics control, the awareness for the explicit modeling of the involved aspects and its requirements rose [3], [4], [5]. Abstracting from the mathematical equations, which hold a high amount of knowledge in a strongly condensed manner, makes such aspects comparable and more easily understandable. Already in 1981, Mason [6] recognized the problem that at that time it was common practice that the abstractions—mostly in form of control block diagrams—had no technical link to the equations or even to the implementation. Therefore, he introduced the *Task Frame Formalism* (TFF) as an interface to abstract from the low-level control. Since without such a link, there is no traceability. Thus, inconsistency and divergence between abstraction and implementation is almost impossible to prevent. Model-driven engineering (MDE) provides a solution for increased comparability and understandability as well as to reduce or even close the mentioned gap by establishing a link to the implementation via model transformation and code generation. Since 1990, MDE especially in form of domain-specific languages (DSL) experienced a steady growth in the field of robotics [3]. During that time,

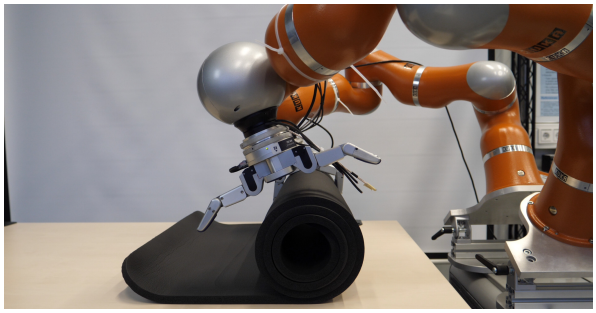

Fig. 1. Compliant rolling of a deformable yoga mat with two KUKA LWR IV+, executing the synthesized control architecture. The experiment can be seen in the provided video.

Hasegawa et al. [4] encouraged the modeling of high-level robotic tasks in combination with the explicit modeling of the environment. Nowadays, DSLs are frequently used in various robotic (sub-)domains [3], such as robot control [7], behavior or motion planning [8], and component-based systems [9]. In the domain of robot motion control and planning, DSLs are commonly employed on different levels. High-level task and skill abstractions are used to model a desired behavior, while on a lower level, control architectures are modeled using e.g., Simulink[1] or Scilab[2].

With the increasing number of commonly available (and affordable) collaborative robots, capable of highly precise force-torque control and sensing, new fields of application are opened up. We classify the high-level tasks defined by such applications into the Compliant Interactions (CI) domain. CI tasks often require force-based manipulation (e.g., assembly) that compliantly handles or actively exploits contacts, while uncertainties related to the robot's internal representation of the environment can be coped with. Further, these tasks are characterised by a close physical collaboration between robots and humans–more general–physical entities. Examples of such tasks are: performing a guided motion along a surface, using a hand rail for support and stability [10], and snap-fitting an object [5].

The required knowledge of the control formulations that actively considers environmental forces [11], [12] exist since decades. However, the explicit modeling of the interactions with the environment is still often completely neglected, or insufficient. In the latter case, the environmental information is either directly entangled with a control component or skill as a hidden assumption (e.g., [7]), or implicitely modeled

[1]CoR-Lab, Technical Faculty, Bielefeld University, Germany, {dwigand, swrede}@techfak.uni-bielefeld.de
[2]CNRS-University of Montpellier, LIRMM, France, niels.dehio@lirmm.fr
This work acknowledges the support by the European Union's Horzion 2020 projects № 732410 RobMoSys and № 644727 CogIMon.

[1]https://www.mathworks.com/products/simulink.html
[2]https://www.scilab.org/

as selection matrices [6] or as constraints on *feature coordinates* [13] (e.g., [14], [15], [8], [5]). While approaching to model the environment as *feature coordinates* seems reasonable, it only covers a small part that in turn neglects the explicit modeling of contacts surfaces, contact situations, and the transitions between those.

The contribution of this paper is a model-based approach that uses a domain-specific language (DSL) to describes the desired robot behavior in compliant interaction with the environment.

After a discussion of related work (II), the relevant concepts are presented (III) to explicitely model (physical) interactions with the environment and to link them to existing skills and control tasks. This link is mandatory to trace and to compare specific aspects of the interaction, such as contact situations, and their influence on the robot's behavior. By decoupling the environmental model form the skill or control component, hence avoiding hidden assumptions, the potential of adaptation to other environmental situations is increased, without the need of a domain expert to rewrite the control equation. To bridge the gap between the behavior specification for CI tasks and the actual execution, a synthesis mechanism is presented (IV) that generates a real-time capable component-based robot control architecture that realizes the CI specification. During the synthesis, the explicit model of the contact situations is integrated into the task, providing an environmental context that constraints the execution according to the presented contact situation. Finally, the relation between the execution and specification for a dual-arm yoga mat rolling experiment (see Fig. 1) is discussed V. The paper closes with a conclusion in VI.

II. STATE-OF-THE-ART

According to the literature, one or more tasks are usually used to define a robot's behavior w.r.t. a particular goal. In several works, tasks describe actions such as Peg-In-Hole [16], while skills represent smaller actions that are composed to realize a task, e.g., Move-To and Screwing [8]. However, the other way around can also be found in the literature [5], where skills act as container for multiple tasks. In general, there seems to be no common agreement about the definition of tasks and skills, since in some cases these terms are even used interchangeably. In our work, we refer to the overall goal as *task*, which is composed of sequenced *skills* that realize configurable and reusable subtasks. Further, we refer to control algorithms and control objective as *control tasks*. Hence, a controller implements a control task.

In a huge amount of publications, tasks do not consider contacts with the environment at all, however some agree that a task should explicitely contain a model of the environment as a context to execute skills in. Publications that consider the environment [5], [14], [15], [7] usually base the specification of skills and their composition on the TFF or on constraint-based programming approaches [13], which extend the TFF to support relations between multiple controllable frames on which constraints can be imposed. Both formalisms are commonly used in combination with DSLs to gain the advantages of a model-based approach [8], [5], [14], [15], [3]. For the coordination of the skills, often (finite) state machines [5], [14], [15], [7], or state charts [8] are used.

The contact situations are however very often directly entangled with the control task of the skills and in the case of using the TFF, the contact and constraints that result from a physical interaction are implicitely encoded by selection matrices. In contrast, work that is based on the constraint-based programming approach [13], however creates the link to the feature coordinates associated with the contacting bodies.

To the best of our knowledge, the explicit modeling of contact situations and contact state transitions that are composable and free from hidden assumptions, are still not sufficiently considered. Whereas, the general modeling of tasks and skills has already a quite long history in the literature. In related (sub)domains however, the constraints introduced by a contact are modeled using geometrical contact primitives [17] or kinematic mechanisms to gain knowledge about the constraint and unconstraint DoF [18]. In [17] contact state spaces are modeled based on discrete contact situations, while in [19] the life cycle of a contact is investigated to estimate when a contact is completely established, or if it is about to break. Certain aspects of this kind are integrated into our proposed approach in III.

III. COMPLIANT INTERACTION MODELING

In this section, we present the concepts required to model the contact situations for compliant interaction tasks. The aim is to integrate the relevant concepts from related (sub)domains, which focus on the different aspects of a contact, into our *Compliant Interaction Task* DSL that allows to use this knowledge to specify contact situations for CI tasks. The DSL is now part of the CoSiMA [20] modeling tool. By this, an environmental context can be provided by the task for the skill execution, which is decoupled from the skills and the desired behavior (i.e. trajectories). Thus, making these models composable and reusable in different applications. The modeling of couplings can be done using the extended Gazebo[3] simulator (see Fig. 3) as well as a textual representation in JetBrains MPS[4] (see Fig. 4).

A. CI Modeling via Contacts

Since a `Contact` is the natural interface through which (physical) interaction is possible, we select it as the central concept to model contact situations. In this case, we limit a contact to two contacting entities, i.e. geometric bodies. The relationship introduced by a contact situation over the two entities is described by a `Coupling`. One of the entities needs to be a controllable frame, i.e. a frame attached to an actuated kinematic chain. The type of interaction is defined by an `Coupling Formalism` (`F`) that realizes a contact constraint based on a specific (e.g., rigid-body) contact model, a behavior that ensures the compliance of

[3]http://gazebosim.org/

[4]https://www.jetbrains.com/

the controllable frame in a particular direction, or a second-order linear relationship between position and force that causes the controllable frame to behave as a physical mass-spring-damper system. Depending on the type of reference geometry, the nature of the coupling changes to either a `TaskSpaceCoupling` or a `JointSpaceCoupling`.

1) Constraint Coupling Type: Contacts can be represented by a combination of (in)equality constraints, depending on the contact situation [12]. In this paper, we model constraints, based on the rigid-body contact model, as bilateral force and velocity constraints with zero Cartesian contact velocity and acceleration [21], [22]:

$$\mathbf{J}_c^i \ \dot{\mathbf{q}} = 0 \text{ and } \mathbf{J}_c^i \ \ddot{\mathbf{q}} + \dot{\mathbf{J}}_c^i \ \dot{\mathbf{q}} = 0 \ , \tag{1}$$

where $\mathbf{J}_c^i \in \mathbb{R}^{6\times D}$ describes the constrained Jacobian associated with the ith contact point, and D the number of joints. However, considering that many contact situations impose unilateral rather than bilateral constraints, the constraints in Eq. 1 need to be extended by $\boldsymbol{\lambda}_{f,n} \geqslant 0$, ensuring that a contact force[5] $\boldsymbol{\lambda}_{f,n}$ can only be applied in the contact normal direction (n), to allow pushing and prevent pulling. A constraint can also be virtual, realizing an interaction that is not enforced by the environment, but by the high-level task in our framework. A constraint coupling is visualized as a red line with cross (see Fig. 3).

2) Compliance Coupling Type: The compliance type is similar to a contact constraint. However, instead of allowing a particular force to be exerted in the contact normal direction (n), the contact wrench needs to be zero: $\boldsymbol{\lambda}_{f,n} = 0 \wedge \boldsymbol{\lambda}_{t,n} = 0$. A compliance coupling is visualized as a green line (see Fig. 3).

3) Mass-Spring-Damper (MSD) Coupling Type: The behavior of a coupling can also be formalized such that when pertubed, the controllable frame behaves as a physical mass-spring-damper system, using a second-order linear relationship between position and force. Even though the coupling's formalism is realization-independent, such a behavior is commonly realized by an impedance controller, which is especially suited to cope with unknown forces that result from unmodeled dynamics and interactions with e.g., humans. Further, it ensures a stable response to anticipated or unstructured disturbances [23]. The external pertubation $\mathbf{F}_x$ of a cartesian controller is given by:

$$\mathbf{F}_x = \boldsymbol{\Lambda}_d \ddot{\tilde{\mathbf{x}}} + \mathbf{D}_d \dot{\tilde{\mathbf{x}}} + \mathbf{K}_d \tilde{\mathbf{x}} \ , \tag{2}$$

where $\tilde{\mathbf{x}} = \mathbf{x} - \mathbf{x}_d$ and $\mathbf{x}_d$ is the virtual equilibrium point, $\boldsymbol{\Lambda}_d$ is the desired inertia, $\mathbf{D}_d$ is the desired damping, and $\mathbf{K}_d$ is the desired stiffness matrix. A MSD coupling is visualized in blue, similar to the mechanical representation (see Fig. 3).

B. Contact Surfaces

Constraints that are naturally imposed by the surface geometry of a contact can be categorized as topological contact primitives based on the contacting geometries: point contacts, edge contacts, and surface contacts. These primitives can be combined in form of different *principal contacts* (PCs) [17]: $pc_i = (\texttt{a-b}),\ \texttt{a,b} \in \{\textbf{v}\text{ertex}, \textbf{e}\text{dge}, \textbf{f}\text{ace}\}$. Each PC relates to a set of natural constraints, which can be combined with further artificial constraints that are not bound to any physical relation, but purely chosen to express a desired behavior. In our approach, the contact surface is defined by the set of couplings that form the contact, where the couplings together shape the (virtual) contacting surface that can be represented in terms of PCs. Using TFF or a similar approach, it is also possible to derive the set of couplings from the PCs, since one or more constraint couplings can be used to represent the constraint directions imposed by the PCs. The relation between PCs and DoFs can be found in [18].

C. Controllable Frame and Virtual Manipulator

A controllable `Frame` (`CF`), attached to a body, can be actively controlled by an actuated kinematic chain. In order to control an object, which does not have any actuated joints, a contact can be established between a controllable frame and a frame of the object, turning the uncontrollable frame into a `Virtual Manipulator` (`VM`). Creating a VM is not limited to one kinematic chain, instead multiple chains can be combined as a closed kinematic tree as long as the internal forces are accounted for by employing a *force closure* or even a *form closure* using a suitable grasp.

In this paper we use a projection-based realization for a VM, based on the *grasp matrix* $\mathbf{G}$ that relates manipulated object twist to the contact twists of B manipulators [21]:

$$\mathbf{G} = \begin{bmatrix}\mathbf{G}_i \dots \mathbf{G}_B\end{bmatrix} \in \mathbb{R}^{6\times 6B}, \mathbf{G}_i = \begin{bmatrix}\mathbf{R}_i & \mathbf{0} \\ \mathbb{S}(\mathbf{r}_i) & \mathbf{R}_i\end{bmatrix} \in \mathbb{R}^{6\times 6}, \tag{3}$$

where $\mathbf{R}_i$ represents the rotation matrix of the ith contact frame. $\mathbf{r}_i$ is the distance between the ith contact position and the object's Center-of-Mass (CoM). $\mathbb{S}(\mathbf{r})$ is the skew-symmetric matrix. The geometry of the (virtually) manipulated object is modeled in the DSL and thus explicitely encoded into $\mathbf{G}$.

To control the internal forces, the nullspace projection of $\mathbf{G}$ is used. The resulting contact wrench does not produce any net wrench, i.e. $\mathbf{G}\mathbf{F}_c = \mathbf{0}$. Since the motion task should not be affected by the task that enforces the grasp, only the internal wrench is allowed to be controlled by

$$\mathbf{J}_{int} \in \mathbb{R}^{6B\times D} = \left(\mathbf{I} - \mathbf{G}^T(\mathbf{G}^+)^T\right) \begin{bmatrix}\mathbf{J}_1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \mathbf{J}_B\end{bmatrix}, \tag{4}$$

where $\mathbf{J}_i$ represents the ith manipulator Jacobian. For more detail, including the compensation for object dynamics, please refer to [21], [11], [24].

D. Relation to Control Tasks

The `Contact Situation` (`CS`) concept represents a set of contacts, where the behavior for each contact is defined by the types of couplings they are composed of. In order to express the desired behavior in a CS, the different formalisms

[5]The subscripts $(\ )_f$ and $(\ )_t$ refer to the contact forces and torques of the contact wrench $\boldsymbol{\lambda}$.

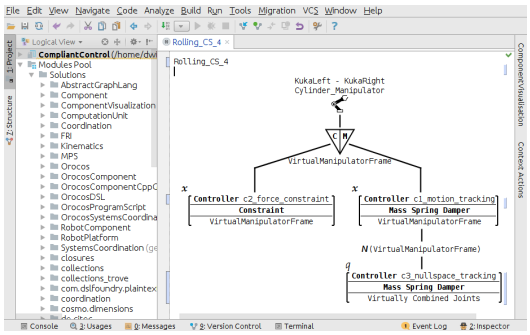

Fig. 2. Screenshot of a CCA DSL model that shows an exemplary prioritization structure using a virtual manipulator.

of all couplings per controllable frame and kinematic chain need to be composed and prioritized. Each formalism is realized by a `Control Task` (`CT`) concept. We use our *Compliant Control Architectures* (CCA) DSL (see Fig. 2), based on CoSiMA, to explicitely model the prioritization of control tasks. For each CS, a `CCA` concept is created that contains a `Prioritization Structure` (`PS`) concept for each involved kinematic chain.

1) Prioritization Modeling: To prioritize the control tasks for a PS, different types of relations are used. The concept of a `Strict` prioritization relation (depicted by an N in Fig. 2 and Fig. 5) is used to ensure that the second control task does not interfere with the primary one. That also means that the second task can only be accurately performed if enough DoF are available. Otherwise it is performed at best, without interfering with the primary task. Whereas, a `Soft` prioritization relation concept (depicted by a Σ in Fig. 2) can be employed to super impose a weighted set of control tasks using scalar priorities [25], [26]. The task space can be divided by a `Subspace` relation concept (depicted by a triangle in Fig. 2) into separate subspaces (e.g., constraint and unconstraint space), to avoid the interference of control tasks.

2) Projection-Based Realization: In this work, a strict hierarchy is realized utilizing a nullspace projection,

$$\mathbf{N} = \mathbf{I} - \mathbf{J}^T(\mathbf{J}\mathbf{J}^T)^{-1}\mathbf{J} = \mathbf{I} - \mathbf{J}^T(\mathbf{J}^{\#})^T \ , \tag{5}$$

where $\mathbf{J}$ is the task Jacobian of a (redundant) manipulator. The Projected Inverse Dynamics Control (PIDC) [27] approach provides a framework that allows to realize an impedance control law in the task space, while independently controlling the contact wrenches. To this end, it employs two orthogonal subspaces to prevent the reinforcement of contact and friction constraints, controlled by the joint torques $\boldsymbol{\tau}_c \in \mathbb{R}^D$ in the constrained subspace from affecting impedance characteristics and $\boldsymbol{\tau}_m \in \mathbb{R}^D$ in the unconstrained subspace:

$$\boldsymbol{\tau} = \mathbf{P}\boldsymbol{\tau}_m + (\mathbf{I} - \mathbf{P})\boldsymbol{\tau}_c \ . \tag{6}$$

It is important to note that the prioritization mechanisms are generally independent of a particular formalism. Thus, the application of a quadratic programming (QP) framework such as [28] instead of a projection-based approach is also possible.

E. Contact Lifecycle and Transition

A CS acts as a container for a set of contacts that exist in a particular interval in time. Multiple CSs together model the contact state space in a discrete way [17]. Transitions between CSs allow the switching between different contacts as well as between non-contact (free-space motion) and contact (constraint-space motion), in form of deactivating and activating sets of contacts. This is mandatory to sucessfully accomplish CI tasks [29], [30]. Each CS can have `Guards` that restrict or allow the traversal based on predefined conditions. Those guards can be uni- or bi-directional. The former type allows entering a CS (e.g., when a contact is established), while the latter actively triggers a transition to another CS (e.g., when a contact breaks). These conditions can be based on the internal high-level task state or sensor data. In case a transition from one CS to another CS is triggered and the set of active contacts changes, the realization of the contacts on the control level, as well as their internal state, defined by a `LifeCycle` [19], might change. This change is reflected by three different aspects: (1) the set of control tasks that are used, (2) the parametrization of these tasks, e.g., compensation for friction, and (3) the particular prioritization of the control tasks that expresses the desired behavior subject to the constraints enforced by the active contacts.

IV. SYNTHESIS

A. Representation of the Task and Prioritization

To synthesize a control architecture that realizes the behavior of the modeled contact situations, the reference architecture (see Fig. 6) needs to be instantiated according to the prioritization structures of the specified couplings.

A multi-staged model-to-model (M2M) transformation (see Alg. 1 and Fig. 5) is used to transform the modeled PSs to a set of prioritization graphs G_i^{CCA} for each CCA_i (see III-D). Such a graph is denoted as 5-tuple $(V, E, \sigma_v, \sigma_e, \sigma)$, where $V = V(G)$ is the set of vertices and $E = E(G) \subseteq V \times V$ is the set of edges of the graph. Each edge $e \in E$ is represented as an ordered pair of vertices, and the label functions $\sigma_v : V \to \sigma$, $\sigma_e : E \to \Sigma$ map vertices and edges to labels of the set Σ.

1) Transformation A: Each prioritization structure PS_j of a CCA_i gets transformed into a graph representation $\hat{G}_{i,K_r}$, where K_r is the kinematic chain addressed by PS_j. Every control task sharing the same controllable frame (CF) and formalism (F) is mapped to the same vertex (τ_l)[6] in the graph. The relations, i.e. `Strict`, `Soft`, and `Subspace`, are mapped to edges that connect two associated control tasks and are directed towards the higher prioritized one. The label $\sigma_e((\tau_u, \tau_v))$ indicates the type (i.e. "N" or "S") and CF of the relation. Since all control tasks will be present in the reference architecture as control components, all of them need to be considered in every CS, even if they are not part of the CS, because in this case the control task needs to be disabled. Hence, the resulting graphs ($\hat{G}$) are combined into

[6]Vertices are indicated by τ, because they refer to control task.

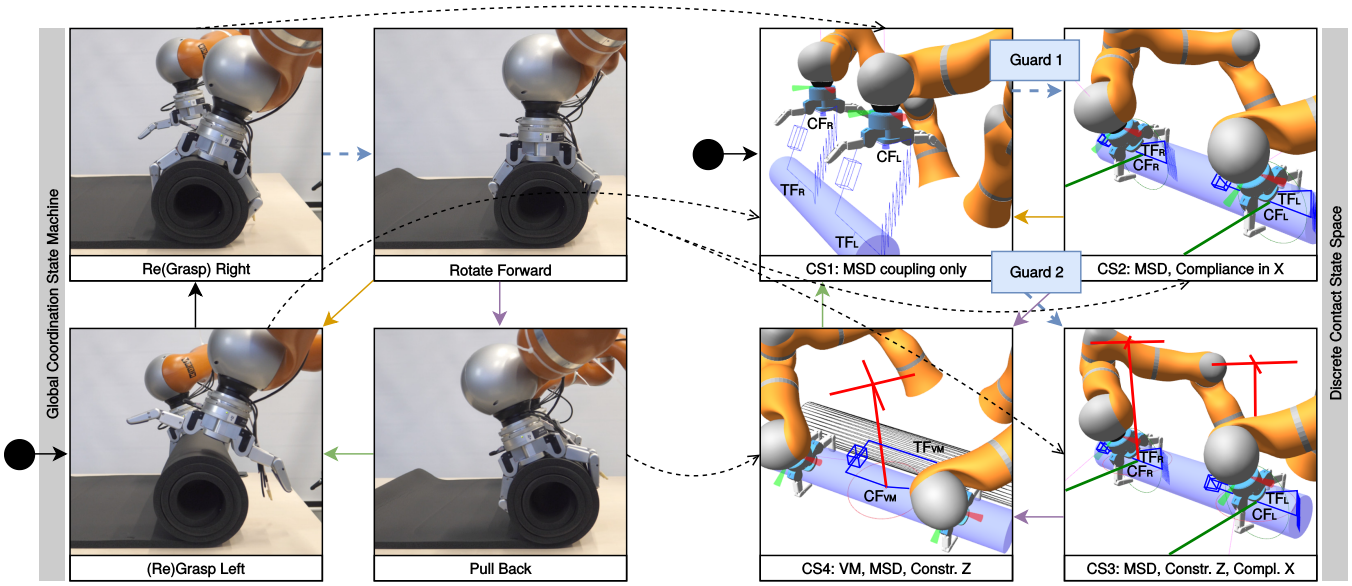

Fig. 3. The states of the scenario coordination (left) relate to the contact situation (right). They are executed in (curved dashed arrows). Contact state transitions are either triggered by a transition in the coordination (solid colored arrow), or using a guard to integrate external sensory information (dashed colored arrow), e.g., the force sensed in the EEF tool frame. Guards are used for the transition from CS1 to 2 or 3, depending on the sensed force after both hands finished grasping. Note, some of the arrows are visually overlayed for readability.

```
Taskspace Coupling: motion_tracking
State: active

Base: VirtualManipulatorFrame
Target: VF_MatTarget (external)

Formalism: Mass Spring Damper
Stiffness 30.0, 30.0, 30.0, 30.0, 30.0, 30.0
Damping 10.0, 10.0, 10.0, 10.0, 10.0, 10.0
realized by PositionPDController

Taskspace Coupling: force_constraint
State: inactive

Base: VirtualManipulatorFrame
Target: x: 0.0 , y: 0.0 , z: 1.0 , rr: 0.0 , rp: 0.0 , ry: 0.0 : world

Formalism: Constraint
...
```

Fig. 4. A model describing contacts in form of `Couplings` between (physical) entities. In addition to contact constraints, compliant behaviors, such as a mass spring damper, can be specified in different ways. The default control task, chosen as realization can be changed as well. Note that per default the inertial parameters of the controlled link are used.

unified graphs $\{U_{K_r}\}$, grouped by the associated kinematic chain (K_r) of the source PS, where

$$U_{K_r} = \bigcup A_r \mid A_r = \{\hat{G}_{i,K_p} \in \hat{G} \mid \dot{\exists} p : p = r\} \ . \tag{7}$$

2) Transformation B: Blending between two contact situations, represented as CCA_a and CCA_b, requires $\hat{G}_{a,K_r}$ to blend with $\hat{G}_{b,K_r}$ for all kinematic chains r. To allow this, every U_{K_r} provides the common ground for every CS involving the respective chain. Using Alg. 1, the combination with $\hat{G}_{i,K_r}$ then yields $i = 1, \ldots, |A_r|$ new graphs G_{i,K_r} that are grouped into sets of graphs G_i, where i refers to a particular source CCA_i. For each contact situation, the graphs (G_{i,K_r}) then contain the prioritization information of the control tasks $(V(G_{i,K_r}))$ for the kinematic chain (K_r), which the control tasks' CFs are associated with.

B. Instantiation of the Reference Architecture

A component-based architecture (CBA) is synthesized using the reference scheme shown in Fig. 6. The scheme is independent of a particular formalism. We synthesize a projection-based instantiation of the scheme, building on the PIDC framework and the work done in [21], [24]. To realize the modeled CSs, the CBA needs to be configured with the synthesized knowledge from IV-A.

1) Control Components: Each control task $\boldsymbol{\tau}_{r,i}$ in all U_{K_r} that shares the same formalism F and controllable frame CF, is executed by the same control component, which corresponds to the chosen realization of F and CF (e.g., a task-space impedance controller with stiffness and damping parameters according to F). To execute multiple control tasks in one controller, all the necessary information for each task are stacked and processed simultaneously. Eventually the resulting torque control signal for each task (stacked) is passed to the prioritization component. This mechanism allows to reduce the amount of instantiated control components. Each controller that realizes a `MSD` coupling needs to be of the form:

$$\boldsymbol{\tau}_m = \mathbf{P}\mathbf{J}_x^T \left[\mathbf{h}_c + \boldsymbol{\Lambda}_c \ddot{\mathbf{x}}_d - \mathbf{D}_d \dot{\tilde{\mathbf{x}}} - \mathbf{K}_d \tilde{\mathbf{x}}\right] \ , \tag{8}$$

where $\mathbf{h}_c = \boldsymbol{\Lambda}_c \mathbf{J}_x \mathbf{M}_c^{-1}(\mathbf{P}\mathbf{h} - \dot{\mathbf{P}}\dot{\mathbf{q}}) - \boldsymbol{\Lambda}_c \dot{\mathbf{J}}_x \dot{\mathbf{q}}$ denotes the operational space gravitational, centrifugal, and coriolis effects. `Constraint` or `Compliance` couplings share the same realization, with the desired wrench λ subject to the respective constraints:

$$\boldsymbol{\tau}_c = (\mathbf{I} - \mathbf{P})\mathbf{J}_c^T \boldsymbol{\lambda} \ . \tag{9}$$

*2) **Task** Component:* draws on the synthesized graphs G_{i,K_r} for the active CS to provide the required information to the other components. Control tasks $(\boldsymbol{\tau}_q)$ connected by the arrow's head of an edge with a subspace relation label (i.e. $\mathbf{S}_{CF_p}$) are employed in the constraint space, while tasks connected by the arrows' tail are employed in the

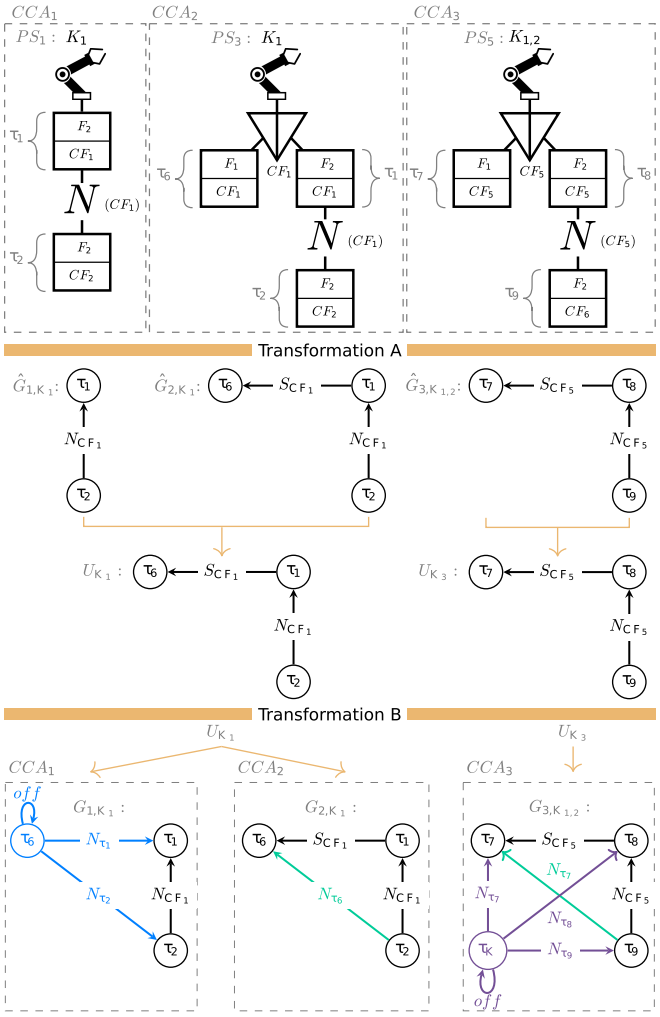

Fig. 5. Model-to-model transformation pipeline for the generation of prioritization graphs from CCA models.

unconstraint space. Two selection matrices are formed, based on the DoF of all constraint/compliant couplings associated with the respective edge: $\mathbf{S}_f^{CF}$ for the constraint space and $\mathbf{S}_m^{CF} = \mathbf{I} - \mathbf{S}_f^{CF}$ for the unconstraint space. The respective constraint ($\mathbf{J}_c^{CF}$) and unconstraint ($\mathbf{J}_x^{CF}$) Jacobian are denoted as

$$\mathbf{J}_c^{CF} = \mathbf{R}_O^{CF}\ \mathbf{S}_f^{CF}\ \mathbf{R}_{CF}^{O}\ \mathbf{J}^{CF} \tag{10}$$

$$\mathbf{J}_x^{CF} = \mathbf{R}_O^{CF}\ \mathbf{S}_m^{CF}\ \mathbf{R}_{CF}^{O}\ \mathbf{J}^{CF}\ , \tag{11}$$

where $\mathbf{R}_O^{CF}$ is the transformation between the controllable and the world frame. However, if the edge label references a nullspace relation, the control task has the full DoF in the unconstraint space ($\mathbf{S}_m^{CF} = \mathbf{I}$) and the projection is hence $\mathbf{P}^{CF} = \mathbf{I}$.

While the subspace projection for control tasks in the unconstraint space is denoted as

$$\mathbf{P}^{CF} = \mathbf{I} - \mathbf{J}_c^{CF^+}\mathbf{J}_c^{CF}\ , \tag{12}$$

$\mathbf{I} - \mathbf{P}^{CF}$ defines the projection for control tasks in the constraint space. In addition to that, the constraint consistent

Algorithm 1. Transformation B

Input: The set of $\{\hat{G}_{i,K_r}\}$ and $\{U_{K_r}\}$ from the previous transformation

for each $\hat{G}_{i,K_r}$ **do**

1) Make implicit nullspace relations explicit:

$E_p \leftarrow \{(v,u) \mid \forall v,u \in V(\hat{G}_{i,K_r}) : path(v,u) \wedge \neg path^{direct}(v,u)\}$

let $\sigma_e((v,u)) = "N"_{\sigma_v(u)} \mid \forall v,u : (v,u) \in E_p$

$G_{i,K_r} \leftarrow \hat{G}_{i,K_r} \cup E_p$

2) Deactivate irrelevant vertices (i.e. $\notin G_{i,K_r}$) and turn them into subordinates of the relevant vertices:

$V'' \leftarrow V(U_{K_r} - G_{i,K_r})$

$E'' \leftarrow \{(v,v) \mid v \in V''\}$

let $\sigma_e(e) = "off" \mid \forall e \in E''$

let $\sigma_e((v,u) \in E'') = "N"_{\sigma_v(u)} \mid \forall v,u : v \in V'' \wedge u \in V(G_{i,K_r})$

$G_{i,K_r} \leftarrow ((G_{i,K_r} \cup V'') \cup E'') \cup \{(v,u) \in E(U_{K_r}) \mid \forall v,u \in V''\}$

3) Introduce a deactivated vertex for the virtually closed kinematic chain as subordinate to all other vertices:

if K_r relates to a virtually closed kinematic chain **then**

let v_m be a vertex,

and let $\sigma_v(v_m) = "\tau_K"$

$E''' \leftarrow \{(v_m,u) \mid u \in V(G_{i,K_r})\}$

let $\sigma_e(e) = "N"_{\sigma_v(u)} \mid e = (v_m,u) \in E'''$

$G_{i,K_r} \leftarrow (G_{i,K_r} \cup v_m) \cup E'''$

end if

add G_{i,K_r} into the graph set G_i^{CCA} associated with the CCA_i

end for

Output: $G_{i,K_r} \in G_i^{CCA}$

joint-space inertia matrix is defined as

$$\mathbf{M}_c^{CF} = \mathbf{P}^{CF}\mathbf{M}^{CF} + \mathbf{I} - \mathbf{P}^{CF}\ . \tag{13}$$

For each VM, the Jacobian for the task, constraint, and internal wrench, based on Eq. 4, are provided analogeous to non-VM manipulators.

*3) **Prio**ritization Component:* receives the control signals and uses a mixture-of-controllers inspired approach, to achieve a continuous transition between different prioritization of K tasks [24], [31] using the matrix

$$\mathbf{\Psi} = (\alpha_{ij}) \in \mathbb{R}^{K \times K}\ , \tag{14}$$

where the priority for each pair of tasks $<i,j>$ is represented as a scalar value $\alpha_{i,j} \in \{0,1\}$ for a strict or soft $\alpha_{i,j} \in (0,1)$ hierarchy. The entries $\alpha_{i,i} \in [0,1]$ define the (de)activation of a task in the hierarchy. By smoothly adjusting $\mathbf{\Psi}$, the prioritization of tasks can be changed, which regarding the realization of constraints means that a smooth traversal between contact constraints and thus contact situations is possible.

The Task component is configured with the PS graphs (G_{i,K_r}) for each CS (CCA_i) synthesized in IV-A that are represented as $\mathbf{\Psi}_{i,K_r} = Adj(G_{i,K_r})$ per kinematic chain K_r,

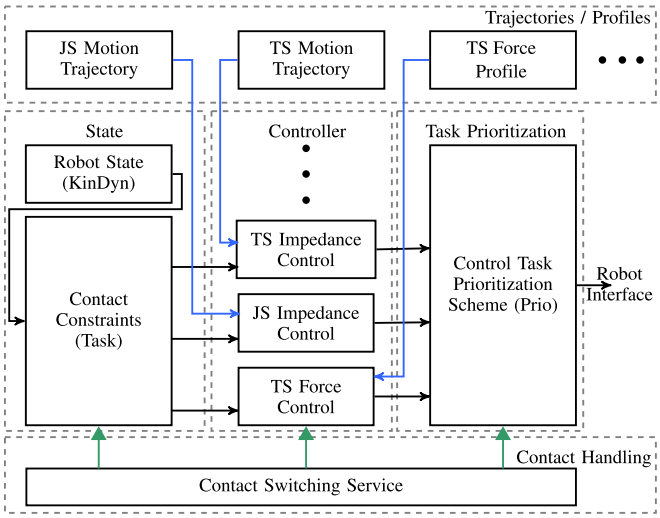

Fig. 6. Structural view of the reference architecture that is instantiated with the synthesized knowledge from the contact situations.

where $Adj(\cdot)$ denotes the adjacency matrix of a graph. In case of a VM, the resulting control signal of the referenced kinematic chains, is inserted as virtual control task $\boldsymbol{\tau}_K$ into the prioritization process of the VM.

4) CS Switching Service: provides an interface for the high-level task coordination to switch between CSs. The modeled contact state space is represented as a graph of CSs, including the transitions and guards. Guards are directly linked to the task coordination, or to the respective sensor data source by the application integrator. When a transition is triggered or finished, the Task, the Prioritization, and the control components are notified. During a transition the Task component shapes the data it provides according to the couplings in the new CS. Whereas, the Prioritization component blends the PSs by interpolating $\boldsymbol{\Psi}_{current}$ and $\boldsymbol{\Psi}_{new}$. After a transition is triggered, the controllers deactivate compensation effects (e.g., for friction) if specified by the associated coupling, while after the transition is finished, the compensation mechanisms are activated.

5) High-Level Task Coordination and Trajectories: The component-based control architecture that is synthesized requires trajectories in terms of set-points for the different control tasks. This information is not modeled with the presented DSLs and thus needs to be provided by the application or skill. Further, the application or skill needs to provide a task coordination if it aims at actively switching CSs and not only relies on passive switching based on sensory data.

V. EXPERIMENT

We chose the bimanual rolling of a yoga mat as experiment, because this high-level task involves several changes of contact situations (see Fig 3). While the task entails a state machine (SM) for coordination, the skills, i.e. *Regrasp*, *Rotate*, and *Pull*, contain their respective trajectories, according to the (position/force) goals provided by the SM. In addition to the SM, the task contains the model of the environment in form of a CS graph. For each active skill a CS needs to be active as well, to provide the environmental context for the execution of the skill. The *Regrasp* skill triggered by the SM for both manipulators, is executed in free space (CS1). Once the two hands are in contact with the mat, a transition is triggered to *Rotate Forward*. Thus, triggering the transition from CS1 to CS2 or CS3, depending on the sensed EEF force during the approaching of the mat. If the mat is not yet thick enough, CS2 is chosen to prevent the hands from pressing into the table. Otherwise, CS3 is chosen to compress the mat and compensate for the increasing radius while rolling. In both cases a compliance coupling in the rolling direction ensures a compliant forward motion guided by the mat. This prevents friction forces to cause the commanded joint torques to exceed their safety limit. An exerpt of the data collected from the experiment is shown in Fig. 7. For illustrative reasons, a segment is chosen where the mat is thick enough to show CS3. In each rolling phase, the skill moves the robots after the respective CS is activated. Once the motion is finished, the new position of the robots, caused by the compliant forwards motion, is used by the task to update the start and goal position for the next skill.

After the rolling phase, the mat is *Pull*ed back to avoid reaching the limits of the KUKAs workspace. Here, a VM for both manipulators is created with the geometry of a cylinder. While the mat is pulled back using a MSD coupling, a constraint in the world's z direction, prevents the VM's trajectory to lift the mat while pulling it back. Instead, a small force ($\boldsymbol{\lambda}_{f,z} = -2$ N) is applied to ensure staying in contact and to compensate for disturbances that arise from dragging on an uneven surface.

Once CS3 or CS4 is active, the desired trajectories in the constraint direction are ignored. Instead, a forces are exerted in that direction. With more rolling cycles, the radius of the mat increases, but the manipulator adapts accordingly.

For the experiment two KUKA LWR IV+ are torque controlled via FRI and the synthesized control system is deployed with the real time execution environment of CoSiMA [32] that builds on OROCOS RTT[7].

VI. DISCUSSIONS & CONCLUSIONS

In this paper, we discussed the current lack and need of a model of the interaction with the environment. This is even more important due to the increasing amount of robotic applications, involving force-based tasks that entail close physical interaction between robots or humans. Hence, these tasks require to compliantly handle or exploit environmental forces. With this motivation, our model-based approach is presented that enables the modeling of the environment in terms of contacts, contact situations, and transitions. A synthesis is used to generate an executable control architecture that executes skills constraint by the environment model. A dual arm experiment is used to compare the synthesized behavior to the model.

[7]https://www.orocos.org/rtt

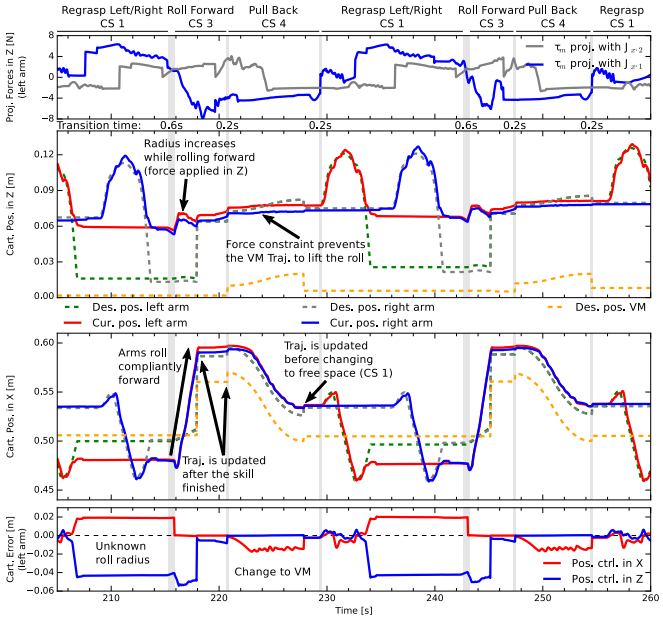

Fig. 7. Interpretation of the yoga mat rolling experiment. Note, the first plot shows the forces in Z, projected from the joint torques of the motion controller ($\boldsymbol{\tau}_m$), using its associated (sub space) Jacobian ($\mathbf{J}_x, 1$) and the counterpart provided to the force controller ($\mathbf{J}_x, 2$). The intersection of the signals indicates a (de)activation of a constraint coupling in Z.

Future work will focus on integrating aspects, such as material properties, into the synthesis, and on using the presented concepts for an automatic learning of couplings and the respective prioritization of control tasks.

References

[1] A. Björkelund, L. Edström, M. Haage, J. Malec, K. Nilsson, P. Nugues, S. G. Robertz, D. Störkle, A. Blomdell, R. Johansson, M. Linderoth, A. Nilsson, A. Robertsson, A. Stolt, and H. Bruyninckx, "On the integration of skilled robot motions for productivity in manufacturing," in *Int. Symp. Assem. Manuf.*, May 2011, pp. 1–9.

[2] M. R. Pedersen, L. Nalpantidis, R. S. Andersen, C. Schou, S. Bøgh, V. Krüger, and O. Madsen, "Robot skills for manufacturing: From concept to industrial deployment," *Robot. Comput.-Integr. Manuf.*, vol. 37, pp. 282 – 291, 2016.

[3] A. Nordmann, N. Hochgeschwender, D. Wigand, and S. Wrede, "A survey on domain-specific modeling and languages in robotics," *J. Softw. Engr. Robot.*, no. 7, pp. 75–99, 2016.

[4] T. Hasegawa, T. Suehiro, and K. Takase, "A model-based manipulation system with skill-based execution," *IEEE Trans. Robot. Autom.*, vol. 8, no. 5, pp. 535–544, Oct 1992.

[5] F. Nägele, L. Halt, P. Tenbrock, and A. Pott, "A prototype-based skill model for specifying robotic assembly tasks," in *IEEE Int. Conf. Robot. Automat.*, May 2018, pp. 558–565.

[6] M. T. Mason, "Compliance and force control for computer controlled manipulators," *IEEE Trans. Syst. Man Cybern.*, vol. 11, no. 6, pp. 418–432, June 1981.

[7] A. Nordmann, S. Wrede, and J. Steil, "Modeling of movement control architectures based on motion primitives using domain-specific languages," in *IEEE Int. Conf. Robot. Automat.*, 2015, pp. 5032–5039.

[8] U. Thomas, G. Hirzinger, B. Rumpe, C. Schulze, and A. Wortmann, "A new skill based robot programming language using uml/p statecharts," in *IEEE Int. Conf. Robot. Automat.*, May 2013, pp. 461–466.

[9] D. L. Wigand and S. Wrede, "Model-driven scheduling of real-time tasks for robotics systems," in *IEEE Int. Conf. Robot. Comput.*, Feb 2019, pp. 46–53.

[10] K. Koyanagi, H. Hirukawa, S. Hattori, M. Morisawa, S. Nakaoka, K. Harada, and S. Kajita, "A pattern generator of humanoid robots walking on a rough terrain using a handrail," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2008, pp. 2617–2622.

[11] H.-C. Lin, J. Smith, K. Kouhkiloui Babarahmati, N. Dehio, and M. Mistry, "A projected inverse dynamics approach for multi-arm cartesian impedance control," in *IEEE Int. Conf. Robot. Automat.* United States: Institute of Electrical and Electronics Engineers (IEEE), 9 2018, pp. 5421–5428.

[12] S. P. Patarinski and R. G. Botev, "Robot force control: A review," *Mechatronics*, vol. 3, no. 4, pp. 377–398, Aug. 1993.

[13] J. D. Schutter, T. D. Laet, J. Rutgeerts, W. Decré, R. Smits, E. Aertbeliën, K. Claes, and H. Bruyninckx, "Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty," *The International Journal of Robotics Research*, vol. 26, no. 5, pp. 433–455, 2007.

[14] M. Klotzbucher, R. Smits, H. Bruyninckx, and J. de Schutter, "Reusable hybrid force-velocity controlled motion specifications with executable domain specific languages," in *IEEE/RSJ Int. Conf. Intell. Robots Syst.*, I. Staff, Ed. [Place of publication not identified]: IEEE, 2011, pp. 4684–4689.

[15] D. Vanthienen, M. Klotzbuucher, J. de Schutter, T. de Laet, and H. Bruyninckx, "Rapid application development of constrained-based task modelling and execution using domain specific languages," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Piscataway, NJ: IEEE, 2013, pp. 1860–1866.

[16] H. Mosemann and F. M. Wahl, "Automatic decomposition of planned assembly sequences into skill primitives," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 5, pp. 709–718, Oct 2001.

[17] J. Xiao and X. Ji, "Automatic generation of high-level contact state space," *Int. J. Robot. Res.*, vol. 20, no. 7, pp. 584–606, 2001.

[18] J. Morrow and P. Khosla, "Manipulation task primitives for composing robot skills," in *Int. Conf. Robot. Automat.* IEEE, 1997.

[19] T. Lens, K. Radkhah, and O. von Stryk, "Simulation of dynamics and realistic contact forces for manipulators and legged robots with high joint elasticity," in *Int. Conf. Advanced Robot.*, June 2011, pp. 34–41.

[20] D. L. Wigand, A. Nordmann, N. Dehio, M. Mistry, and S. Wrede, "Domain-specific language modularization scheme applied to a multi-arm robotics use-case," *J. Softw. Engr. Robot.*, vol. 8, no. 1, pp. 45–64, 2017.

[21] N. Dehio, J. Smith, D. L. Wigand, G. Xin, H.-C. Lin, J. J. Steil, and M. Mistry, "Modeling and control of multi-arm and multi-leg robots: Compensating for object dynamics during grasping," in *IEEE Int. Conf. Robot. Automat.* IEEE, may 2018.

[22] R. Featherstone, S. S. Thiebaut, and O. Khatib, "A general contact model for dynamically-decoupled force/motion control," in *IEEE Int. Conf. Robot. Automat.*, vol. 4, May 1999, pp. 3281–3286.

[23] N. Hogan, "Impedance Control: An Approach to Manipulation: Part I—Theory," *J. Dyn. Syst. Meas. Control*, vol. 107, no. 1, pp. 1–7, Mar. 1985.

[24] N. Dehio, "Prioritized multi-objective robot control," Ph.D. dissertation, Technische Universität Braunschweig, 2018.

[25] K. Bouyarmane and A. Kheddar, "Using a multi-objective controller to synthesize simulated humanoid robot motion with changing contact configurations," in *IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2011, pp. 4414–4419.

[26] F. L. Moro, M. Gienger, A. Goswami, N. G. Tsagarakis, and D. G. Caldwell, "An attractor-based whole-body motion control (wbmc) system for humanoid robots," in *IEEE-RAS Int. Conf. Humanoid Robots*, Oct 2013, pp. 42–49.

[27] M. Mistry and L. Righetti, "Operational space control of constrained and underactuated systems," in *Robot. Science Syst.*, June 2011.

[28] E. M. Hoffman, A. Rocchi, A. Laurenzi, and N. G. Tsagarakis, "Robot control for dummies: Insights and examples using opensot," in *IEEE-RAS Int. Conf. Humanoid Robotics*, Nov 2017, pp. 736–741.

[29] J. R. Flanagan, M. C. Bowman, and R. S. Johansson, "Control strategies in object manipulation tasks," *Current Opinion in Neurobiology*, vol. 16, no. 6, pp. 650–659, Dec. 2006.

[30] R. S. Johansson and K. J. Cole, "Sensory-motor coordination during grasping and manipulative actions," *Current Opinion in Neurobiology*, vol. 2, no. 6, pp. 815–823, Dec. 1992.

[31] N. Dehio and J. J. Steil, "Dynamically-consistent generalized hierarchical control," in *IEEE/RSJ Int. Conf. Robot. Automat.*, 2019.

[32] D. L. Wigand, P. Mohammadi, E. M. Hoffman, N. G. Tsagarakis, J. J. Steil, and S. Wrede, "An open-source architecture for simulation, execution and analysis of real-time robotics systems," in *IEEE Int. Conf. Simul. Model. Program. Auton. Robots*, 2018, pp. 93–100.