Constraint- and synergy-based specification of manipulation tasks

Gianni Borghesan, Erwin Aertbeliën 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 manipulation tasks. iTaSC offers two advantages with respect to other methods: the ability to specify tasks in different spaces (and not only in Cartesian coordinates as for the Task Frame Formalism), and the treatment of geometric uncertainties. These properties may be very useful within a manipulation context, where tasks are executed by robots with many degrees of freedom, which calls for some degree of abstraction; by choosing a suitable set of coordinates, it is possible to reduce the complexity and the number of constraints that fully describe such tasks; in addition, controlling only the subspace that is needed to fulfil a task allows us to use the remaining degrees of freedom of the robot system to achieve secondary objectives. This paper discusses the instruments and techniques that can be employed in manipulation scenarios; in particular it focuses on aspects like the specification of a grasp and control of the stance of the robotic arm.**

iTaSC offers the possibility of specifying a grasp. While this approach allows for very fine control of a grasping task, in most cases a less fine-grain specification suffices to guarantee a successful execution of the grasping action. To this end synergy-based grasp specification is formulated within iTaSC.

We also show how to take into account secondary objectives for the arm stance. In particular we consider, as an example, the manipulability index along a given direction. Such indexes are maximised by exploring the null space of the other tasks.

The proposed approach is demonstrated by means of simulations, where a robotic hand grasps a cylindrical object.

I. Introduction

One of present challenges in robotic research is to bring robots outside their fences, and make them operate in environments designed for people rather than for robots, [1]. Lack of structure, uncertain *a priori* knowledge of the environment, and unexpected changes are difficulties that robots should cope with autonomously.

Vagueness in the environment description reflects also in vagueness of a task. While in an industrial scenario a robot may be programmed by means of joint trajectories, a household task is expressed in a more high level language: many of the details for actually computing the robot control action may be available only at run-time, and they may change with each execution. For such reasons, employing a framework that allows to express a task in local reference frames, in a suitable space, and to omit the specification of whatever is not strictly needed, seems to be an appropriate approach to develop such applications.

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

The authors are with the Department of Mechanical Engineering, K.U.Leuven, Heverlee, Belgium. name.surname@mech.kuleuven.be

Among these frameworks (*e.g.* the stack of tasks, [2], the operational space formulation, [3], task frame formalism by Mason, [4] and more recent developments, [5], *etc.*) we opted for iTaSC, [6], a velocity-resolved framework that allows for the automatic derivation of the control law, starting from the geometric specification of the task. While this aspect will not be emphasized in this work, one of the characteristics of iTaSC is the treatment of geometric uncertainties, i.e. unknown object positions, kinematic uncertainties, *etc.*, which are definitely relevant for the above mentioned applications.

iTaSC introduces feature variables and feature frames, as well as the concept of virtual kinematic chains. This allows us to model the quantities (*outputs*) to be controlled in the most convenient way. For example, cylindrical or spherical coordinates represent the distance between a point and a line, or between two points, respectively, and therefore should be introduced whenever such distances are the subject of constraints.

By employing iTaSC and the modelling procedure described in Sec. II the robot programmer does not have to worry about the derivation of geometric jacobians that are needed in velocity-resolved schemes. The iTaSC controller takes care of the robot control in an optimal way (given the cost function to be used in the optimization process), solves conflicts between constraints when the problem is over-constrained, and is able to execute more than one task at a time.

What iTaSC is partly missing, and what this work focuses on, is the control and representation of robotic grasping and the control of the arm configuration. We address the first issue by looking at hand control, Sec. III, with a particular emphasis on grasp synergy integration, while for the latter issue we focus on how to express complex constraints that live in the joint space (such as a manipulability index) in the iTaSC formulation, Sec. IV.

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

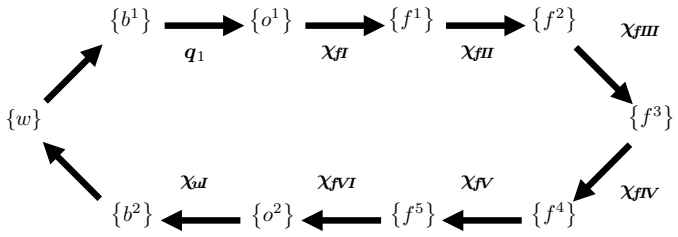


Fig. 1: Kinematic task loop with different frames and robot ($\boldsymbol{q}$) and feature ($\boldsymbol{\chi_f}$) coordinates. $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ are the controllable DOF of the first and second object respectively (typically the robot joints). The feature coordinates $\boldsymbol{\chi_f}$ 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.

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 [6]. 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 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:

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

This equation is normally implemented as a feedback/feed-forward combination:

$$\dot{\boldsymbol{y}}_d^\circ = \boldsymbol{K}_p(\boldsymbol{y}_d - \boldsymbol{y}) + \dot{\boldsymbol{y}}_d. \tag{8}$$

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 ($\dot{\boldsymbol{y}}_d^\circ$):

$$\dot{\boldsymbol{q}}_d = \boldsymbol{A}_{\boldsymbol{W}}^{\#}\dot{\boldsymbol{y}}_d^\circ,\ \boldsymbol{A} = \boldsymbol{C_q} - \boldsymbol{C_f}\boldsymbol{J_f}^{-1}\boldsymbol{J_q}. \tag{9}$$

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. Hand control

This section deals with the problem of controlling the hand. We propose two approaches: direct specification of the contact constraints, merely considering the robotic hand as a branched robot, and a synergy-based approach. Further on, in Sec. IV, we will introduce the tasks influencing the whole body configuration.

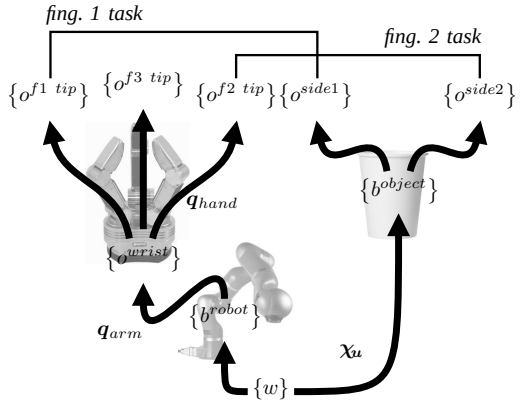


Fig. 2: Pinch grasping can be achieved by specifying the desired locations of two or more contact points: in iTaSC this can be achieved by imposing a zero distance between the origins of the object frames $\{o^{fi\ tip}\}$ and $\{o^{side\ i}\}$.

A. Direct specification of contacts

The first, natural approach to define a grasp is to specify it in terms of where each contact point should be on the surface of the object; in other words, the desired position of the robotic fingers' "end-effector" is defined (with respect to an object feature or locally defined frame). Many approaches investigate how to optimize the grasp contact point positions in order to achieve different goals, such as limiting the force in order to ensure the minimum internal force, minimise the number of contacts, e.g. [7]. Regardless of the method used to specify the location of the contact points, the outcome is a set of points that the fingers should reach: this translates into a set of kinematic loops (each one relating a frame attached to the robot and a frame on the object surface), and a set of constraints.

Fig. 2 shows an example of how a pinch grasp may be specified. Such grasp consists of opposing fingers making contacts on opposite sides of an object; the goal is to position the finger tips in the desired locations, while satisfying additional constraints on their relative orientation.

Employing iTaSC to specify the grasping by means of the contact points does not simplify the description of the grasp itself; however, its use allows us to take advantage of the other properties of this formalism, such as the use of geometric uncertainties, the combination with other tasks (for example imposing a certain behaviour to the robotic arm, as will be discussed in Sec. IV), and fine tuning the behaviour by means of weights.

This kind of task description can be realised with tools already offered in iTaSC, and will not be further discussed.

B. Synergy-based specification of hand configuration

When the goal of a robotic hand-centred task is grasping, it is possible to resort to the so-called postural synergies [8]. Synergies provide a reduced dimension base for the hands joint, that allows us to specify most of the hand postures needed for grasping as a combination of eigen-grasp vectors, [9]. Synergies $\boldsymbol{s}$ are defined as

$$\boldsymbol{q}_{hand} = \mathfrak{S}^{-1}(\boldsymbol{s})\,, \tag{10}$$

where $\mathfrak{S}^{-1}(\cdot)$ may be realised with a simple linear map $\mathbf{S}$ (that contains the eigen-vectors) plus an initial offset $\boldsymbol{q}_{so}$:

$$\boldsymbol{q}_{hand} = \mathfrak{S}^{-1}(\boldsymbol{s}) \triangleq \mathbf{S}\boldsymbol{s} + \boldsymbol{q}_{so} \tag{11}$$

In order to define the grasp, along with the desired configuration $\boldsymbol{s}_d$, the relative pose of the hand's wrist with respect to the object must be given: we follow and expand the guidelines given by Prats, [10], where the object frames (called hand frame $\boldsymbol{H}$ and grasp frame $\boldsymbol{G}$) employed in the task depend on the task itself, and optionally on the hand configuration.

The remainder of the section addresses the synthesis of the synergy controller and the specification of the wrist position.

1) synergy-based controller: Synergies fully constrain the joint space of the hand (at each $\boldsymbol{s}$ corresponds a given joint vector), without spanning the whole space, and so the matrix $\mathbf{S}$ has more rows than columns. By pseudo-inverting (11), we obtain the surjective function $\mathfrak{S}(\cdot)$, which maps different hand configurations to the same synergy without discerning whether a joint configuration actually belongs to the space spanned by the eigen-grasp base $\mathbf{S}$. Hence, unless additional constraints are added, a constraint in the synergy space cannot ensure that the hand will reach the correct configuration.

For this reason, the constraints are enforced directly in joint space: given the desired configuration of the hand, that is fully described by $\boldsymbol{s}_d$, the desired hand joint value $\boldsymbol{q}_{d,hand}$ is computed, and imposed as a set of joint constraints. The selection matrix $\boldsymbol{C_q}$ is chosen accordingly.

2) Control of the wrist frame: The complementary aspect of controlling the configuration of the hand is positioning the wrist in the correct pose with respect to the object that will be grasped. This desired wrist pose strongly depends on the hand configuration itself, but also on the task being performed.

For this reason, a set of object frames is associated with the hand, where each frame is associated with one or more grasp strategies. For example, Fig. 3 introduces two frames used for the envelope and pinch grasps. For the envelope grasp, the corresponding frame is positioned above the palm, in such a way it coincides with the center of a virtual object that is typically manipulated with a full grasp, while for the pinch grasp the corresponding frame is located where the finger-tips join together.

The robot system, including the hand and the arm, is thus described by the forward kinematic relation $\boldsymbol{T}_o^b(\boldsymbol{q}_{arm}, \boldsymbol{q}_{hand})$ from the base to each object frame defined, along with the corresponding jacobian matrix.

Without loss of generality, frames have been chosen to be independent of the hand configuration (*i.e.* $\boldsymbol{T}_o^b(\boldsymbol{q}_{arm}, \boldsymbol{q}_{hand}) \triangleq \boldsymbol{T}_o^b(\boldsymbol{q}_{arm})$). Subtler and smarter choices involving the adjustment of the object frame locations depending on the hand configuration and the characteristics of the objects to be grasped are deferred to future works, where

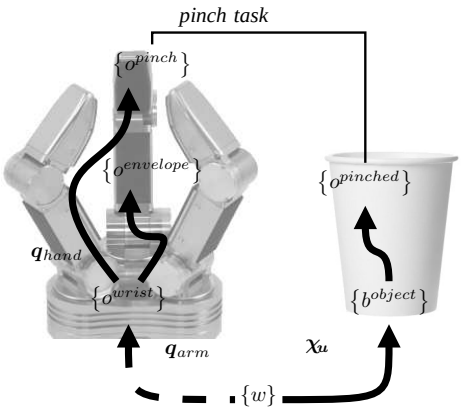


Fig. 3: By using synergies to define grasps, the wrist pose (w.r.t. the object) must be specified explicitly. Object frames (*e.g.* $\{o^{pinch}\}$ and $\{o^{envelope}\}$) are used to define wrist positioning. Depending on the object, more than one frame can be defined to provide reference for a specific grasp.

we will address the synthesis of grasping tasks rather than just their description.

IV. Arm Control: Specification of the arm configuration

Tasks are often specified with respect to only a single frame attached to the robotic hand and a single frame on the object to be grasped, and the constraints are therefore applied to just one virtual kinematic chain. However, systems that are inspired by human kinematics offer some degrees of redundancy; in iTaSC (and similar frameworks) redundancy translates in the possibility to execute additional tasks, involving different parts of the robot. Such additional tasks are executed in the null space of the primary task, such that the execution of former does not influence the latter.

In addition, robotic systems have tasks related to the preservation of their integrity, such as joint limit and obstacle avoidance, as well as self-collision avoidance. In a framework that allows for task prioritization, these tasks are given the highest priority, and they are normally formulated as inequalities. Therefore, they reduce the feasible space of the solution.

On the other side, there are other tasks (scheduled at lower priorities) that may support the execution of other tasks, e.g. by maximising manipulability at one of the frames involved in higher level tasks, or by mimicking human behaviour [11]. Several of these options exist, most of which are based on the computation of functions involving the manipulator jacobian (commonly applied to the object frame, and expressed in some other reference frame), which, in turn, depends on the robot configuration. These scalar functions can be stacked together in the function vector $\mathfrak{I}(\cdot)$, and the corresponding results in index vector $\boldsymbol{i}$:

$$\boldsymbol{i} = \mathfrak{I}(\boldsymbol{q})\,. \tag{12}$$

In order to impose the new constraints, eqs. (4) and (5) are modified as follows:

$$\boldsymbol{y} = \boldsymbol{C_i}\mathfrak{I}(\boldsymbol{q}) + \boldsymbol{C_q}\boldsymbol{q} + \boldsymbol{C_f}\boldsymbol{\chi_f}, \tag{13}$$

$$\dot{\boldsymbol{y}} = \boldsymbol{C_i}\left(\frac{\partial \mathfrak{I}(\boldsymbol{q})}{\partial \boldsymbol{q}}\right)\dot{\boldsymbol{q}} + \boldsymbol{C_q}\dot{\boldsymbol{q}} + \boldsymbol{C_f}\dot{\boldsymbol{\chi}}_{\boldsymbol{f}}, \tag{14}$$

where $\boldsymbol{C_i}$ is a selection matrix that selects which kind of "behaviour" amongst $\mathfrak{I}(\boldsymbol{q})$ is imposed.

V. Proposed scenario

As a practical example we consider: (*i*) the grasp of a cylindrical object (the glass), (*ii*) with a pinch grasp, (*iii*) while maintaining maximum manipulability along the z-axis, which coincides with the axis of the cylindrical object. This task is executed in simulation, employing a seven dof system (a Kuka LWR) equipped with an eight dof, three fingered robotic hand (a Schunk SDH). Each finger of the hand has two joints, while the hand includes two additional joints to rotate two fingers w.r.t the wrist in order to achieve wide grasps.

Specification of the task requires the choice of object frames and virtual kinematic chain (Sec. V-A), synergy directions (Sec. V-B), and manipulability index (Sec. V-C).

A. wrist pose control

For the pinch grasp, the object frames between which the virtual kinematic chain is built are $\{o^{pinch}\} \triangleq \{o^2\}$ and $\{o^{pinched}\} \triangleq \{o^1\}$, attached to the hand and to the object, respectively (see Fig. 4). The decision process on which virtual kinematic chain is the most suited takes into account the nature of the object and task, and whether it is necessary (or not) to completely specify the pose of the hand. Having chosen a cylindrical object, a cylindrical coordinate system is most appropriate to describe the task. In this case the virtual kinematic chain is built with the following choice of feature coordinates:

$$\boldsymbol{\chi_f} = (h,\, \theta,\, r,\, \alpha,\, \beta,\, \gamma)\,,$$

where h is the height along the cylinder axis, θ is the angle in cylindrical coordinates, and r is the distance from the axis of the cylinder. α, β, and γ are three generic angles, whose rotations close the kinematic loop, as shown in Fig. 4.

Lacking the need to identify a preferential direction of approach, θ is left unconstrained; conversely, we impose that: (*i*) the hand is directed toward the object axis, and with a given orientation w.r.t. to the cylinder axis (three constraints), (*ii*) the hand grasps the object between a minimum and maximum height ($h \in [-0.1, 0.1]\,\mathrm{m}$), and (*iii*) the distance r goes to zero (*i.e.* the origin of $\{o^2\}$ belongs to the z-axis of $\{o^1\}$). These specifications result in four equality constraints, and two inequalities enforced the same output (*i.e.* the height expressed relative to the cylinder axis).

B. The synergy function

Once the wrist is in the correct position, the fingers are brought into a given configuration by: (*i*) computing the desired joint (finger) positions, given the desired synergy vector, and (*ii*) imposing these positions to each joint, resulting in a total of eight constraints. Given the structure

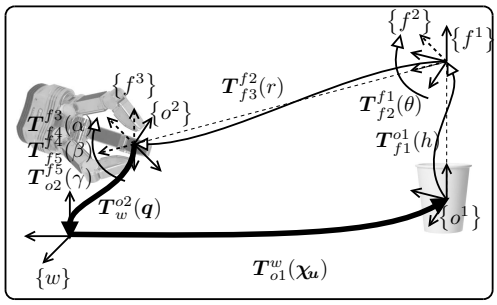


Fig. 4: Representation of the virtual kinematic chain used for grasping a cylindrical object. It consists of a cylindrical coordinate system, with $\boldsymbol{\chi}_f = (h, \theta, r, \alpha, \beta, \gamma)$. The figure shows the object frames $\{o^{pinched}\} \triangleq \{o^1\}$ and $\{o^{pinch}\} \triangleq \{o^2\}$ 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 their 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 ($\boldsymbol{T}_{o2}^{o1}(\boldsymbol{\chi}_f)$).

of the hand, the relation between synergies and joint angles is given by the following equations:

$$\begin{aligned}\boldsymbol{q}_{hand} &= \begin{bmatrix} \pi/2 & 0 & \pi/2 & 0 & \pi/2 & 0 & 0 & 0 \\ & & & \vdots & & & & \end{bmatrix}^T \boldsymbol{s} + \boldsymbol{q}_{so} \\ &\triangleq \mathbf{S}\boldsymbol{s} + \boldsymbol{q}_{so},\end{aligned} \tag{15}$$

where the synergy matrix base $\mathbf{S}$ is specified only for the first component (relative to the pinch grasp) that will be actually used, for imposing the desired value $\boldsymbol{s}_d = [1, 0, \cdots]$. The desired synergy vector $\boldsymbol{s}_d$ corresponds to a configuration where all fingers are perpendicular to the wrist base. In (15) the non-zero components are relative to the first joints of each finger.

C. The manipulability function

The specification given in Sec. V-A involves some redundancy (*i.e.* do not fully specify the configuration) in the positioning of the wrist (that can freely rotate along one axis, and can translate along the z-axis of $\{o^1\}$ between $0.1\,\mathrm{m}$ and $-0.1\,\mathrm{m}$), as well as in the joint space of the robot arm (the robot is redundant, having seven joints).

Taking advantage of these unconstrained degrees of freedom, a low-priority constraint on manipulability is enforced: for example, we consider the manipulability along a specific direction with respect to the world frame, expressed in the object frame used for grasping. This manipulability index (m_i) can be computed by intersecting the manipulability ellipsoid in six dimensions with a specific axis, (e.g. $\hat{\boldsymbol{z}} = [0, 0, 1, 0, 0, 0]$). These manipulability indexes are expressed along three axes as follows:

$$\boldsymbol{i} \equiv \begin{bmatrix} 1/m_x^2 \\ 1/m_y^2 \\ 1/m_z^2 \end{bmatrix} = \mathfrak{I}(\boldsymbol{q}) = \begin{bmatrix} \hat{\boldsymbol{x}}^T \boldsymbol{J}_r^{\dagger T} \boldsymbol{J}_r^{\dagger} \hat{\boldsymbol{x}} \\ \hat{\boldsymbol{y}}^T \boldsymbol{J}_r^{\dagger T} \boldsymbol{J}_r^{\dagger} \hat{\boldsymbol{y}} \\ \hat{\boldsymbol{z}}^T \boldsymbol{J}_r^{\dagger T} \boldsymbol{J}_r^{\dagger} \hat{\boldsymbol{z}}, \end{bmatrix} \tag{16}$$

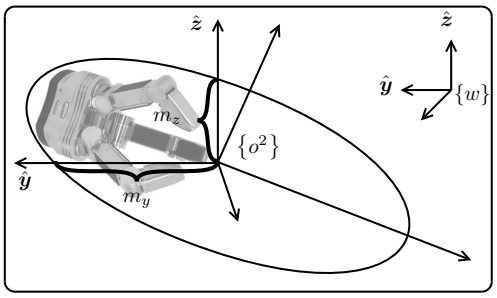


Fig. 5: Visual representation, on two dimensions, of the manipulability ellipsoid. Manipulability along a specific direction is measured by the coefficient m that is obtained by measuring the length of the vector between the intersection of the manipulability ellipsoid with the given axis, and the origin of the frame where the manipulability is measured.

where m_x, m_y, m_z are the distances between the axis origin and the intersection between the manipulability ellipsoid and the unit vectors $\hat{\boldsymbol{x}}$, $\hat{\boldsymbol{y}}$, and $\hat{\boldsymbol{z}}$, respectively. $\boldsymbol{J}_r^{\dagger}$ corresponds to the pseudo-inverse of the robot jacobian $\boldsymbol{J}_r$. If the manipulability analysis does not consider all robot joints to be equal, a columnspace-weighted pseudo-inverse of $\boldsymbol{J}_r$ can be used. A graphical representation of these indexes is reported in Fig. 5. As shown in (16), the manipulability indexes are the inverse of the square value of the m_i indexes: this choice allows to maximize the index m_i by imposing the respective index in $\boldsymbol{i}$ to be zero. In this application we would like to maximize the manipulability in the z direction, thus the selection matrix $\boldsymbol{C_i}$ in eqs. (13) and (14) must be chosen accordingly.

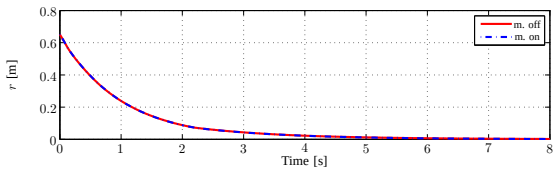
D. Simulations

We present two simulations, each one divided in two sequential actions: an approach motion that brings the object frame $\{o^2\}$ in a position where the constraints are fulfilled, and a closing grasp sequence. In the two experiments we compare the behaviour of the system when the manipulability constraints are active or not. It is worth noticing that, when redundancy is available, the solution algorithm minimizes the modulus of the instantaneous joint velocity vector.

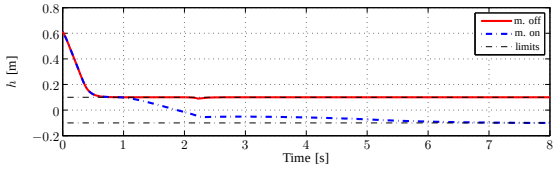
The most significant data of the approach motion are reported in Fig. 6. Fig. 6a shows that the behaviour of output r is the same in both cases, and evolves as a first order system as governed by (8). This behaviour is expected, since the constraint on distance is a high priority task, and no other task is conflicting with it.

The same reasoning applies to the first part of Fig. 6b: the hand is constrained to stay within the length of the cylinder (delimited by the two vertical dashed black lines), and for this reason the height of the hand decreases on both cases until the limit is met; then, in the experiment without manipulability optimization, the movements parallel to this axis practically stops, while, in the other case, the hand reaches the lower limit while trying to bring the index i_z as near as possible to zero.

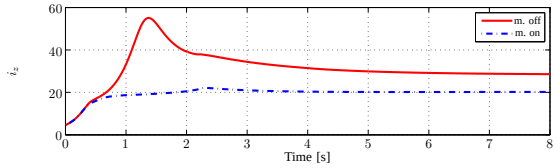
The final position of the wrist is different in both height and approach angle, as shown in Fig. 7. Fig. 7a shows the



(a) distance from the the cylinder axis.

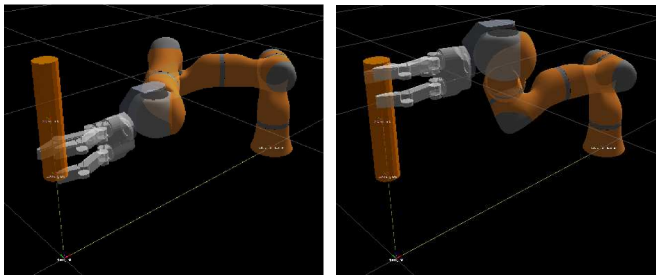


(b) Height on the cylinder axis.



(c) Manipulability index $i_z = 1/(m_z)^2$.

Fig. 6: Comparison between some of the outputs recorded in the two simulations; the red solid line and dash-dotted blue line refer to the simulation without and with the manipulability optimization, respectively.



(a) Constraint on manipulability active.

(b) Constraint on manipulability not active.

Fig. 7: Final pose of the robot in the two simulations.

arm configuration that maximizes the manipulability along the z-axis.

In the second part of the movement, the hand closes. The first element of the synergy vector is brought from 0 to 1 in 10 s, following a ramp; the three base angles, that are linearly related to it, follow the same behaviour, as reported in Fig. 8. Since the hand closing is not influenced by the presence of the manipulability constraint, the executions are identical in both experiments.

VI. Conclusion

This paper has shown how to employ iTaSC to describe manipulation tasks. This paper focused on reaching movement followed by a pinch grasp, but the same general criteria discussed in the first part can be applied to many other tasks. We pointed out how a grasping task can be expressed in a specifically chosen frame, as in [10], but the same concepts can be extended from the task frame formalism to iTaSC.

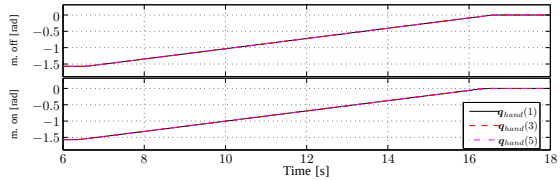


Fig. 8: Hand closing motion. the positions of finger joints positions are reported. While the value of the first element of the synergy vector is swept from 0 to 1, the base finger joints changes from $-\pi/2$ to 0. The other five joints, not reported here, are constantly zero.

Moreover, we have illustrated how joint space constraints, like synergies and manipulability indexes, can be easily brought into the task specification. In particular, we treated in a different way constraints that fully describe the desired pose, the synergy vector, and constraints that work in the null space of other constraints, like the proposed manipulability constraint.

In the discussion we omitted some details about the actual implementation: we omitted to report, for example, velocity and position joint's limits. Moreover, we did not take into account real physical interaction, limiting our investigation on the positioning problem, and leaving these aspects for future work.

References

[1] 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.

[2] N. Mansard, O. Khatib, and A. Kheddar, "A unified approach to integrate unilateral constraints in the stack of tasks," *Robotics, IEEE Transactions on*, vol. 25, no. 3, pp. 670–685, 2009.

[3] 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.

[4] M. T. Mason, "Compliance and force control for computer controlled manipulators," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 11, no. 6, pp. 418–432, 1981.

[5] T. Kröger, B. Finkemeyer, and F. M. Wahl, "A task frame formalism for practical implementations," in *Proc. of the IEEE International Conference on Robotics and Automation*, New Orleans, LA, USA, April 2004, pp. 5218–5223.

[6] 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.

[7] D. Berenson, R. Diankov, K. Nishiwaki, S. Kagami, and J. Kuffner, "Grasp planning in complex scenes," in *Humanoid Robots, 2007 7th IEEE-RAS International Conference on*, 2007, pp. 42–48.

[8] M. Santello, M. Flanders, and J. F. Soechting, "Postural Hand Synergies for Tool Use," *The Journal of Neuroscience*, vol. 18, no. 23, pp. 10 105–10 115, Dec. 1998.

[9] M. Ciocarlie and P. Allen, "Hand posture subspaces for dexterous robotic grasping," *The International Journal of Robotics Research*, vol. 28, pp. 851–867, 07/2009 2009. [Online]. Available: http://ijr.sagepub.com/cgi/reprint/28/7/851

[10] M. Prats, A. P. D. Pobil, and P. J. Sanz, *Robot Physical Interaction through the combination of Vision, Tactile and Force Feedback - Applications to Assistive Robotics*, ser. Springer Tracts in Advanced Robotics. Springer, 2013, vol. 84.

[11] A. Zanchettin, P. Rocco, L. Bascetta, I. Symeonidis, and S. Peldschus, "Kinematic analysis and synthesis of the human arm motion during a manipulation task," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 2011, pp. 2692–2697.