2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Constraint-based specification of hybrid position-impedance-force tasks

Gianni Borghesan and Joris De Schutter

Abstract— **This work aims to extend the application field of the constraint-based control framework called iTaSC (*instantaneous task specification using constraints*) toward tasks where physical interaction between the robot and the environment, or a human, is contemplated. iTaSC, in its original formulation, allows for a systematic derivation of control schemes from task descriptions; tasks are defined as constraints enforced on outputs (e.g. distances, angles), and the iTaSC control takes care to fulfil such constraints by computing desired velocities to be commanded to the robot(s) joints. This approach, being based on a velocity resolution scheme, principally addresses tasks where positioning is the main issue. However, tasks that involve contacts with the environment or with the user, either desired or accidental, can be considered as well, taking advantage of impedance control, when position is controlled, or with force control. This paper describes the implementation of force tasks, and, by the combination of conflicting force and position tasks, impedance control, within the iTaSC formalism. This result is achieved by taking advantage of an approximate physical modelling of the robotic system and the environment. The proposed control scheme is tested by means of experiments where constraints on forces and/or positions described in cylindrical coordinates are imposed on a Kuka LWR arm.**

I. Introduction

Focus in robotic research is moving from applications defined in classical industrial settings, where the environment and tasks the robot performs are clearly defined, toward applications where the environment can be partially unknown, and physical interaction with objects, operators, and co-workers, is required, e.g. [1]. These aspects are even more relevant when the robot acts in a household environment: many tasks consist of physical interaction with tools designed for humans [2], or kinematically constrained objects [3].

Tasks with physical interaction are often realized by means of control strategies that involve hybrid force-position control [4], or impedance control, [5]; these two strategies have a 3-decades-long history, since the first works date back to the '80s.

On the other hand, iTaSC and other methods derived from the *task function approach* [6] offer several advantages in terms of task description. iTaSC [7] introduced some concepts to ease the description of tasks, namely feature frames, feature coordinates, and virtual kinematic chains (VKC). These concepts allow us to describe the outputs in any (minimal) coordinate system, to define tasks imposing desired values to such outputs, and tasks in order to define more complex tasks; such flexibility has proven to be very useful whenever several tasks are executed concurrently by highly articulated robots [8].

However, iTaSC has been designed for velocity controlled robots, and so most of the applications described in its original formulation focus on positioning tasks. Force control can by achieved with robots equipped with force sensors, as already hinted in [7].

Past works [8]–[10] already made some steps toward extending iTaSC toward physical interaction. These works proposed a way to exploit the back-drivability of robots for defining force tasks in a teleoperation context, without employing force sensing, and force nulling schemes.

In this work, we extend previous results and present them in a unified way; the main contributions with respect to the state of art is *to formalize the force and impedance control scheme derivation, so that it is possible to describe hybrid position-impedance-force tasks within the iTaSC framework*.

To reach this objective it is necessary to model the system admittance in the output space, as well as the forces, which are either measured or estimated in Cartesian or joint space.

II. iTaSC modelling procedure

An iTaSC application consists of tasks, robots and objects, a scene-graph, and a solver. For every application, the programmer first has to identify the **robots and objects**. In the framework an *object* can be any object in the robot system (for example the robot end-effector, a robot link, or an object attached to it) or in the robot environment. Next, the programmer defines *object frames* $\{o\}$ on the robots and objects (i.e. frames on their kinematic chains) at locations where a task will take effect, for instance the robot end-effector or an object to be tracked.

The object frames $\{o\}$ are defined w.r.t. the respective base frames $\{b\}$ which are placed at the base of a robot or in reference frame of an object; these frames, in turn, are described in function of a world frame $\{w\}$.

The actual **tasks** define the space between pairs of object frames ($\{o1\}$ and $\{o2\}$), the *feature space*, as a *virtual kinematic chain (VKC)*. To simplify the task definition, *feature frames* are introduced [7]. The feature frames are linked to an *object*, and indicate a *physical entity* on that object (such as a vertex or surface), or an *abstract geometric property* of a physical entity (such as the symmetry axis of a cylinder). Each task needs at least *two object frames* (called $\{o1\}$ and $\{o2\}$, each attached to one of the objects), and any number of *feature frames* (called $\{f1\}$, $\{f2\}$, ...) For an application in 3D space, there are in general six DOF between $\{o1\}$ and $\{o2\}$. Without loss of generality, we restrict to the case where the six DOF are distributed over

All authors gratefully acknowledge the European FP7 project RoboHow (FP7-ICT-288533).

G. Borghesan, and J. De Schutter are with the Department of Mechanical Engineering, K.U.Leuven, Heverlee, Belgium. name.surname@mech.kuleuven.be

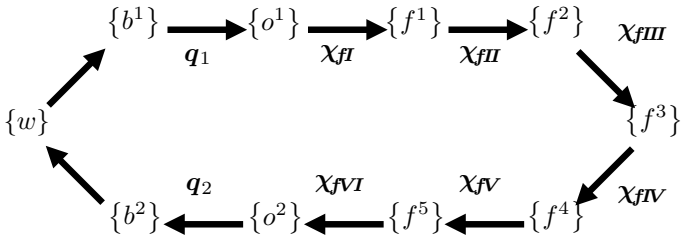


Fig. 1: Kinematic task loop with different frames and robot ($\boldsymbol{q}$) and feature ($\boldsymbol{\chi_f}$) coordinates. $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ are the controllable DOF of the first and second object respectively (typically the robot joints). The feature coordinates $\boldsymbol{\chi_f}$ are the DOF between $\{o1\}$ and $\{o2\}$, which, by introducing the feature frames, are distributed over six submotions: the relative motion of $\{f1\}$ with respect to $\{o1\}$ (submotion I, with feature coordinates $\boldsymbol{\chi_{fI}}$), the relative motion of $\{f2\}$ with respect to $\{f1\}$ (submotion II, with feature coordinates $\boldsymbol{\chi_{fII}}$), and so on.

six sub-motions, as shown in Fig. 1, i.e. each sub-motion is characterized with a sole degree of freedom.

The general framework allows to account for geometric uncertainties inside kinematic chains of objects or robots, or virtual kinematic chains. Uncertainties are represented with a minimal set of coordinates in strict analogy with feature coordinates, and are indicated with $\boldsymbol{\chi_{uI}}$, $\boldsymbol{\chi_{uII}}$, *etc*.

The treatment of uncertainties goes beyond the scope of this paper and (without loss of generality) will be omitted, assuming that all the geometrical properties of objects are known.

At this point, it is necessary to define how the robots and objects are located in the application scene. This is achieved by defining the relations between the reference frames of the robots and objects and a global world reference frame $\{w\}$. By connecting the VKC of the tasks to the object frames on the robots and objects, the programmer defines which robots execute the tasks on which objects. Each task defines a kinematic loop in the scene as shown in Fig. 1.

The kinematic loops introduce constraints between the robot coordinates $\boldsymbol{q}$ and the feature coordinates $\boldsymbol{\chi_f} = \left[\boldsymbol{\chi_{fI}}^T, \boldsymbol{\chi_{fII}}^T, \dots\right]^T$ expressed by the *loop closure equation*:

$$\boldsymbol{l}\left(\boldsymbol{q}, \boldsymbol{\chi_f}\right) = \boldsymbol{0}, \tag{1}$$

from which is possible to compute the loop closure equation at velocity level:

$$\frac{\partial \boldsymbol{l}\left(\boldsymbol{q}, \boldsymbol{\chi_f}\right)}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} + \frac{\partial \boldsymbol{l}\left(\boldsymbol{q}, \boldsymbol{\chi_f}\right)}{\partial \boldsymbol{\chi_f}}\dot{\boldsymbol{\chi}}_{\boldsymbol{f}} \triangleq \boldsymbol{J_q}\dot{\boldsymbol{q}} + \boldsymbol{J_f}\dot{\boldsymbol{\chi}}_{\boldsymbol{f}} = \boldsymbol{0}, \tag{2}$$

At this point, in order to obtain the desired task behaviour, one has to **impose constraints** on the relative motion between the two objects. To this end, the programmer has to choose the outputs that have to be constrained by defining an *output equation*:

$$\boldsymbol{y} = \boldsymbol{f}\left(\boldsymbol{q}, \boldsymbol{\chi_f}\right). \tag{3}$$

Feature coordinates are usually chosen so that they include the output, and thus $\boldsymbol{f}\left(\cdot\right)$ reduces to selection matrices:

$$\boldsymbol{y} = \boldsymbol{C_q}\boldsymbol{q} + \boldsymbol{C_f}\boldsymbol{\chi_f}, \textit{ and} \tag{4}$$

$$\dot{\boldsymbol{y}} = \boldsymbol{C_q}\dot{\boldsymbol{q}} + \boldsymbol{C_f}\dot{\boldsymbol{\chi}}_{\boldsymbol{f}}, \tag{5}$$

where $\boldsymbol{C_q}$ and $\boldsymbol{C_f}$ only contain zeros except for one '1' in each row.

The **imposed constraints** used to specify the task are then directly expressed on the outputs as:

$$\boldsymbol{y} = \boldsymbol{y}_d, \tag{6}$$

where subscript d denotes a desired value.

Constraints are enforced by a **controller**, and a **solver**.

The **controller** receives the desired output values ($\boldsymbol{y}_d$) and its derivatives ($\dot{\boldsymbol{y}}_d$) from a *set-point generator*, and computes the desired velocity $\dot{\boldsymbol{y}}_d^{\circ}$ in the output space:

$$\dot{\boldsymbol{y}}_d^{\circ} = \boldsymbol{g}\left(\boldsymbol{y}, \boldsymbol{y}_d, \dot{\boldsymbol{y}}_d\right). \tag{7}$$

The **solver** provides a solution for the optimization problem of calculating the desired robot joint velocities $\dot{\boldsymbol{q}}_d$ from the desired velocities computed by the controller ($\dot{\boldsymbol{y}}_d^{\circ}$):

$$\dot{\boldsymbol{q}}_d = \boldsymbol{A}_{\boldsymbol{W}}^{\#}\dot{\boldsymbol{y}}_d^{\circ},\ \boldsymbol{A} = \boldsymbol{C_q} - \boldsymbol{C_f}\boldsymbol{J_f}^{-1}\boldsymbol{J_q}. \tag{8}$$

The weighed pseudo-inverse computation involves two weighting matrices which allow us to weight conflicting constraints (over-constrained case), and to weight the actuation cost at joint velocity level (under-constrained case).

III. From force constraint specification to force control

The typical implementation of (7) is

$$\dot{\boldsymbol{y}}_d^{\circ} = \boldsymbol{K}_p(\boldsymbol{y}_d - \boldsymbol{y}) + \dot{\boldsymbol{y}}_d \tag{9}$$

This control, in the hypothesis that (*i*) the system composed by the robot and its low level control is able to follow the desired joint velocities, and (*ii*) there are no conflicting tasks, achieves perfect tracking, after a transition time which duration is ruled by the gain matrix $\boldsymbol{K}_p$.

As already shown in [7] and other works, in a velocity resolved scheme, reference forces cannot be imposed directly, but must be translated into displacement.

Let the generalized forces exerted by the robot in the output space be defined as $\boldsymbol{\tau_y}$; in order to compute the desired velocity to be actuated from desired force, the admittance of the system must be considered:

$$\delta\boldsymbol{y} = \boldsymbol{a}\left(\boldsymbol{\tau_y}\right) \tag{10}$$

A very good trade-off for (10) is to consider compliance, i.e. a linear, quasi-static approximation of $\boldsymbol{a}\left(\cdot\right)$:

$$\delta\boldsymbol{y} = \mathfrak{C}_{\boldsymbol{y}}\boldsymbol{\tau_y} \tag{11}$$

Where $\mathfrak{C}_{\boldsymbol{y}}$ is the total compliance of the system, and the subscript y indicates that it is expressed in the output space; how to compute such compliance is the focus of next paragraph.

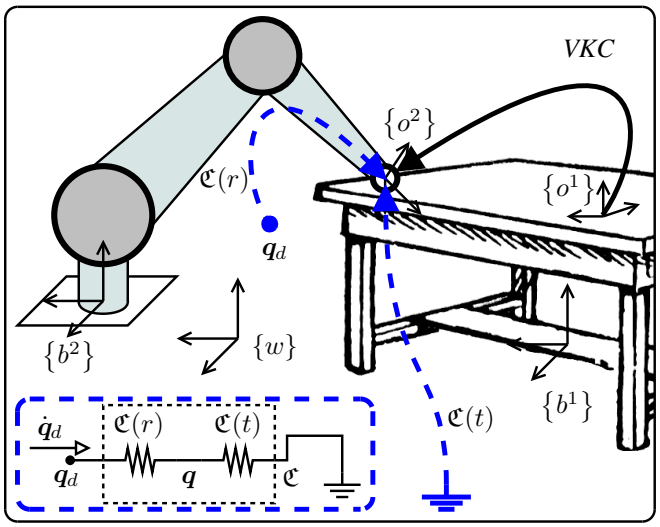


Fig. 2: Compliance of the system: the total compliance of the system is the sum of robot compliance, and the environment compliance. The robot is modelled as spring which stiffness is given by the control gains, and one of the end point is determined by the $\boldsymbol{q}_d$, that in turn is achieved by integration of $\dot{\boldsymbol{q}}_d$. On the bottom: the equivalent compliance of the system is shown. The total compliance of the system $\mathfrak{C}$ is the sum of the two compliances.

A. Compliance of the kinematic loop

The system represented in Fig. 2 shows a typical example of tasks that include physical interaction. The robot and the table are characterized by a compliance matrix, that is coordinate-dependent. For the sake of simplicity, we will make the assumption that the compliance depends only on the robot joints: this hypothesis implies that (*a*) the low level controller of the robot is a proportional joint position controller, and (*b*) the environment is characterized by linear compliance that does not depend on the objects configuration (e.g. fixed, not articulated objects). In the condition shown in Fig. 2 we have two compliances.

Let the robot be driven by the following law:

$$\boldsymbol{\tau_q} = \boldsymbol{K}_q(\boldsymbol{q}_d - \boldsymbol{q}) + \boldsymbol{g}(\boldsymbol{q}) \tag{12}$$

where $\boldsymbol{\tau_q}$ is the torque commanded at joint level (computed with a proportional controller, plus a gravity compensation term $\boldsymbol{g}(\boldsymbol{q})$), and $\boldsymbol{q}_d$ is obtained by integration of $\dot{\boldsymbol{q}}_d$, computed by (8).

Then, assuming that the compliance of the robot joints and links is negligible w.r.t. the control compliance, the compliance of the robot in joint space can be approximated as:

$$\mathfrak{C}_{\boldsymbol{q}}(r) \approx {\boldsymbol{K}_q}^{-1} \tag{13}$$

In other words, *the robot behaves as a spring in which the position of one attachment point is determined by* $\boldsymbol{q}_d$.

On the other side, the compliance of the object is expressed in some reference frame, (*e.g.* $\{o^1\}$) w.r.t. some application point. With simple operations, it can be expressed in any frame and any application points. For the sake of simplicity, let us consider the compliance of the object (table in Fig. 2) $\mathfrak{C}_{o1,o1}(t)$ be expressed in Cartesian coordinates $\{o1\}$ and applied in $\{o1\}$.

When the robot is pushing on the object, in absence of slipping or tilting, we can consider that they are fixed together i.e. the object frame $\{o1\}$ attached to one object $o1$ (the table), and the frame $\{o2\}$ attached to $o2$ (the robot) cannot experience relative movements.

In this situation, the compliances sum together (as shown at the bottom of Fig. 2), but in order to numerically compute it, all compliances must be represented in the same space and expressed in the same coordinate system, for example we can decide to represent all compliances in $\{o2\}$ and applied in $\{o2\}$. Then, the robot compliance is:

$$\mathfrak{C}_{o2,o2}(r) = \boldsymbol{J}_{ro2,o2}\, \mathfrak{C}_{\boldsymbol{q}}(r)\, \boldsymbol{J}_{ro2,o2}^T \tag{14}$$

where $\boldsymbol{J}_{ro2,o2}$ is the Jacobian that maps joint velocities to the twist expressed and applied in $\{o2\}$. The compliance for the object table, supposing that the object original compliance is given in $\{o1\}$, can be computed as:

$$\mathfrak{C}_{o2,o2}(t) = \mathbf{Ad}(\boldsymbol{T}_{o2}^{o1})\mathfrak{C}_{o1,o1}(t)\mathbf{Ad}(\boldsymbol{T}_{o2}^{o1})^T \tag{15}$$

where $\mathbf{Ad}(\boldsymbol{T}_{o2}^{o1})$ is the Adjoint matrix that maps twists expressed/applied in $\{o1\}$ to twists expressed/applied in $\{o2\}$.

Following this guideline, it is possible to express all compliances in a common frame and combine them in $\mathfrak{C}_{o2,o2}$,

$$\mathfrak{C}_{o2,o2} = \mathfrak{C}_{o2,o2}(r) + \mathfrak{C}_{o2,o2}(t). \tag{16}$$

The total compliance $\mathfrak{C}_*$ can be expressed also in the joint space:

$$\mathfrak{C}_{\boldsymbol{q}} = \mathfrak{C}_{\boldsymbol{q}}(r) + \mathfrak{C}_{\boldsymbol{q}}(t), \tag{17}$$

where $\mathfrak{C}_{\boldsymbol{q}}(t)$ can be computed in analogy with (14).

However, the final goal is to control the system in the output space, therefore the whole system compliance must be represented in such space.

B. Representing the system compliance in the feature and output space

Equations (2) and (5) contain the relations that are necessary for computing compliance in the output space: from $\boldsymbol{J_f}$, the Jacobian of the virtual kinematic chain, it is possible to compute $\boldsymbol{J}_{\boldsymbol{f}o2,o2}$, the Jacobian expressed/applied in $\{o2\}$, and consequently:

$$\mathfrak{C}_{\boldsymbol{f}} = \boldsymbol{J}_{\boldsymbol{f}o2,o2}^{-1}\mathfrak{C}_{o2,o2}\boldsymbol{J}_{\boldsymbol{f}o2,o2}^{-T}, \tag{18}$$

where $\mathfrak{C}_{\boldsymbol{f}}$ is the compliance expressed in the feature space and coordinates, and so relates the generalised positions and the generalised forces $\boldsymbol{\tau_f}$.

Finally, it is possible to compute the compliance in the output space, employing (17) for output defined from joint space, and (18), for output defined starting form feature coordinates, along with the respective selection matrices:

$$\mathfrak{C}_{\boldsymbol{y}} = \boldsymbol{C_f}\mathfrak{C}_{\boldsymbol{f}}\boldsymbol{C_f}^T + \boldsymbol{C_q}\mathfrak{C}_{\boldsymbol{q}}\boldsymbol{C_q}^T. \tag{19}$$

Thanks to (19), it is possible to relate forces and position/velocities in the output space, and also which is the desired displacement in order to achieve a desired force.

The last factor that is needed for implementing a force feedback law are the force measurements, *i.e.* the forces that are applied by the robot, either measured or estimated. How to express such forces in the output space is the topic of the next paragraph.

C. Force sensing

In order to close the loop over a desired force, the iTaSC controller must be able to measure the force in the output space. Force measurements can be achieved directly, by means of force sensors mounted on the robot, or in case the robot is back-drivable, the force applied by the robot on the environment can be approximated by a function of the force exerted by the motors. Independent of the space (joint or Cartesian) in which the force is sensed or estimated, it must be represented/applied in a way that is consistent with the compliance representation: the output space.

Following the same rationale used for compliance computation, measured forces are represented as wrenches $\boldsymbol{w}_{o2,o2}$ expressed/applied in $\{o2\}$, that could be measured directly (by equipping the robot with a force sensor in such point) or computed indirectly from measured torques:

$$\boldsymbol{w}_{o2,o2} = \boldsymbol{J}_{ro2,o2}^{-T} \boldsymbol{\tau_q} \tag{20}$$

Lastly, for each task, the measured wrench is expressed in feature and then output space:

$$\boldsymbol{\tau_f} = \boldsymbol{J}_{fo2,o2}^{T} \boldsymbol{w}_{o2,o2}, \tag{21}$$

$$\boldsymbol{\tau_y} = \boldsymbol{C_q} \boldsymbol{\tau_q} + \boldsymbol{C_f} \boldsymbol{\tau_f}. \tag{22}$$

At this point all the necessary elements to describe the extended iTaSC controller are present.

IV. The force-position-impedance controller

Section IV shows how to: *a*) formalize a control equation that regulates force, *b*) combine such equation with position constraints in order to realize an impedance behaviour, and *c*) cope with partial modelling of the physical properties of the scene, and discusses the expected behaviour in contact and in free space.

A. Constraint-based force control

The aim is to extend control equation (7) such that it includes generalized forces. The typical implementation of the controller shown in (9) has the nice property that the matrix $\boldsymbol{K}_p$ has as unit $[1/\mathrm{s}]$ independent of is the unit of the output (e.g. $[\mathrm{m}]$ or $[\mathrm{rad}]$). Since $\boldsymbol{K}_p$ is normally chosen as a diagonal matrix, the total system behaves as a set of first order systems whose settling times are ruled by $\boldsymbol{K}_p$ (under the assumptions that the robot executes the commanded velocity, and there are no conflicting constraints).

This property can be maintained by relating force errors to position errors:

$$\dot{\boldsymbol{y}}_d^{\circ} = \boldsymbol{K}_p \mathfrak{C}_{\boldsymbol{y}} (\boldsymbol{\tau_{y}}_d - \boldsymbol{\tau_y}) + \mathfrak{C}_{\boldsymbol{y}} \dot{\boldsymbol{\tau}}_{\boldsymbol{yd}} \tag{23}$$

where $\boldsymbol{\tau_{y}}_d$ and $\dot{\boldsymbol{\tau}}_{\boldsymbol{yd}}$ are the desired forces and their first derivatives.

Supposing that the modelled compliance is a good approximation of the real compliance in the system, the position error is related to the force error by the compliance matrix;

$$\mathfrak{C}_{\boldsymbol{y}} (\boldsymbol{\tau_{y}}_d - \boldsymbol{\tau_y}) = (\boldsymbol{y}_d - \boldsymbol{y}), \tag{24}$$

and hence (23) and (24) will have the same time evolution.

B. Constraint-based impedance control

Combining position and force constraints enables the imposition of compliance behaviour. Let $\boldsymbol{W}$ be a diagonal matrix containing normalised weights:

$$\boldsymbol{W} = \mathrm{diag}(w_1, \dots), \; w_i \in [0, 1]. \tag{25}$$

Let us consider as outputs generalised positions and forces that are expressed in the same space, *i.e.* derived from the same virtual kinematic chain and selected by the same selection matrices $\boldsymbol{C_f}$ and $\boldsymbol{C_q}$. Then, the composition of the two in the same equation results in:

$$\begin{aligned} \dot{\boldsymbol{y}}_d^{\circ} =& (\boldsymbol{1} - \boldsymbol{W}) \left(\boldsymbol{K}_p (\boldsymbol{y}_d - \boldsymbol{y}) + \dot{\boldsymbol{y}}_d \right) \\ &+ \boldsymbol{W} \left(\boldsymbol{K}_p \mathfrak{C}_{\boldsymbol{y}} (\boldsymbol{\tau_{y}}_d - \boldsymbol{\tau_y}) + \mathfrak{C}_{\boldsymbol{y}} \dot{\boldsymbol{\tau}}_{\boldsymbol{yd}} \right), \end{aligned} \tag{26}$$

Where the same gain matrix appears in both terms of the equation in order to impose the same time evolution of both force and position regulation.

The static behaviour of (26) is:

$$(\boldsymbol{1} - \boldsymbol{W})(\boldsymbol{y}_d - \boldsymbol{y}) = -\boldsymbol{W} \mathfrak{C}_{\boldsymbol{y}} (\boldsymbol{\tau_{y}}_d - \boldsymbol{\tau_y}), \tag{27}$$

from which the stiffness that is achieved by the controller can be computed:

$$\boldsymbol{K} = (\boldsymbol{1} - \boldsymbol{W}) \, \boldsymbol{W}^{-1} \mathfrak{C}_{\boldsymbol{y}}^{-1}. \tag{28}$$

Each output stiffness can vary from infinite (pure position control) to null (pure force control), depending on the value of w_i, effectively allowing to choose the behaviour of the system, in each of the output directions, independently.

V. Effects of inexact modelling and free-space behaviour of force control

As highlighted in the introductory section, iTaSC is often used in applications where the environment is partially unknown. Normally, the compliance of the robot in contact situation is dominated by (13), and can be considered known with good approximation. If the object which the robot interacts with is flexible, it will have a non-null compliance, and so the total compliance will increase, along with the feed-back gain $\boldsymbol{K}_p \mathfrak{C}_{\boldsymbol{y}}$ and feed-forward constant $\mathfrak{C}_{\boldsymbol{y}}$ of (23). This can be interpreted in the following way: in order to achieve the same force, the more the object is compliant, the bigger will be the needed compression.

In case the compliance is overestimated, the feed-forward will overcompensate, and the time constant will be reduced. On the contrary, an underestimation will increase the settling time, allowing for safer interaction, at a cost of reduced performance. In most of the cases, the objects are rigid and their compliance can be neglected.

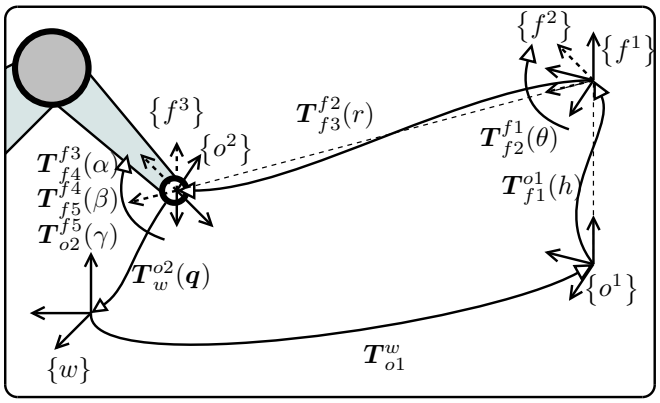


Fig. 3: Representation of the virtual kinematic chain employed in the experiments. It consists of a cylindrical coordinate system, with $\boldsymbol{\chi}_f = (h, \theta, r, \alpha, \beta, \gamma)$. Figure shows the object frames $\{o^1\}$ and $\{o^2\}$ and the first three feature frames, $\{f^1\}$, $\{f^2\}$, $\{f^3\}$. The other two feature frames, $\{f^4\}$, $\{f^5\}$, omitted for the sake of clarity, have the origin in common with $\{o^2\}$. Arrows with empty head represent transformations $\boldsymbol{T}(\cdot)$, that can be constant, ($\boldsymbol{T}_{o1}^{w}$), dependent on joint coordinates ($\boldsymbol{T}_{w}^{o2}(\boldsymbol{q})$), or on feature coordinates ($\boldsymbol{T}_{o2}^{o1}(\boldsymbol{\chi}_f)$).

Exp	pos.	imp.	for.	HRI/Free/Contact	Figure
1	h, θ, r			HRI	Fig. 4
2	r	h, θ		HRI	Fig. 5
3	r		h, θ	HRI	Fig. 6
4	h, θ, r			Free Space	Fig. 7a
5	r	h, θ		Free Space	Fig. 7b
6	θ, r		h	Contact	Fig. 8

TABLE I: List of experiments: In the first three experiments, the robot is pushed by a person (HRI, human robot interaction), with fixed desired position and desired forces. In the exp. 4 and 5, the robot follows a *position trajectory* without contacts, and in the last one a *force trajectory* while in contact with a rigid object. α, β, γ are always controlled in position (constant orientation).

If the robot is not in contact, from a modelling point of view, the compliance of the kinematic loop degenerates and becomes infinity, since no force can be achieved with any displacement: clearly this condition cannot be accounted for, and the compliance is modelled as if the robot is always in contact. This choice, as can be inferred from (23), causes the system to behave as a pure damper with damping constant:

$$\boldsymbol{B} = (\boldsymbol{K}_p \mathfrak{C}_{\boldsymbol{y}})^{-1}. \tag{29}$$

VI. Experimental evaluation

To illustrate the behaviour of the proposed approach, six experiments have been carried out. All experiments employ the same virtual kinematic chain that is anchored in a fixed position above a table from one side, and at the robot end-effector on the other side.

The robot used in the experiment is the KUKA LWR arm, a seven dof arm equipped with torque sensors mounted at each joint. The VKC between $\{o^1\}$ and $\{o^2\}$, where $\{o^2\}$ is placed on the robot end-effector, is shaped as a cylindrical-like coordinate system, and so is built using the following transformations: *i*) TransZ, *ii*) RotZ, *iii*) TransY, *iv*) RotX, *v*) RotY, and *vi*) RotZ. Each transformation is done w.r.t. its local frame, as shown in Fig. 3, and each one is parametrized by one of the feature coordinates:

$$\boldsymbol{\chi}_{\boldsymbol{f}} = (h, \theta, r, \alpha, \beta, \gamma).$$

The first three coordinates are cylindrical coordinates, while the last three angles represent the three DoF rotation that brings $\{f^3\}$ to $\{o^2\}$.

As output is chosen the whole feature space, $\boldsymbol{C}_{\boldsymbol{f}} = \mathbf{1}$, and no joint variables, *i.e.* $\boldsymbol{C}_{\boldsymbol{q}} = \emptyset$. For all experiments, we control the last four feature coordinates (r, α, β, γ) in position, with a fixed value: so $\{o^2\}$ (the robot end-effector) is commanded to maintain a constant distance w.r.t. the z-axis of $\{o^1\}$ and the end-effector is always facing the above mentioned axis. The other two feature coordinates (h, θ) are controlled in position ($w_i = 0$), impedance ($w_i = 0.8$), or force ($w_i = 1$). The end-effector is either pushed and released by a user, left in free space, or in contact with a rigid object, while the desired output values are commanded with a trajectory (free-space and hard-contact case) or kept constant (human interaction case). The experiments are summarised in Table I.

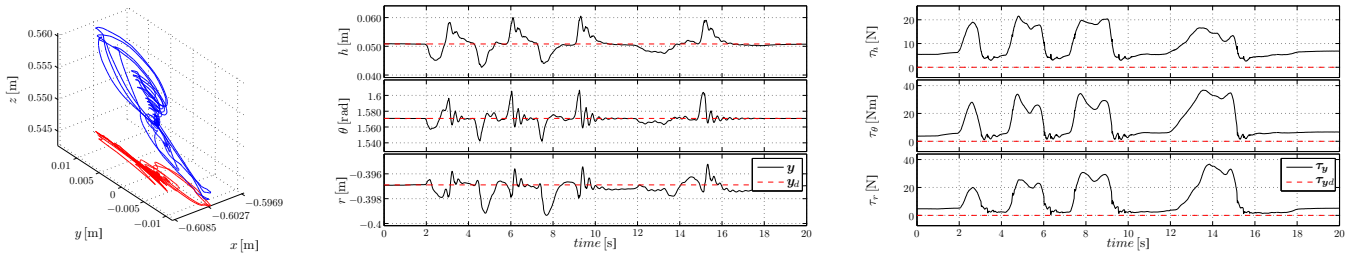
A. Physical interaction with user: exp. 1,2, and 3

The first three experiments show the behaviour of the robot when a user exerts force on it. Three possibilities are tested: (*i*) all outputs controlled in position (Fig. 4), (*ii*) the outputs h and θ controlled with the impedance strategy (Fig. 5), (*iii*) or controlled in force (Fig. 6), where the force reference in the last two cases is null. The expected behaviour is that:

(*i*) in the first case the iTaSC controller brings the position error to zero, trying to compensate for all disturbances,
(*ii*) in the second, the user experiences a spring like behaviour, but is constrained to a cylindrical surface,
(*iii*) in the latter case, the robot behaves as a damped system, that, again is constrained to move on the cylinder surface.

From Fig. 4 it is possible to appreciate the position accuracy, which is around $1\,\mathrm{cm}$, and the dynamics that the robot shows while recovering the reference position after a force disturbance. Note that, in a static condition, the position controller has infinite stiffness, since the iTaSC controller behaves as an integral controller, on top of the low level proportional controller: if a deviation from the original position is maintained for a long period, the controller will continue increasing the torque. For this reason, this kind of control is not safe and predictable when unforeseen collisions are likely to happen.

In the second experiment, the impedance control is tested. The weights for h and θ are set to 0.8. This translates to the physical stiffness described by (28). The user pushes the robot (that is constrained on the cylinder surface, whose axis is along the z-axis, and its origin is in $(-1, 0, 0.5)[\mathrm{m}]$). When the robot is released (i.e. when the estimated torques diminish, Fig. 5c), it goes back to the rest position, roughly

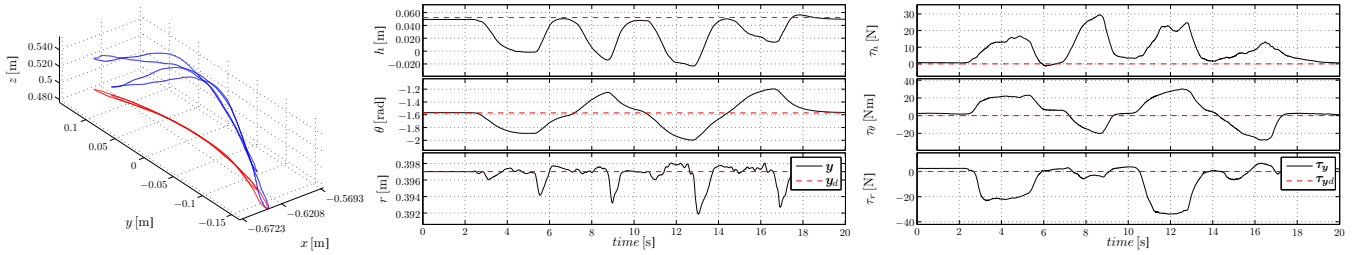


(a) Trajectory of $o2$ expressed in the robot base; in red, projection of trajectory in the $x-y$ plane.

(b) Output (gen. position)

(c) Output (generalised forces)

Fig. 4: Experiment N.1: all degrees of freedom are controlled in position.

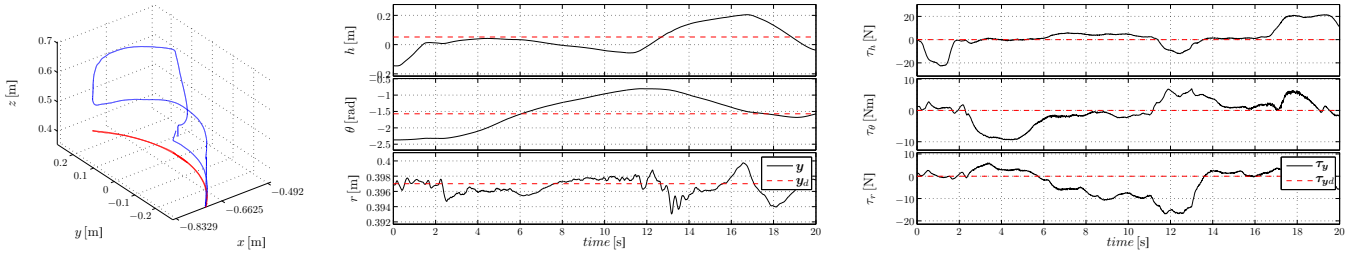


(a) Trajectory of $o2$ expressed in the robot base; in red, projection of trajectory in the $x-y$ plane.

(b) Output (gen. position)

(c) Output (generalised forces)

Fig. 5: Experiment N.2: Output h and θ are controlled with impedance.



(a) Trajectory of $o2$ expressed in the robot base; in red, projection of trajectory in the $x-y$ plane.

(b) Output (gen. position)

(c) Output (generalised forces)

Fig. 6: Experiment N.3: Output h and θ are controlled in force, (with a null reference force).

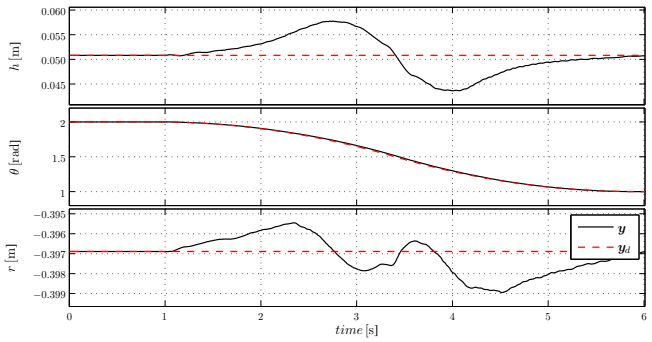
following first order dynamics in each of the output space direction, as shown by Fig. 5b.

The third experiment shows instead the force controller behaviour. Now the user is free to steer the robot end-effector on the cylinder surface, experiencing a force that is proportional to the velocity that he imposes to the robot, as ruled by (29). In Fig. 6a. it is possible to recognize the cylinder surface.

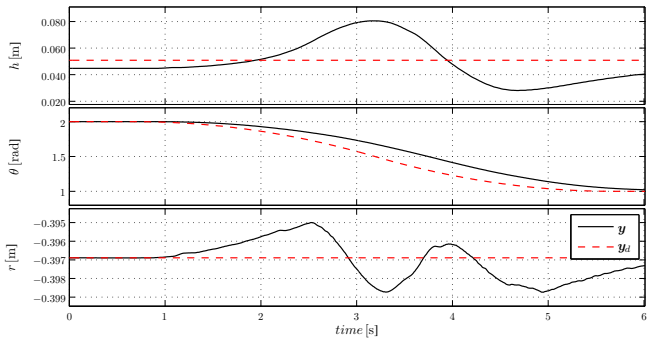
It is worth noticing that due to the kinematics of the virtual kinematic chain, for the same position of the the end-effector two solutions exist, one with positive r (e.g. Figs 5 and 6), or negative r (Fig. 4), that is mainly determined by the initialization of the inverse kinematic algorithm that solves the loop closure (1).

B. Position control in free space: exp. 4,5

In this case, the robot is commanded to move along a circular arc. The same movement is performed either with all outputs controlled in position, or with h and θ in impedance mode. Fig. 7 allows to compare the two experiments; as expected, the position control (Fig. 7a) behaves better, during the movement (smaller tracking error), and in steady state (steady state error tends to zero). The impedance controller, instead, shows a phase lag: since the force measured by the robot joint sensor is influenced by link inertias, sudden movements conflict with the zero force constraint. However, the degradation of performance can be acceptable when unexpected contacts are possible. Note that, in case forces would have been measured by a force sensor on the end-effector (or any other means to discern the forces due to



(a) Experiment 4, all outputs in position control.



(b) Experiment 5, outputs θ and h in impedance control (with null desired output torques).

Fig. 7: Experiment N. 4 and 5: The output θ is commanded to follow a trapezoidal velocity profile of time length $t = 5\,\mathrm{s}$, from $\theta_1 = 2\,\mathrm{rad}$ from $\theta_2 = 1\,\mathrm{rad}$.

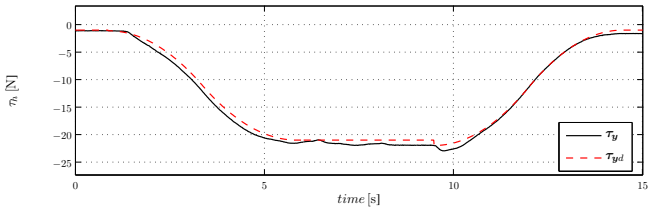


Fig. 8: Experiment N. 6: The output τ_h is commanded to follow two trapezoidal velocity profiles of time length $t = 5\,\mathrm{s}$ each: from $-1\,\mathrm{N}$ from $-21\,\mathrm{N}$, and *vice-versa*.

inertia from forces due to contact), the system would behave as in the first case.

C. Force control in contact situation, exp. 6

In this case the robot is commanded to exert a force on a table, starting from a contact situation. The desired force profile is composed by two trapezoidal profiles, from $-1\,\mathrm{N}$ to $-21\,\mathrm{N}$, and *vice-versa*. These profiles are characterized by the reference force and its derivative, as requested by the control equation (23).

The other degrees of freedom are commanded in position (i.e. the robot can move only along h), and since the h direction is physically constrained by the table that is very stiff, the robot does not move significantly.

The table compliance being negligible, assuming as total compliance the compliance of the low-lever robot controller is an accurate hypothesis, which results in the good force tracking shown in Fig. 8. In this case it can be observed that the feed-forward term is dominant w.r.t. the feedback, as the error in the second part of the trajectory is very reduced, while in the first five seconds the robot is actually anticipating a little the desired value.

VII. Final remarks

In this work, a systematic way to formalize hybrid position-impedance-force control within the constraint-based iTaSC-framework has been presented, along with some experiments to show the resulting behaviour. The method allows to deal with a number of tasks where contact situations is sought, expected, or possible. This method has been designed specifically to be used with iTaSC, but can be easily extended to any method where the control is solved at velocity level, as long as a Jacobian that maps output velocity to the control variable is computed, either by means of geometrical relations, analytically, or numerically.

On the other hand, we did not make any assumption on the method for measuring the force, as long as it is possible to compute the force applied on the origin of the object frame attached to the robot.

We did not mention, for the sake of brevity, other characteristics of the iTaSC framework, for example the treatment of geometric uncertainties, and the possibility to execute tasks where both objects are robots (e.g. bi-manual manipulation), or are articulated (e.g. a drawer). This contribution has been developed keeping in mind these aspects as well, and so it can be integrated with minimal effort in a complete iTaSC controller.

References

[1] S. Haddadin, M. Suppa, S. Fuchs, T. Bodenmller, A. Albu-Schffer, and G. Hirzinger, "Towards the robotic co-worker," in *Robotics Research*. Springer Berlin Heidelberg, 2011, vol. 70, pp. 261–282.

[2] C. C. Kemp, A. Edsinger, and E. Torres-Jara, "Challenges for Robot Manipulation in Human Environments," *IEEE Robotics & Automation Magazine*, vol. 14, no. March, pp. 20–29, 2007.

[3] C. Ott, B. Bauml, C. Borst, and G. Hirzinger, "Employing cartesian impedance control for the opening of a door: A case study in mobile manipulation," in *IFAC Symposium on Intelligent Autonomous Vehicles*, 2007.

[4] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, Feb. 1987.

[5] N. Hogan, "Impedance control: An approach to manipulation, parts i-iii," *Trans. of the ASME, Journal of Dynamic Systems, Measurement, and Control*, vol. 107, pp. 1–24, 1985.

[6] C. Samson, M. Le Borgne, and B. Espiau, *Robot Control, the Task Function Approach*. Oxford, England: Clarendon Press, 1991.

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

[8] D. Vanthienen, T. De Laet, W. Decré, H. Bruyninckx, and J. De Schutter, "Force-Sensorless and Bimanual Human-Robot Comanipulation Implementation using iTaSC," in *Proc. of 10th IFAC Symposium on Robot Control*, 2012, pp. 832–839.

[9] G. Borghesan, B. Willaert, T. De Laet, and J. De Schutter, "Teleoperation in Presence of Uncertainties : a Constraint-Based Approach," in *Proc. of 10th IFAC Symposium on Robot Control*, 2012, pp. 385–392.

[10] G. Borghesan, B. Willaert, and J. De Schutter, "A constraint-based programming approach to physical human-robot interaction," in *IEEE Proc. of the Int. Conf. on Robotics and Automation*, 2012.