Force-Sensorless and Bimanual Human-Robot Comanipulation
Implementation using iTaSC

Dominick Vanthienen, Tinne De Laet, Wilm Decré *
Herman Bruyninckx, Joris De Schutter *

* *Dept. of Mechanical Engineering, KU Leuven,*
Celestijnenlaan 300B, B-3001 Leuven, Belgium.
(email: Dominick.Vanthienen@mech.kuleuven.be)

Abstract: This paper demonstrates the iTaSC approach to a force-sensorless and bimanual human-robot comanipulation task on a tree-structured robot, comprising (i) co-manipulation of an object with a person, (ii) dynamic and static obstacle avoidance with its base, (iii) maintaining visual contact with the operator, and (iv) unnatural pose prevention. The task is implemented in a structured way in a reusable software framework. The paper presents a simple sensorless wrench-nulling control scheme, enabling direct human-robot interaction without the use of a force sensor. A video shows the performance of the full task in different scenarios. The paper includes quantitative results for different scenarios, to validate the ability to activate and deactivate, as well as to change the weights of different parts of the task in a stable way.

Keywords: manipulation tasks, task specification, human-robot comanipulation, bimanual, iTaSC, sensorless

1. INTRODUCTION

Previous work on specification of sensor-based robot tasks, such as force-controlled manipulation (Hogan, 1985, 1987; De Schutter and Van Brussel, 1988; Kazerooni, 1989) or force-controlled compliant motion combined with visual servoing (Baeten et al., 2003), was based on the concept of the compliance frame (Mason, 1981) or task frame (Bruyninckx and De Schutter, 1996). In this frame, different control modes, such as trajectory following, force control, visual servoing, or distance control, are assigned to each of the translational directions along the frame axes and to each of the rotational directions about the frame axes. For each of these control modes a setpoint or a desired trajectory is specified.
For (geometrically) simple tasks this approach has proven its effectiveness. This approach however scales poorly to more complex tasks resulting in a variety of ad-hoc implementations. Examples of these more complex tasks, not fitting in the task frame approach, are control of multi-point contacts (Bruyninckx and De Schutter, 1996) and bimanual robot manipulation, which is the topic of this paper.

Constraint-based programming takes a conceptually different approach. It does not consider the robot joints nor the task frame as *centric*. Instead, the core idea behind the approach is to describe a robot task as a set of constraints, which do not necessarily have to be formulated in a single task frame, and one or multiple objective functions. Two such approaches are presented by Samson et al. (1991) and Mansard et al. (2009). Our own original task specification approach based on constraints was presented by De Schutter et al. (2007). In more recent work of Decré et al. (2009), the method was denoted iTaSC (**i**nstantaneous **Ta**sk **S**pecification using **C**onstraints) and the approach was extended to support inequality constraints.

The key advantages of iTaSC over classical motion specification methodologies are: (i) *composability of constraints*: multiple constraints can be combined and they can be partial, hence they do not have to constrain the full set of degrees-of-freedom (DOF) of the robot system; (ii) *reusability of constraints*: constraints specify a relation between frames attached to objects that have a semantic meaning in the context of a task, therefore the same task specification can be reused on different objects; (iii) *derivation of the control solution*: the iTaSC methodology provides a systematic approach to obtain expressions for the task constraints, to evaluate these expressions at run time, and to generate a robot motion that complies with the constraints by automatically deriving the input for a low-level controller; (iv) *modelling of uncertainty*: it provides a systematic approach to model uncertainties in this geometric model, to estimate these uncertainties at run time, and to adapt the control solution accordingly.

Recently, component-based software support was developed for iTaSC (Vanthienen et al. (2011)) that allows implementing iTaSC applications in a systematic way. The software follows the concept of the five C's (Radestock and Eisenbach, 1996; Prassler et al., 2009): **c**ommunication, **c**omputation, **c**oordination, **c**onfiguration, and **c**onnectivity.

⋆ All authors gratefully acknowledge the support by Willow Garage *PR2 Beta program*, the Flemish FWO project G040410N, KU Leuven's Concerted Research Action GOA/2010/011, KU Leuven—BOF PFV/10/002 Center-of-Excellence Optimization in Engineering (OPTEC), and European FP7 project RoboHow (FP7-ICT-288533). Tinne De Laet is a PostDoctoral Fellow of the Research Foundation - Flanders (FWO) in Belgium.

It is developed within the Orocos ecosystem (**O**pen **R**obot **C**ontrol **S**oftware (Bruyninckx, 2001)), complemented with rFSM scripts (**r**educed **F**inite **S**tate **M**achines (Klotzbuecher et al., 2010; Klotzbuecher, 2011)) for the state charts that take care of coordination and configuration. The core of an iTaSC application is built out of several types of components that will be introduced further on: robots, objects, virtual kinematic chains (VKC), constraint/controllers, a scene-graph, and a solver. The systematic approach of both specification and software implementation enables maximal reusability. While the same tasks can be easily executed by other robot systems, tasks implemented and specified before can be reused and combined with new tasks into a new application.

The contribution of this paper is the application of the iTaSC approach to a force-sensorless [1] and bimanual human-robot comanipulation task. In this application a PR2 robot: (i) co-manipulates an object with a person with his two hands, (ii) avoids dynamic and static obstacles with its base, (iii) maintains visual contact with the operator, and (iv) prevents unnatural poses. This application comprises three key challenges: (*i*) the definition of several constraints in different control spaces for a high DOF robot setup (20 DOF in total), (*ii*) the implementation of this task in a reusable software framework, and (*iii*) the implementation of a direct human-robot interaction task without the use of a force sensor. Preliminary results of the application were shown at the "Standard Platform Demonstration" booth at IROS 2011.

Different approaches to estimate the external wrench based on a disturbance observer are presented in literature (Tachi et al., 1991; Eom et al., 1998; Katsura et al., 2007). These approaches require a precise dynamic model of the robot. Since such a model is unavailable for the PR2, and the application does not require accurate force control, we use a simple estimation scheme.

This paper is organized as follows: Section 2 summarizes the iTaSC task modelling procedure, while Section 3 applies this modelling procedure to the bimanual human-robot comanipulation application. Section 4 elaborates on the force-sensorless wrench-nulling control used to achieve natural human-robot comanipulation. Section 5 presents the experimental results showing the successful application of iTaSC on the comanipulation application. Finally, Section 6 concludes the paper by summarizing the contributions and discussing future work.

2. ITASC MODELLING - GENERAL [2]

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 rigid object in the robot system (for example the robot end effector, link, or an object attached to it) or in the robot environment. Next, the kinematic chains of the robots and objects have to be defined as explained by De Schutter et al. (2007). They start at a reference frame (called base frame $\{b\}$) of the robot or object. The state of the kinematic chain is determined by the *joint coordinates* $\boldsymbol{q}$. 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 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 (De Schutter et al., 2007). 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 four frames: *two object frames* (called $\{o1\}$ and $\{o2\}$, each attached to one of the objects), and *two feature frames* (called $\{f1\}$ and $\{f2\}$, each attached to one of the corresponding features of the objects). For an application in 3D space, there are in general six DOF between $\{o1\}$ and $\{o2\}$. By introducing the feature frames, they are distributed over three submotions shown in Figure 1(a): submotion I, between $\{f1\}$ and $\{o2\}$ (feature coordinates $\boldsymbol{\chi_{fI}}$), submotion II, between $\{f2\}$ and $\{f1\}$ (feature coordinates $\boldsymbol{\chi_{fII}}$), and submotion III, between $\{o2\}$ and $\{f2\}$ (feature coordinates $\boldsymbol{\chi_{fIII}}$).

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*: $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{q}, \boldsymbol{\chi_f})$. De Schutter et al. (2007) provide guidelines on how to define the task's VKC such that the outputs are simple functions, in most cases simple selectors, of the feature and joint coordinates. The **imposed constraints** used to specify the task are then directly expressed on the outputs as: $\boldsymbol{y} = \boldsymbol{y}_d$, for equality constraints, or $y \geqslant y_d$ or $y \leqslant y_d$, for inequality constraints. Each output constraint is enforced by a **controller**, which receives the desired output values ($\boldsymbol{y}_d$) from a *set-point generator*.

By defining the relations between the reference frames of the robots and objects and a global world reference frame $\{w\}$ in the **scene-graph**, the programmer defines how the robots and objects are located in the application scene. 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. As such, each task defines a kinematic loop in the scene as shown in Figure 1(a). The kinematic loops introduce constraints between the robot coordinates $\boldsymbol{q}$ and the feature coordinates $\boldsymbol{\chi_f} = \left[\boldsymbol{\chi_{fI}}^T, \boldsymbol{\chi_{fII}}^T, \boldsymbol{\chi_{fIII}}^T\right]^T$, expressed by the loop closure equation: $\boldsymbol{l}(\boldsymbol{q}, \boldsymbol{\chi_f}) = \boldsymbol{0}$.

The **solver** provides a solution for the optimization problem of calculating the desired robot joint values (i.e. the joint velocities $\dot{\boldsymbol{q}}_d$ for a velocity-based control scheme) out of the task constraints. This allows to take into account different task priorities, different task constraint weights (in the case of conflicting constraints, i.e. overconstrained), and weights for the joints of the robots (to solve the kinematic redundancy of the robot in the underconstrained case).

[1] This is preferred over estimating the forces by measuring the currents in the actuators, because these estimates are disturbed by joint friction forces

[2] To prevent overloading the notation, this section leaves out uncertainty coordinates.

The new iTaSC software (Vanthienen et al., 2011) provides support to design and configure the different components separately.

The following section describes the details of the iTaSC modelling of the bimanual human-robot comanipulation application.

3. ITASC MODELLING - COMANIPULATION APPLICATION

This section details the iTaSC modelling of the bimanual human-robot comanipulation application. In particular, it discusses the different components of the iTaSC application: the robot and objects (Section 3.1), the tasks (Section 3.2), the scene-graph (Section 3.3), and the solver (Section 3.4).

3.1 Robots and objects

The application involves one PR2 robot and two objects: an obstacle and a moving person.

The **PR2 robot** has a tree-structured model, consisting of a branch from a reference frame $\{b\}$ fixed to the world through the robot base to its spine, from where the two seven DOF arm branches and the two DOF head branch start. Object frames $\{o\}$ are defined on the PR2's grippers, base, and head. As the PR2 robot moves in the application scene, its pose (i.e. the entire kinematic chain of the PR2) in the scene-graph is continuously updated using the PR2's odometry and encoders.

The **obstacle**, which the PR2 has to avoid in the application at hand, is considered fixed in the scene for sake of simplicity, but could be moving and/or uncertain. Therefore the obstacle's reference frame $\{b\}$ is fixed to the world. In this case a model with uncertainty coordinates $\boldsymbol{\chi_u}$ can be included with or without sensor measurements and motion model to update the obstacle's pose. An object frame $\{o\}$ is attached at the centre of the obstacle, with a fixed transformation with respect to the obstacle's reference frame $\{b\}$.

The position of the **moving person** is estimated by a face detection algorithm, applied on the robot's camera images. The face detection algorithm estimates the position of the head, which is then used to update the position of the moving person, and of the moving person's reference frame, in the scene-graph. An object frame $\{o\}$ is defined on the head of the person with a fixed transformation with respect to the moving person's reference frame.

3.2 Tasks

As explained in Section 2, a task consists of the definition of a VKC between object frames (*feature space*), *constraints* on the outputs $\boldsymbol{y}$ of the task's kinematic loop, *controllers* enforcing the different constraints, and *set-point generators* delivering the desired output values to the controllers. This section specifies and details all the tasks involved in the application.

The bimanual human-robot comanipulation application involves different tasks. Since the PR2 has to manipulate a rigid object with its two hands, a **grippers-parallel task** keeps the grippers parallel such that both hands are not moving with respect to each other and with respect to the manipulated object. Furthermore, the PR2 should comanipulate the object with a human, and therefore should react on the wrenches the human is applying to the object. Implementing this reaction to wrenches applied by the human results in a following behavior. To this end **a wrench-nulling task for each hand** is specified, i.e. a task whose goal is to minimize the wrench applied by the human. During the comanipulation the PR2's base has to avoid a fixed obstacle in the scene, specified by an **obstacle-avoidance task** and the PR2 head should watch the human operator, specified by a **head-tracking task**. Finally, a **joint-limits task** is defined to keep the robot joints away from the limits. The following paragraphs elaborate on the task definitions.

The **grippers-parallel task** defines a *Cartesian feature space* between the object frame on the left gripper $\{o1^p\}$ and the object frame on the right gripper $\{o2^p\}$.
The first feature frame $\{f1^p\}$ coincides with $\{o1^p\}$ and the second feature frame $\{f2^p\}$ coincides with $\{o2^p\}$. The feature coordinates defining the six DOF of the resulting *VKC* are all located between $\{f1^p\}$ and $\{f2^p\}$, i.e. $\boldsymbol{\chi_{fI}}^p = 0$ and $\boldsymbol{\chi_{fIII}}^p = 0$, while an intuitive definition of $\boldsymbol{\chi_{fII}}^p$ is obtained by expressing these coordinates in the first feature frame $\{f1^p\}$:

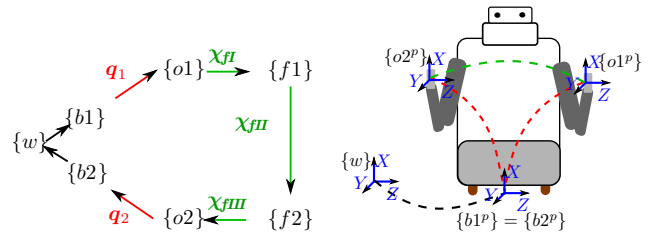
$$\boldsymbol{\chi_{fII}}^p = [x^p, y^p, z^p, \phi^p, \theta^p, \psi^p]^T , \tag{1}$$

where $[x^p, y^p, z^p]^T$ define the $3D$ position coordinates of the second gripper with respect to the first gripper and $[\phi^p, \theta^p, \psi^p]$ is a set of Euler-angles defining the orientation between the second and first gripper.
The task *constrains* all DOF of the VKC in order to keep the full pose between the grippers fixed. Therefore, the output vector $\boldsymbol{y}^p$ equals the feature coordinate vector $\boldsymbol{\chi_f}^p$.
A *proportional controller* is used to achieve the desired fixed pose, which is delivered by a *set-point generator* simply generating a fixed value ($x_d^p = 0m$, $y_d^p = 0m$, $z_d^p = 0.3m$, $\phi_d^p = 0rad$, $\theta_d^p = 0rad$, $\psi_d^p = 0rad$). The full rotation matrix is used in the control law to prevent singularities.
Figure 1(b) shows the kinematic loop of this task.

The applications contains **a wrench-nulling task for each gripper**, nulling the wrench exerted on that gripper. One wrench-nulling task defines a *Cartesian feature space* between the object frame on a gripper $\{o1^n\}$ and an object frame coinciding with the robot base $\{o2^n\}$.
The first feature frame $\{f1^n\}$ coincides with $\{o1^n\}$ and the second feature frame $\{f2^n\}$ coincides with respect to $\{o2^n\}$. The feature coordinates defining the six DOF of the resulting *VKC* are all located between $\{f1^n\}$ and $\{f2^n\}$, i.e. $\boldsymbol{\chi_{fI}}^n = 0$ and $\boldsymbol{\chi_{fIII}}^n = 0$, while an intuitive definition of $\boldsymbol{\chi_{fII}}^n$ is obtained by expressing these coordinates in the second feature frame $\{f2^n\}$:

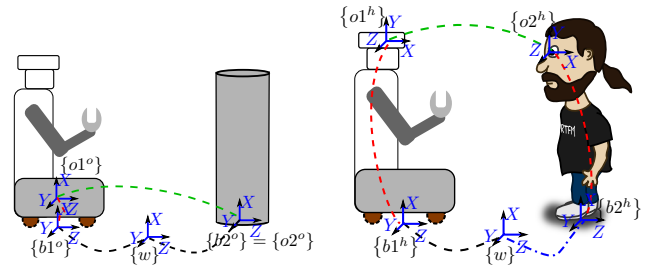
$$\boldsymbol{\chi_{fII}}^n = [x^n, y^n, z^n, \phi^n, \theta^n, \psi^n]^T , \tag{2}$$

where $[x^n, y^n, z^n]^T$ define the $3D$ position coordinates of the gripper with respect to the robot base and $[\phi^n, \theta^n, \psi^n]$ is a set of Euler-angles defining the orientation between the gripper and the robot base.
The output we are interested in is the wrench $\boldsymbol{y}^n = \left[F_x^n, F_y^n, F_z^n, M_x^n, M_y^n, M_z^n\right]^T$, which is considered in a



(a) General kinematic loop of a task.

(b) Kinematic loop of the grippers-parallel task (dashed lines).



(c) Kinematic loop of the obstacle-avoidance task (dashed lines).

(d) Kinematic loop of the head-tracking task (dashed lines).

Fig. 1. The red lines represent the kinematic chains of the robot and objects, the black line represents the (fixed) relation between the robot's fixed reference frame ($\{b\}$) and the world reference frame ($\{w\}$) defined in the scene-graph, and the green lines represent the *VKC* between the object frames ($\{o1\}$ and $\{o2\}$).

Cartesian feature space between the object frames on respectively the robot gripper and the robot base. Section 4 explains how the wrench output can be controlled using a wrench-nulling control scheme and the available task coordinates $\boldsymbol{\chi_f}^n$.

The **obstacle-avoidance task** for the robot base, defines a feature space with a cylindrical coordinate system, between the object frame on the obstacle $\{o1^o\}$ and the robot's base frame $\{o2^o\}$.
The first feature frame $\{f1^o\}$ coincides with $\{o1^o\}$ and the second feature frame $\{f2^o\}$ coincides with respect to $\{o2^o\}$. The feature coordinates defining the six DOF of the resulting *VKC* are all located between $\{f1^o\}$ and $\{f2^o\}$, i.e. $\boldsymbol{\chi_{fI}}^o = 0$ and $\boldsymbol{\chi_{fIII}}^o = 0$, while an intuitive definition of $\boldsymbol{\chi_{fII}}^o$ is obtained by expressing these coordinates in the first feature frame $\{f1^o\}$ using a cylindrical coordinate system:

$$\boldsymbol{\chi_{fII}}^o = [\theta^o, r^o, z^o, \alpha^o, \beta^o, \gamma^o]^T, \tag{3}$$

where $[\theta^o, y^o, z^o]^T$ define the $3D$ position coordinates of the robot base with respect to the obstacle and $[\phi^o, \theta^o, \psi^o]$ is a set of Euler-angles defining the orientation between the robot base and the obstacle.
As a result the distance to the object is represented by a single coordinate r^o. Since the distance is the only value of interest, only this DOF is *constrained*. Therefore, the output y^o is equal to r^o.
An inequality constraint is defined for the output, keeping the robot at a safe distance of the obstacle: $y^o \geqslant D$. To implement the inequality constraint, we use an approach using equality constraints enforced by a proportional controller and constraint monitoring. This approach uses adaptive constraint weights, which can gradually increase (decrease) to activate (deactivate) the constraints depending on (near) constraint violation. When the robot approaches the obstacle closer than a configured value, the distance constraint is activated. Consequently a *set-point generator* delivers a, in this case fixed, desired distance to the object (a little bit bigger than the D in order to drive the robot away from the obstacle.). When the robot is far enough from the obstacle, the task is deactivated by putting the task weight to zero. In the software, state charts handle the coordination of the constraint weights. Figure 1(c) shows the kinematic loop of this task.

The **head-tracking task** defines a *Cartesian feature space* between the object frame on the robot's head $\{o1^h\}$ and the object frame on the head of the person $\{o2^h\}$.
The first feature frame $\{f1^h\}$ coincides with $\{o1^h\}$ and the second feature frame $\{f2^h\}$ coincides with respect to $\{o2^h\}$. The feature coordinates defining the six DOF of the resulting *VKC* are all located between $\{f1^h\}$ and $\{f2^h\}$, i.e. $\boldsymbol{\chi_{fI}}^h = 0$ and $\boldsymbol{\chi_{fIII}}^h = 0$, while an intuitive definition of $\boldsymbol{\chi_{fII}}^h$ is obtained by expressing these coordinates in the first feature frame $\{f1^h\}$:

$$\boldsymbol{\chi_{fII}}^h = [x^h, y^h, z^h, \phi^h, \theta^h, \psi^h]^T, \tag{4}$$

where $[x^h, y^h, z^h]^T$ define the $3D$ position coordinates and the Euler-angles $[\phi^p, \theta^p, \psi^p]$ define the orientation, between the person's and the robot's head.
The z^h-direction connects the two object frame origins, i.e. the PR2's and the person's head. The task *constrains* the x^h and y^h coordinates, perpendicular to the z^h direction, to be zero. Therefore, the output vector $\boldsymbol{y}^h$ is equal to $[x^h, y^h]^T$. A *set-point generator* delivers a zero value for both the x^h and y^h coordinate, enforced by a *proportional controller*. As a result, the head of the robot will align with the head of the person, hereby keeping the PR2's cameras pointed towards the person's head. Figure 1(d) shows the kinematic loop of this task.

The **joint-limits task** keeps the PR2's arm joints out of its limits. It *constrains* the robot joints themselves, therefore there is no need for extra loop closure equations and hence *no VKC*. As for the obstacle avoidance task, an equality constraint approach is chosen. When a joint approaches its lower or higher joint limit, this task is activated. The task applies a velocity to the joint to move it away from the approached limit. When the joint position is far enough from its limits, the task deactivates. The magnitude of the desired velocity, delivered by the *set-point generator*, as well as the weight of the task, increase with the proximity to the joint limits, as shown in figure 2. As a result, the activation of the joint-limits task causes a smooth transition, rather than a discontinuity in behaviour.

3.3 The scene-graph

The scene-graph keeps track of all robots and objects involved in the application, and in particular of their poses with respect to the world reference frame $\{w\}$. The scene component of the software support contains this scene-graph. Other components (the robots and objects, the tasks and the solver) of iTaSC deliver their status

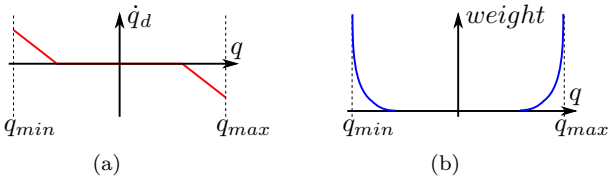


Fig. 2. Desired joint velocity $\dot{q}_d$ (a) and weight of the joint-limits task (b) in function of the joint position q. q_{min} and q_{max} define the upper and lower joint limits respectively.

information, e.g. current pose, desired velocities, ..., to this scene component and request similar scene-graph information from it.

3.4 The solver

The solver calculates the desired joint velocities $\dot{\boldsymbol{q}}_d$, to be sent to the controllable DOF of the robots by solving an optimization problem involving the task constraints, the constraint weights, the task priorities, and the robot weight matrix. In this application the solver has to calculate the desired velocities for the 20 controllable DOF of the PR2, out of maximum 35 constraint equations. By taking weights and priorities of the different task constraints and the robot DOF into account, the behaviour of the application can be configured.

Figure 3 shows the overall control scheme of all tasks combined and the solver calculating the joint velocities.

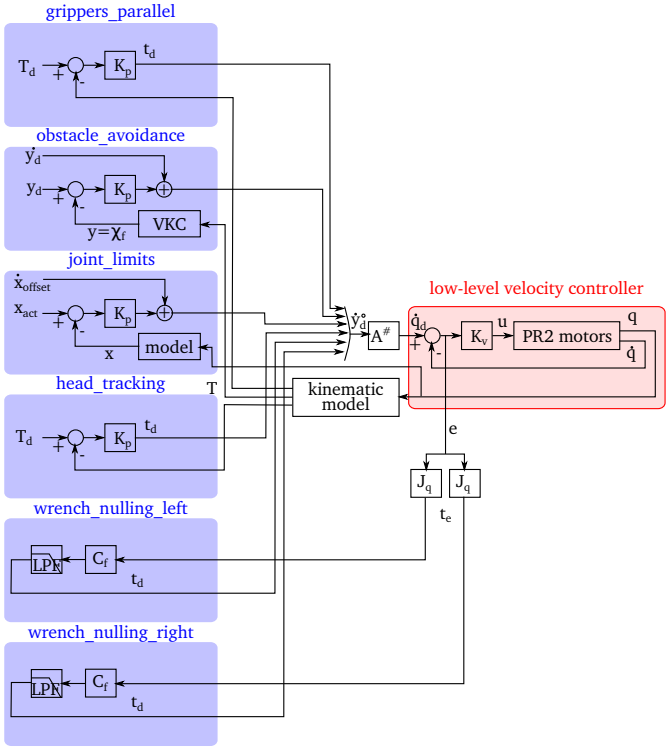


Fig. 3. Control scheme of the full application.

In this application, all tasks have the same priority. The robot joint weights are well-chosen to achieve natural human-robot comanipulation behaviour: putting higher weights on the PR2's base DOF than on the PR2's arm and head DOF, will favour arm and head motions over base motions. As a consequence, when pulling an arm of the robot, the robot will tend to move its arms first and only when approaching joint limits, its base. Section 5 elaborates on the interaction of the different tasks.

The resulting desired joint velocities $\dot{\boldsymbol{q}}_d$ are sent to the PR2's standard low-level velocity controller of the robot_mechanism_controllers package, as available on ros.org (Willow Garage, 2008).

The following section details the control aspects of the wrench-nulling tasks.

4. FORCE-SENSORLESS WRENCH NULLING

The **wrench-nulling tasks** constrain the Cartesian feature/task space between the robot grippers and the robot base. The resulting output equations are used in the optimization problem of the solver, together with the other active tasks to calculate the desired joint velocities for the low-level velocity controller of the PR2. Hence the wrench-nulling control action operates on this task level and not directly on the low-level velocity controller.

To avoid incorporating expensive force sensors, and since the robot actuators are backdrivable, the force is estimated from the velocity control errors.

To simplify the analysis we first focus on one DOF, i.e. one robot joint (figure 4). The low-level proportional velocity controller has a gain K_v. The joint motor model includes inertia I and damping c. If the operator applies an input wrench d to a hand of the robot, the corresponding robot arm will move, due to its backdriveability. This causes a joint velocity v different from the desired joint velocity v_d, expressed by velocity error $e = v_d - v$. The relation of this velocity error e with input wrench d can be written in the Laplace domain as:

$$e = \frac{(Is + c)v_d - d}{Is + c + K_v}, \tag{5}$$

with s the Laplace operator.

By applying a control law C_f (at task level), a new desired velocity v_d is calculated out of the velocity error[3] e. C_f is chosen in order to reinforce the velocity caused by the input wrench, such that the operator senses less resistance of the robot mechanism, i.e. the robot assists the operator. In practice, C_f is applied after transformation of the velocity error from the joint space to the task space using the jacobian of the robot arm. The desired velocity results from the iTaSC solver and involves a transformation back to joint space. When performed correctly, the transformation from the joint space to the task space and back cancel out, and the transfer function T_e from the input wrench d to the velocity error e is described by,

$$T_e = \frac{e}{d} = \frac{1}{(C_f - 1)(Is + c) - K_v}. \tag{6}$$

Accordingly, the transfer function T_v from the input wrench d to the joint velocity v can be written as

[3] Note that by providing an offset to e in the force feedback loop, the wrench-nulling control scheme can be used to apply a controlled non-zero contact force. This application is however not considered further in this paper.

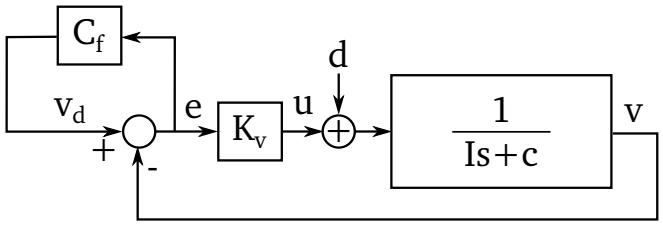


Fig. 4. One DOF wrench-nulling control scheme.

$$T_v = \frac{v}{d} = \frac{1}{Is + c - \frac{K_v}{C_f - 1}}. \tag{7}$$

The pole of T_v equals $(\frac{K_v}{C_f-1} - c)\frac{1}{I}$, which is a real number. The system is stable for any value $C_f < 1$ or $C_f > \frac{K_v+c}{c}$. The latter limit of stability,

$$C_f = \frac{K_v}{c} + 1, \tag{8}$$

describes the situation in which all damping is compensated by the wrench feedback.

The time constant τ of the system is given by

$$\tau = \frac{I}{c - \frac{K_v}{C_f-1}}, \tag{9}$$

and the gain A by

$$A = \frac{1}{c - \frac{K_v}{C_f-1}}. \tag{10}$$

Figure 5 shows the time constant τ and gain A in function of C_f. Both differ only by a factor equal to the inertia I. Consequently, the choice of C_f is a trade-off between a higher gain and a slower response versus a lower gain and a faster response.

A closer look on equation (9) shows that the time constant of the open-loop system τ_{OL}, i.e. when $K_v = 0$, is the asymptote for τ, i.e. when C_f approaches infinity. Furthermore, the time constant of the low-level velocity loop can be found at $C_f = 0$. Similar conclusions hold for the gain. Figure 5 shows the low-level velocity loop time constant τ_{VL} and gain A_{VL} by the black dash-dotted line. The open-loop time constant τ_{OL} and gain A_{OL} are shown by the magenta dash-dotted line.

We choose a gain A higher than the gain of the low-level velocity loop A_{VL}, as a result increasing the sensitivity to the input wrench and hence creating robot assistance. We choose the time constant τ smaller than the open-loop time constant τ_{OL}, hence a faster response than open-loop. Therefore, the area of interest lies between the dash-dotted lines on figure 5. Remark that positive values for C_f do not satisfy the desired behavior, since they have a slower response time than the open-loop system.

5. RESULTS

The bimanual human-robot comanipulation application consists of the joint execution of all described tasks: wrench-nulling, keep grippers parallel, keep visual contact with the operator, avoid joint limits, and avoid obstacles. A video (Vanthienen et al., 2012) proves the performance of the full task with all described constraints active in different scenarios.

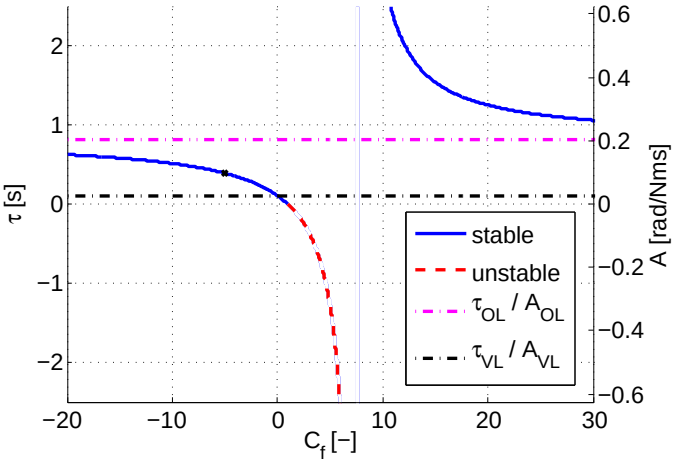


Fig. 5. Time constant τ and gain A in function of control factor C_f. The following parameters apply for the joint: damping $c = 0.32\frac{Nms}{rad}$, inertia $I = 0.25\frac{Nms^2}{rad}$, and velocity controller gain $K_v = 2\frac{Nms}{rad}$. The red dashed line indicates the unstable part with negative gain values. The part of the curve between the low-level velocity loop gain A_{VL} and the open loop time constant τ_{OL}, marked by the black dash-dotted line and magenta dash-dotted line respectively, indicates the area of interest. The cross marks the time constant and amplitude for the chosen $C_f = -5$.

This section provides quantitative results of some key interactions. To this end, we have set up a reduced, but repeatable experiment involving a subset of tasks: avoid joint limits, wrench-nulling, and obstacle avoidance. While the second task is always active, the first and the third task show the ability to activate and deactivate, as well as to change the weights of different constraints in a stable way.

5.1 Experimental setup

A cable connects a mass of 1.5kg to the robot's left hand through a pulley system. When releasing the mass, the weight of the mass pulls the hand with a constant force in a direction approximately aligned with the x-direction. The robot arm joints are far from their limits before applying the force, with the elbow pointing downwards. Figure 6 depicts the experimental setup.

5.2 Joint limit activation

First consider a wrench-nulling task and joint-limits task on the left arm, both with the same priority. When applying a constant force to the robot's left hand, the hand will move in the direction of the applied force. This movement uses the available DOF in a way that depends on the weight and priority of each task. The weight (*penalty*) on the DOF of the robot's mobile base is set higher than the DOF of the arms.

Figure 7 shows the position of the left elbow joint and the x-coordinate of the base over time, after releasing the mass. Both the elbow and the base move, at a ratio determined by the weights on the DOF of the robot. The base does not start moving at the same instant as the arm due to a threshold implemented to prevent nervous base

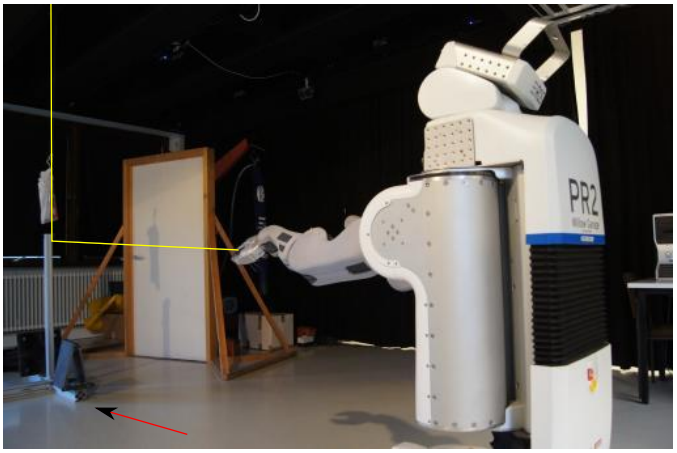


Fig. 6. Experimental setup: the yellow cable and pulley system connects the PR2 robot's hand to the white bag with the weight, shown at the left. The the red arrow indicates the x-direction.

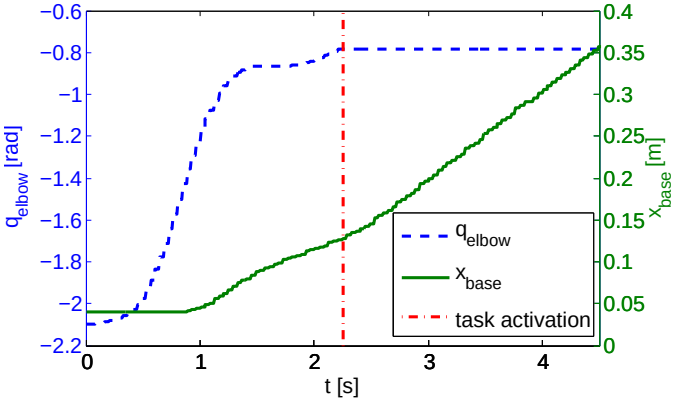


Fig. 7. Elbow and base position over time, in blue dashed and full green line respectively. The red dash-dotted line indicates the moment the joint limit constraint is activated on the left elbow joint.

behavior. The elbow joint reaches its joint limit of -0.82rad after about 2s, as indicated by the red dash-dotted line in figure 7. This event activates the joint-limit task that pushes the joint position away from its limit. The elbow joint finds an equilibrium between the joint-limit task and the wrench-nulling task at -0.78rad. From that moment on, the robot's base satisfies the wrench-nulling constraint by moving faster. Figure 7 shows this faster increase of the base position after the red dash-dotted line.

5.3 Wrench nulling

Equation (6) defines the relation of the error e with the input wrench d for one DOF. A similar relation holds for a direction of the task space, assuming that the transformations between the joint space and the task space cancel out. Applying a constant force in the x-direction to one of the robot hands as in the previous experiment, causes a velocity error e_x in this direction.

A control factor $C_f = -5$ is chosen for all DOF of the task space in this experiment. Figure 8 shows the velocity error e_x over time, after applying this force. The pitch θ^{ee}_{base} of the end effector's orientation with respect to the robot's

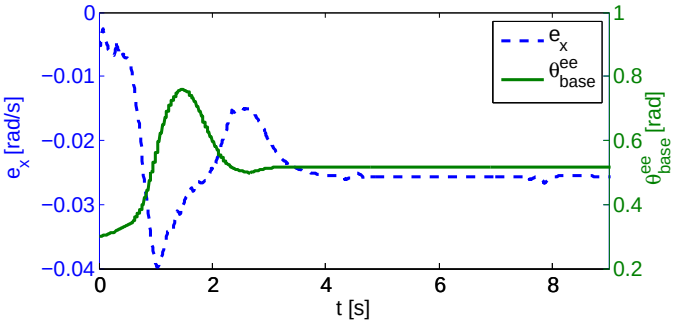


Fig. 8. The velocity error in the x-direction e_x and pitch θ^{ee}_{base} of the end effector with respect to the robot's base, in a blue dashed line and a full green line respectively.

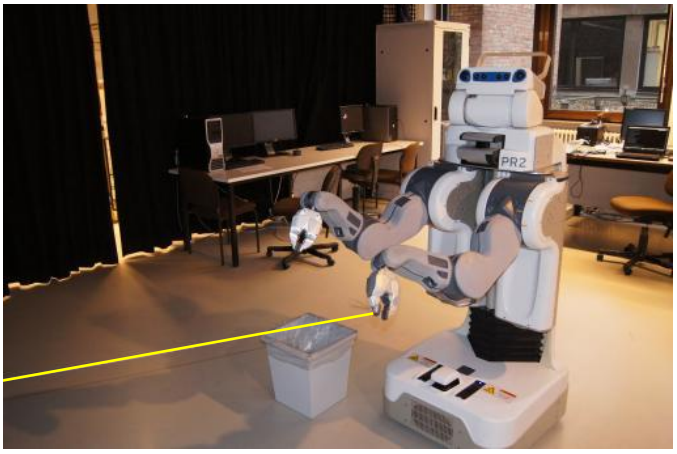


Fig. 9. The PR2 avoiding the recycle bin. The yellow full line is the pulling cable.

base illustrates the transition phase, just after the mass has been released, when the pulled arm makes an up and down movement. This transient is due to the dynamics of each individual joint, as described by equation (6), and the complex geometric interaction between the joints, i.e. our simplified assumptions are not satisfied during transients. After this movement damps out, e_x reaches a constant value. This error is proportional to the applied force, with a negative factor of proportionality, as stated by (6). Hence, applying a constant wrench results in a constant assistance, after transition behavior.

5.4 Obstacle avoidance

An obstacle is added to the setup of the previous experiment at coordinates $(x, y) = (1.3m, -0.1m)$ with respect to $\{w\}$, as shown in figure 9. Adding an obstacle avoidance task, prevents that the robot base collides with the obstacle. The task constrains the robot base to keep a distance r^o of 0.6m when approaching the object closer than 0.5m.

Figure 10 shows the path of the robot in the x, y-plane with a blue dashed line. The red full lines indicate the parts of the path where the obstacle avoidance task was active during the experiment. The green dashed circle segment indicates the obstacle, the magenta dash-dotted circle segment indicates a distance r of 0.5m from the obstacle center. As can be seen, the robot moves first

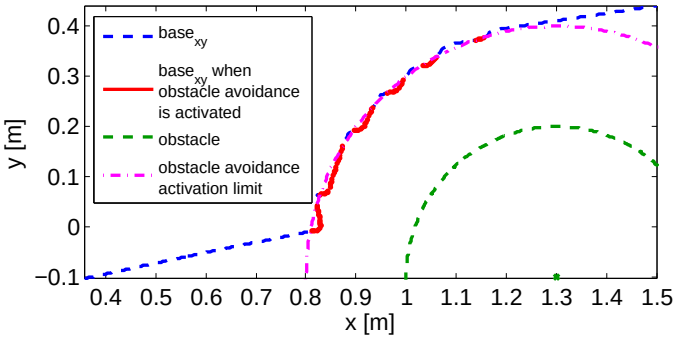


Fig. 10. The blue dashed line indicates the path of the robot base in x, y-plane. The red full lines indicate the parts of the path where the obstacle avoidance task was active. The green dashed circle segment indicates the position of the obstacle, the magenta dash-dotted circle segment the distance r at which the obstacle avoidance task is activated.

along a line in the direction of the applied force. When the robot's base approaches the obstacle, the obstacle-avoidance task is activated and hence the the robot's base avoids a collision, while still trying to follow the wrench and joint-limit constraints.

6. CONCLUSION

This paper demonstrated the iTaSC approach to a force-sensorless and bimanual human-robot comanipulation task. The task comprises 35 constraints in different control spaces for a 20 DOF, tree-structured robot. The task is implemented in a structured way in a reusable software framework. A simple sensorless wrench-nulling control scheme was presented, enabling direct human-robot interaction without the use of a force sensor. The video shows the performance of the full task with all described constraints in different scenarios. Quantitative results are provided for an experiment involving a subset of constraints. The experiments validate the wrench-nulling, the joint-limits, and obstacle avoidance performance. The latter two tasks show the ability to activate and deactivate, as well as to change the weights of different constraints in a stable way.

REFERENCES

Baeten, J., Bruyninckx, H., and De Schutter, J. (2003). Integrated vision/force robotics servoing in the task frame formalism. *Int. J. Robotics Research*, 22(10), 941–954.

Bruyninckx, H. (2001). Open ROBot COntrol Software. `http://www.orocos.org/`. Last visited June 2011.

Bruyninckx, H. and De Schutter, J. (1996). Specification of force-controlled actions in the "Task Frame Formalism": A survey. *IEEE Trans. Rob. Automation*, 12(5), 581–589.

De Schutter, J., De Laet, T., Rutgeerts, J., Decré, W., Smits, R., Aertbeliën, E., Claes, K., and Bruyninckx, H. (2007). Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty. *Int. J. Robotics Research*, 26(5), 433–455.

De Schutter, J. and Van Brussel, H. (1988). Compliant Motion I, II. *Int. J. Robotics Research*, 7(4), 3–33.

Decré, W., Smits, R., Bruyninckx, H., and De Schutter, J. (2009). Extending iTaSC to support inequality constraints and non-instantaneous task specification. In *Int. Conf. Robotics and Automation*, 964–971. Kobe, Japan.

Eom, K.S., Suh, I.H., Chung, W.K., and Oh, S.R. (1998). Disturbance observer based force control of robot manipulator without force sensor. In *Int. Conf. Robotics and Automation*, 3012–3017. Leuven, Belgium.

Hogan, N. (1985). Impedance control: An approach to manipulation. Parts I-III. *Trans. ASME J. Dyn. Systems Meas. Control*, 107, 1–24.

Hogan, N. (1987). Stable execution of contact tasks using impedance control. In *Int. Conf. Robotics and Automation*, 1047–1054. Raleigh, NC.

Katsura, S., Matsumoto, Y., and Ohnishi, K. (2007). Modeling of force sensing and validation of disturbance observer for force control. *IEEE Trans. Industrial Electronics*, 54(1), 530 –538. doi:10.1109/TIE.2006.885459.

Kazerooni, H. (1989). On the robot compliant motion control. *Trans. ASME J. Dyn. Systems Meas. Control*, 111, 416–425.

Klotzbuecher, M. (2011). rfsm statecharts. `http://www.orocos.org/rFSM`. Last visited January 2012.

Klotzbuecher, M., Soetens, P., and Bruyninckx, H. (2010). OROCOS RTT-Lua: an Execution Environment for building Real-time Robotic Domain Specific Languages. In *Int. Workshop on DYn. languages for RObotic and Sensors*, 284–289. URL `https://www.sim.informatik.tu-darmstadt.de/simpar/ws/sites/DYROS2010/03-DYROS.pdf`.

Mansard, N., Stasse, O., Evrard, P., and Kheddar, A. (2009). A versatile generalized inverted kinematics implementation for collaborative working humanoid robots: The stack of tasks. In *Int. Conf. Advanced Robotics*. Munich,Germany.

Mason, M.T. (1981). Compliance and force control for computer controlled manipulators. *IEEE Trans. on Systems, Man, and Cybernetics*, SMC-11(6), 418–432.

Prassler, E., Bruyninckx, H., Nilsson, K., and Shakhimardanov, A. (2009). The use of reuse for designing and manufacturing robots. Technical report, Robot Standards project. `http://www.robot-standards.eu//Documents_RoSta_wiki/whitepaper_reuse.pdf`.

Radestock, M. and Eisenbach, S. (1996). Coordination in evolving systems. In *Trends in Distributed Systems. CORBA and Beyond*, 162–176. Springer-Verlag.

Samson, C., Le Borgne, M., and Espiau, B. (1991). *Robot Control, the Task Function Approach*. Clarendon Press, Oxford, England.

Tachi, S., Sakaki, T., Arai, H., Nishizawa, S., and Pelaez-Polo, J.F. (1991). Impedance control of a direct-drive manipulator without using force sensors. *Advanced Robotics*, 183–205.

Vanthienen, D., De Laet, T., Decré, W., Bruyninckx, H., and De Schutter, J. (2012). `http://people.mech.kuleuven.be/~dvanthienen/SYROCO2012/`.

Vanthienen, D., De Laet, T., Smits, R., and Bruyninckx, H. (2011). itasc software. `http://www.orocos.org/itasc`. Last visited January 2012.

Willow Garage (2008). Robot Operating System (ROS). `http://www.ros.org`. Last visited 2011.