

Unified Constraint-Based Task Specification for Complex Sensor-Based Robot Systems

Joris De Schutter, Johan Rutgeerts, Erwin Aertbeliën, Friedl De Grootte,
Tinne De Laet, Tine Lefebvre, Walter Verdonck, Herman Bruyninckx
Dept. of Mechanical Engineering, K.U.Leuven, Belgium
johan.rutgeerts@mech.kuleuven.ac.be

Abstract—This paper presents a unified task specification formalism and a unified control scheme for the lowest control level of sensor-based robot tasks. The formalism is based on: (i) the integration of any sensor that provides (direct or indirect) distance (and time derivatives) and force information; (ii) the possibility to use multiple “Tool Centre Points”, e.g. defined relative to the robot end effector, other links or the environment; (iii) the integration of optimization functions for underconstrained as well as overconstrained specifications with linear constraints; (iv) the integration of on-line estimators; and (v) compatibility with all major low-level control approaches.

The unified formalism applies to the whole range from industrial manipulators over cooperating robots to humanoid robots, and from pure position control tasks over industrial processes to interaction between a humanoid robot and its environment.

I. INTRODUCTION

Until about a decade ago, most effort in robotics research went into the development of better robot arms and better low-level controllers. This has culminated in very mature control approaches, such as hybrid force/position control [11], impedance control [9], parallel force control [5], and the operational space control of manipulators and humanoids [?], [10]. The last decade has seen a lot of research on mobile and humanoid systems, and in the integration of various sensors into the controllers; this has led to much progress in “intelligent” robot control, for single as well as mobile and multiple robot systems.

The focus of this paper is on *task specification*, a component that has known much less research efforts. This paper presents a specification framework that takes into account *all* components of the above-mentioned complex robot systems (limited to the lowest, non-decision making level): different types of inner loops (position/velocity or torque), (dynamic) interaction with the environment (via contact, or via non-contact “processes” such as welding or laser cutting), model-based estimation and control adaptation; and redundant and/or overconstrained robot systems.

The presented specification framework provides: (1) *Unification* of the existing, controller-oriented specification approaches, such as Mason’s Compliance Frame or Task Frame [4], the Tool Centre Point motion specification, or Hogan’s desired end-effector impedance; (2) *Optimization* using the degrees of freedom that have not been fully specified by the abstract task, or making optimal trade-offs between conflicting specifications.

Constraint-based specification is a natural programming paradigm to reach these goals. In the first place, because it provides *late binding*: adding a constraint to a specification does not rigidly fix from the start (a subset of) the robots’ available degrees of freedom, such that on-line arbitration between conflicting constraints remains possible, as well as optimization of all degrees of freedom that have received (implicit) “*don’t care*” specifications. In addition, constraint-based specification facilitates *minimal coupling*: a constraint involves only the degrees of freedom that are relevant to the constraint, without imposing decisions on the other degrees of freedom. And finally, constraint-based programming facilitates *minimal specification*, in the sense that it allows programmers to, for example, specify a direction of motion without having to specify the magnitude of the velocity in that direction.

Seminal theoretical work in constraint-based programming of robot tasks was done by Popplestone and coworkers [1], [13], and Samson and coworkers [14].

In his work, Popplestone specifies geometric relations (i.e. constraints) between features of objects, to infer the relative *position* of these objects (using *analytic* methods). To support the programmer with the specification of these constraints, feature frames and object frames are introduced. Here, a similar approach with frames as support tools is presented to specify constraints, not only on the position but more general on the desired relative *motion* (i.e. velocity and acceleration), using *numeric* methods.

Samson and coworkers work out an analytic approach to constraint based task specification, but do not provide the tools to support practical application. In this work their approach is combined with Popplestone’s approach to specify geometric constraints between (features of) objects using numerical algorithms.

Generalized pseudo-inverse based task specification of redundant robots is another domain where constraints have since long been adopted as the natural syntax for the job, [8], [12]. All above-mentioned work focused mainly on theory, with little attention for on-line sensor-based estimation or real world experiments. Hogan’s impedance control is another example of constraint-based programming (with the focus mainly on the control aspects): the specified impedance imposes a desired relationship (“constraint”) between motion and force of the robot interacting with the environment, without specifying the magnitudes of forces.

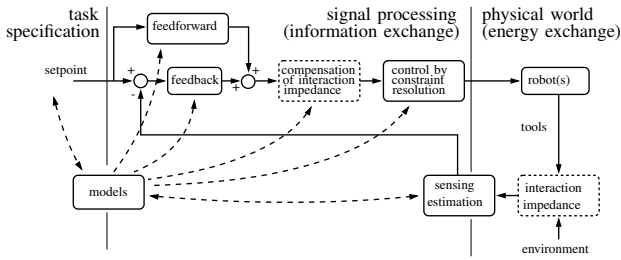


Fig. 1. Generic control scheme. The dashed components only enter in case of contact between tool(s) on the robot and objects in the environment.

Traditional TCP-based industrial robot programming, as well as the hybrid and parallel force control approaches, are much less flexible in their specifications: they impose desired setpoints in all six degrees of freedom of one single robot end-effector frame, thereby making it very difficult to add further motion specifications later on.

Fig. 1 shows the generic control paradigm underlying the presented specification framework. The *off-line* task specification inputs (left-hand side of the Figure) are based on *explicit models* of the robot and its interaction with the environment; these same models are used in all *on-line* signal processing components (sensing, estimation, feedback, interaction impedance compensation, and optimal control setpoint generation). These interactions between model and components are depicted with dashed arrows in Fig. 1.

The paper is structured as follows. Section II provides tools to specify *geometric* task constraints as well as a numerical procedure to translate these task constraints to constraints on the robot motion. Section III does the same for *dynamic* task constraints. Section IV reveals the connection between these task specifications and the most common hybrid control approaches. Section V pays attention to the resolution of redundancy or conflict in the task specification. Section VI presents experimental results. Finally, Section VII gives a brief overview of possible applications of the approach. More details and applications can be found on the accompanying website (<http://www.mech.kuleuven.ac.be/pma/research/manip/>).

II. SPECIFICATION OF GEOMETRIC CONSTRAINTS

A. Introduction

The motion part of robot tasks can be specified in terms of geometric constraints. These geometric constraints allow the user to specify the desired value or time evolution of a geometric quantity that expresses the relative position between features of objects in the robot workspace. Examples of such tasks are a laser cutting application, in which the relative position of the laserspot on a plate is the relevant geometric quantity, or a spray painting application, where the relative pose between the tool and the painted object is controlled.

Each of these geometric quantities z_i is a function of the robot pose (i.e. of the joint positions \mathbf{q}) and of the geometrical properties \mathbf{x}_g of the robot and the relevant

objects:

$$z_i = f_i(\mathbf{q}, \mathbf{x}_g) \quad (1)$$

For velocity resolved control purposes¹, the differential relationship is needed:

$$\frac{dz_i}{dt} = \frac{\partial f_i}{\partial \mathbf{q}} \frac{d\mathbf{q}}{dt} + \frac{\partial f_i}{\partial \mathbf{x}_g} \frac{d\mathbf{x}_g}{dt}, \quad (2)$$

so each specification corresponds to a linear constraint on the joint velocities. By combining all the specifications, and if the geometrical properties are constant, this results in:

$$\dot{\mathbf{z}} = \frac{\partial \mathbf{f}}{\partial \mathbf{q}} \dot{\mathbf{q}} = \mathbf{J}_Z \dot{\mathbf{q}}. \quad (3)$$

\mathbf{J}_Z corresponds to the *task function* in [14] and is called the *geometric constraint jacobian* furtheron. By solving (or optimizing) this set of linear equations, the joint velocities are obtained (see Section V).

For simple tasks and robot systems, it is possible to derive and differentiate the geometric functions f_i analytically. For more complex tasks and robots however, deriving the constraint jacobian \mathbf{J}_Z analytically becomes tedious. The next paragraph presents a numerical approach to calculate \mathbf{J}_Z , without the need to derive \mathbf{f} .

B. Approach

Each row of \mathbf{J}_Z corresponds to one constraint, and each constraint is a specification for the relative motion of a feature of an object with respect to other features or objects.

For each of these objects and features, a frame is introduced (Fig. 2): the world frame $\{\mathbf{w}\}$, the object frames $\{\mathbf{obj}_1\}$ and $\{\mathbf{obj}_2\}$, the feature frames $\{\mathbf{f}_1\}$ and $\{\mathbf{f}_2\}$ and the robot end effector frame $\{\mathbf{ee}\}$. The relative motion between features and objects can be expressed in terms of relative twists between these frames.

These relative twists are constrained by the twist closure equation (written here in a coordinate free way):

$$\mathbf{t}_{ee,w} + \mathbf{t}_{obj_2,ee} + \mathbf{t}_{f_2,obj_2} + \mathbf{t}_{f_1,f_2} - \mathbf{t}_{obj_1,w} - \mathbf{t}_{f_1,obj_1} = \mathbf{0}, \quad (4)$$

where $\mathbf{t}_{i,j}$ represents the relative twist of frame $\{i\}$ with respect to frame $\{j\}$, $\mathbf{t} = [\mathbf{v}^T \boldsymbol{\omega}^T]^T$. Since $\mathbf{t}_{ee,w} = \mathbf{J}_E \dot{\mathbf{q}}$, with \mathbf{J}_E the kinematic robot Jacobian, (4) represents a linear equation in $\dot{\mathbf{q}}$.

In order to perform numerical operations on (4), coordinates have to be introduced. Each relative twist $\mathbf{t}_{i,j}$ is most easily expressed with either the origin of frame i or j as the velocity reference point, and with either frame i or frame j as the reference frame. In order to add the different terms in (4) they have to be transformed to a common velocity reference point and a common reference frame. This is done by premultiplying the twists with the reference point transformation matrix \mathbf{M} and the screw projection matrix \mathbf{P} (see appendix).

A row of \mathbf{J}_Z is then derived as follows. We choose as the common velocity reference point and reference frame

¹Section IV discusses the link with other control approaches.

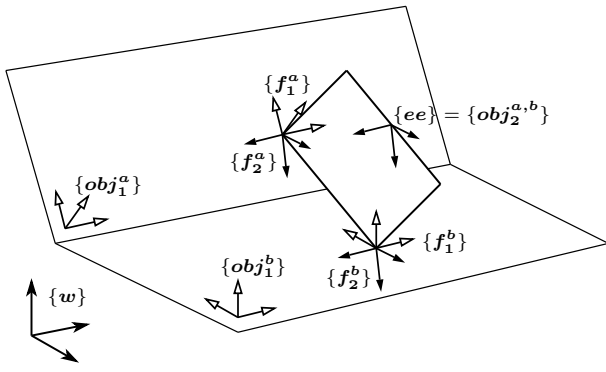


Fig. 2. Incompatible seam following.

the ones in which the geometric constraint is most easily expressed as a constraint on a component (or a linear combination of components) of the twist vector. Solving (4) for the twist term for which a constraint is specified and applying the constraint, yields a row of \mathbf{J}_Z .

C. Example: Incompatible seam following

This first example describes the *incompatible seam following* task, Fig. 2: a tool makes two vertex-face contacts with the seam, which consists of two intersecting planes. The task is called ‘incompatible’ because the tool has a different geometry than the seam.

The goal for the tool is to maintain the contacts, to maintain a perpendicular orientation to the seam and to move along the seam in a specified way.

For each of the contact points (*a* and *b*), the relevant frames are defined as follows:

- $\{obj_1\}$ has its origin in and its z-axis perpendicular to the contact plane,
- $\{f_1\}$ is parallel to $\{obj_1\}$, but has its origin in the contact point,
- $\{f_2\}$ is parallel to $\{ee\}$, but has its origin in the contact point,
- $\{obj_2\}$ coincides at all times with $\{ee\}$ in this example.

The distance of the vertices to the planes (z_1 and z_2) must be zero to maintain the contacts. This distance is measured normal to the plane. This is most easily expressed in $\{f_1^a\}$ and $\{f_1^b\}$.

The perpendicular orientation to the seam can be expressed as a desired value of two Euler angles of $\{f_2^a\}$ with respect to $\{f_1^a\}$ (z_3 and z_4).

The motion along the seam is then defined by the trajectory of the contact point *a* in its contact plane, most easily expressed in $\{f_1\}$ (z_5 and z_6).

The twist closure equation, with as velocity reference point the origin of $\{f_1^a\}$ and reference frame $\{obj_1^a\}$ is given by:

$$\begin{aligned} \mathbf{t}_{f_1^a, obj_1^a} &= \mathbf{t}_{ee, w} + \mathbf{t}_{f_2^a, ee} + \mathbf{t}_{f_1^a, f_2^a} - \mathbf{t}_{obj_1^a, w} \\ \Rightarrow \begin{bmatrix} \dot{z}_5 \\ \dot{z}_6 \\ \dot{z}_1 \\ 0 \\ 0 \\ 0 \end{bmatrix} &= \begin{matrix} ee \\ f_1^a \end{matrix} PM_{f_1^a}^{ee} \mathbf{J}_E \dot{\mathbf{q}} + \mathbf{0} + \begin{matrix} f_2^a \\ f_1^a \end{matrix} PM_{f_1^a}^{f_2^a} \begin{bmatrix} 0 \\ 0 \\ 0 \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} - \mathbf{0} \quad (5) \end{aligned}$$

By selecting the first three rows, the coefficients of \mathbf{J}_Z are obtained for the constraints on z_5 , z_6 and z_1 . In a similar way to the constraint on z_1 , the one on z_2 is obtained.

Note that (5) introduces three variables: ω_x , ω_y and ω_z . These disappear however when selecting the necessary rows from (5).

For the constraints on z_3 and z_4 , the velocity reference point is the origin of $\{f_1^a\}$ and the reference frame is $\{f_2^a\}$:

$$\begin{aligned} \begin{bmatrix} I & 0 \\ 0 & A \end{bmatrix} \mathbf{t}_{f_1^a, f_2^a} &= \begin{bmatrix} I & 0 \\ 0 & A \end{bmatrix} (\mathbf{t}_{obj_1^a, w} + \mathbf{t}_{f_1, obj_1} - \mathbf{t}_{ee, w} - \mathbf{t}_{f_2^a, ee}) \\ \Rightarrow \begin{bmatrix} 0 \\ 0 \\ 0 \\ \dot{z}_3 \\ \dot{z}_4 \\ \dot{\gamma} \end{bmatrix} &= \begin{bmatrix} I & 0 \\ 0 & A \end{bmatrix} \left(\begin{matrix} obj_1^a \\ f_2^a \end{matrix} PM_{f_1^a}^{obj_1^a} \begin{bmatrix} \dot{z}_5 \\ \dot{z}_6 \\ \dot{z}_1 \\ 0 \\ 0 \\ 0 \end{bmatrix} - \begin{matrix} ee \\ f_2^a \end{matrix} PM_{f_1^a}^{ee} \mathbf{J}_R \dot{\mathbf{q}} \right), \quad (6) \end{aligned}$$

where $[\dot{\alpha} \ \dot{\beta} \ \dot{\gamma}]^T = \mathbf{A} [\dot{\omega}_x \ \dot{\omega}_y \ \dot{\omega}_z]^T$, α, β, γ are ZYX Euler angles, and \mathbf{A} represents the linear relation between derivatives of Euler angles and velocities about *X*, *Y* and *Z*.

By selecting the fourth and the fifth row of (6), the coefficients of \mathbf{J}_Z are obtained for the constraints on z_3 and z_4 .

D. Constraint w.r.t. other robot links

In (4), relative twists of frames in which the constraints are expressed easily, are introduced. For each constraint, different frames may be relevant. Furthermore, the end effector twist $\mathbf{t}_{ee, w}$ in (4) suggests that all constraints are relative to the end effector, i.e. the last link of the robot. However, there is no need for the end effector frame to be attached to the last link. E.g. any frame on an intermediate link *i* of a serial robot can be used, with the corresponding geometric Jacobian \mathbf{J}_{Ei} , consisting of the first *i* columns of \mathbf{J}_E .

III. SPECIFICATION OF DYNAMIC CONSTRAINTS

A. Introduction

In addition to geometric constraints, many robot tasks need specification of dynamic constraints. These dynamic constraints allow the user to specify the desired value or time evolution of a force interaction between features of objects in the robot workspace. Examples of such tasks are a surface following application, in which the force between the robot tool and a surface is the relevant dynamic quantity to be controlled, or an assembly operation where the interaction forces and torques between workpieces have to be controlled.

Each of the scalar interaction forces F_i is a function of the robot motion and of the geometrical properties of the robot and the relevant objects:

$$F_i = h_i(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \mathbf{x}_g) \quad (7)$$

where h_i represents a generalized impedance function. For control purposes, the differential relationship is needed:

$$\frac{F_i}{dt} = \frac{\partial h_i}{\partial \mathbf{q}} \frac{d\mathbf{q}}{dt} + \frac{\partial h_i}{\partial \dot{\mathbf{q}}} \frac{d\dot{\mathbf{q}}}{dt} + \frac{\partial h_i}{\partial \ddot{\mathbf{q}}} \frac{d\ddot{\mathbf{q}}}{dt} + \frac{\partial h_i}{\partial \mathbf{x}_g} \frac{d\mathbf{x}_g}{dt}. \quad (8)$$

By combining all the specifications, and if the geometrical properties are constant, this results in:

$$\dot{\mathbf{F}} = \frac{\partial \mathbf{h}}{\partial \mathbf{q}} \dot{\mathbf{q}} + \frac{\partial \mathbf{h}}{\partial \dot{\mathbf{q}}} \ddot{\mathbf{q}} + \frac{\partial \mathbf{h}}{\partial \ddot{\mathbf{q}}} \overset{\cdot\cdot}{\mathbf{q}}. \quad (9)$$

$\frac{\partial h}{\partial \dot{q}}$, $\frac{\partial h}{\partial q}$ and $\frac{\partial h}{\partial \ddot{q}}$ correspond to a generalized stiffness, damping and inertia respectively.

Furtheron in this section we consider the case of a pure stiffness². In this case (9) reduces to:

$$\dot{F} = \frac{\partial h}{\partial \dot{q}} \dot{q} = J_F \dot{q}. \quad (10)$$

(10) represents a linear constraint on the joint velocities and can hence be used as a basis for velocity resolved control³. J_F is called the *dynamic constraint jacobian*.

In case of a pure, linear stiffness, the specification of a set of interaction forces is equivalent to the specification of an equal number of deformations, Δx_i . Deformations are geometric quantities; hence we can convert the dynamic specifications F_i to geometric specifications $z_i = \Delta x_i$:

$$F = Kz, \quad (11)$$

where K is a square stiffness matrix.

Differentiating (11) and introducing (3) yields:

$$\dot{F} = K J_Z \dot{q}. \quad (12)$$

I.e., the derivation of the dynamic constraint jacobian J_F has been reduced to the derivation of a geometric constraint jacobian and a stiffness matrix.

The next paragraph presents a numerical approach to calculate J_Z and K .

B. Approach

Two example cases are worked out.

First, consider the case of a number of parallel and independent force interactions between features of objects. Each of these force interactions is characterized by a given local stiffness k_i in the contact point, i.e. $F_i = k_i \Delta x_i = k_i z_i$, where the force acts in the same direction as the deformation.

Using the feature frames⁴ and the procedure outlined in Section II, the linear relationship between \dot{q}_i and the derivative of Δx_i , and hence between \dot{q}_i and \dot{F}_i , is then determined in a straightforward manner, as shown in the example below.

A second case involves perfectly rigid contacts, but with some compliance built into the robot end effector. Based on the approach developed in [7] a minimal set of deformations can be defined which satisfy (11)⁵. In this case the directions of the deformations or their derivatives (twists) do not coincide with the directions of the forces (wrenches).

²Other impedance cases are briefly treated in Section IV.

³Other control approaches are briefly treated in Section IV.

⁴In case of contact, the feature frames correspond to the *contact frames* defined in [3].

⁵This involves modal decoupling of the end effector dynamics which result from the introduction of the passive compliance, as well as modelling the twist space of the individual contacts. Force control takes place in the degenerate modes, modes with resonance frequency equal zero.

C. Example: Incompatible seam following

The incompatible seam following example of section II was purely geometric. To maintain the vertex-face contacts, constraints were specified on the distance from the vertices to the planes. However, if the experiment is to be conducted on a real robot, force control is necessary to maintain a stable contact. If the contact stiffness is located in the contact points, the first approach can be used. (11) expresses the relation between the force specifications and the corresponding geometric specifications (i.e. deformations in the same directions as the force). In this case, the stiffness matrix K is a diagonal matrix, containing the local stiffnesses k_i . The geometric jacobian in (12) can then be calculated as described in section II.

Experimental results are given in section VI.

IV. CONTROL

This section explains how to convert the specifications of geometric and dynamic constraints into control inputs for the robot by taking into account sensor feedback (see Fig. 1). It is assumed that all geometric and dynamic quantities z_i and F_i can be measured either directly or indirectly (estimation).

This section also reveals the connection between the task specification described in Sections II and III on the one hand, and the most common control approaches on the other hand (for an overview of control approaches, see [6]).

A. Velocity resolved control

In this control approach the control inputs to the robot consist of desired joint velocities. These desired joint velocities are solved from the combined set of constraint equations (3) and (12). However, in order to include feedback, the left hand side of (3) is replaced by

$$\dot{z} = \dot{z}_{ff} + \dot{z}_{fb}, \quad (13)$$

where the feedforward part consists of the desired rate of change as specified by the user, and the feedback part is given by

$$\dot{z}_{fb} = K_{PZ}(z_d - z_m). \quad (14)$$

K_{PZ} is a diagonal matrix of proportional feedback gains; z_d and z_m contain the desired and measured z -values respectively. Similarly, the left hand side of (12) is replaced by

$$\dot{F} = \dot{F}_{ff} + \dot{F}_{fb} = \dot{F}_{ff} + K_{PF}(F_d - F_m). \quad (15)$$

B. Acceleration resolved control

In this control approach the control inputs to the robot consist of desired joint accelerations.

The desired joint accelerations are solved from the combined set of constraint equations (3) and (12). To this end, and in the case of a pure stiffness, (3) and (12) are first differentiated yielding:

$$\ddot{z} = J_Z \ddot{q} + \dot{J}_Z \dot{q}, \quad (16)$$

and

$$\ddot{\mathbf{F}} = \mathbf{K}\{\mathbf{J}_Z\ddot{\mathbf{q}} + \dot{\mathbf{J}}_Z\dot{\mathbf{q}}\}. \quad (17)$$

In addition, in order to include feedback, the left hand side of (16) is replaced by

$$\ddot{\mathbf{z}} = \ddot{\mathbf{z}}_{ff} + \ddot{\mathbf{z}}_{fb}, \quad (18)$$

where the feedforward part consists of the desired acceleration of the geometric quantity as specified by the user, and the feedback part is given by

$$\ddot{\mathbf{z}}_{fb} = \mathbf{K}_{PZ}(\mathbf{z}_d - \mathbf{z}_m) + \mathbf{K}_{DZ}(\dot{\mathbf{z}}_d - \dot{\mathbf{z}}_m). \quad (19)$$

\mathbf{K}_{DZ} is a diagonal matrix of derivative feedback gains. Similarly, the left hand side of (17) is replaced by

$$\ddot{\mathbf{F}} = \ddot{\mathbf{F}}_{ff} + \mathbf{K}_{PF}(\mathbf{F}_d - \mathbf{F}_m) + \mathbf{K}_{DF}(\dot{\mathbf{F}}_d - \dot{\mathbf{F}}_m). \quad (20)$$

Remarks. (i) One way to obtain $\dot{\mathbf{J}}_Z$ is to calculate the time derivative of \mathbf{J}_Z numerically. (ii) If the time derivatives of measurements $\dot{\mathbf{z}}_m$ and $\dot{\mathbf{F}}_m$ are too noisy, they can be approximated by substituting the measured joint velocities into (3) and (12). (iii) In case there is damping in addition to stiffness, the second term of (9) does not equal zero. Hence, instead of differentiating (12), (9) is used to express a constraint in terms of $\ddot{\mathbf{q}}$, with its left hand side replaced by (15) as in velocity resolved control.

C. Torque based control

In this control approach the control inputs to the robot are generated directly in the joint torque space [?].

First, the accelerations of the geometric quantities $\ddot{\mathbf{z}}$, given by (18) and (19), are converted to the required joint torques:

$$\boldsymbol{\tau} = \mathbf{J}_Z^T \boldsymbol{\Lambda}_Z (\ddot{\mathbf{z}} - \dot{\mathbf{J}}_Z \dot{\mathbf{q}}) + \mathbf{C}(\dot{\mathbf{q}}, \mathbf{q}) + \mathbf{g}(\mathbf{q}), \quad (21)$$

where $\boldsymbol{\Lambda}_Z = (\mathbf{J}_Z \mathbf{M}^{-1} \mathbf{J}_Z^T)^{-1}$ is the system inertia matrix $\mathbf{M}(\mathbf{q})$ projected on the task space. $\mathbf{C}(\dot{\mathbf{q}}, \mathbf{q})$ represents the coriolis and centrifugal terms and $\mathbf{g}(\mathbf{q})$ represents gravity.

A second contribution to the joint torques results from the control of the dynamic constraints. This control generates a force to be applied by the robot in the direction of the dynamic constraint and with magnitude:

$$\mathbf{F} = \mathbf{F}_{ff} + \mathbf{F}_{fb}, \quad (22)$$

where \mathbf{F}_{ff} represents the desired interaction force and \mathbf{F}_{fb} represents the output of the feedback controller. Control force \mathbf{F} is projected onto the joint torques by

$$\boldsymbol{\tau} = \mathbf{J}_{E/F}^T \mathbf{F}, \quad (23)$$

where the i -th column of $\mathbf{J}_{E/F}^T$ contains the contributions of unit robot joint velocities to a velocity in the direction of force F_i .

V. REDUNDANCY AND CONFLICT RESOLUTION

The geometric and dynamic constraints specified by the user do not always define the robot task unambiguously. The task may be *underconstrained*, in which case there is some redundancy, or *overconstrained*, in which case the constraints are mutually conflicting. The techniques to solve both redundancy and conflicting constraints are well known, see e.g. [12].

For *redundancy resolution*, the weighting of the generalized inverse of the task jacobian determines the redundant joint motion/position. This weighting is done in joint space. A second specification of the redundant motion is through the nullspace of the jacobian.

For *conflict resolution*, the weighting of the generalized inverse of the task jacobian determines the task execution error. This weighting is in task space. The null space delivers a way of specifying primary tasks which need to be executed perfectly and secondary tasks which are only executed as far as they do not conflict with the primary task set.

VI. EXPERIMENTAL RESULTS

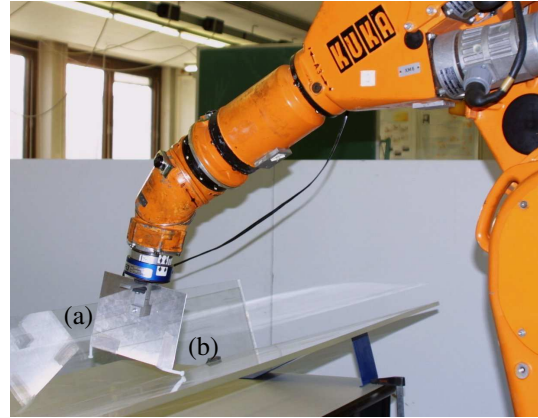


Fig. 3. Experimental setup

Fig. 3 shows the experimental setup for the incompatible seam following task. The Kuka 361 manipulator is controlled by a PC, running RTAI/lxrt and the Orocos control software [2]. Compliance is added in the contact points (a) and (b), so that the force interaction is characterized by the local stiffnesses.

Fig. 4 shows the contact forces. In the first part (10s - 50s), contact is established in both contact points, starting from free space. The desired contact is 10N. In the second part (50s - 80s), the left contact point (b) follows a sinusoidal trajectory in the contact plane (period 10s). The contact forces show a disturbance due to friction and model uncertainties.

VII. OTHER APPLICATIONS

The unified formalism applies to the whole range from industrial manipulators over cooperating robots and robotic hands to humanoid robots, and from pure position control

tasks over industrial processes to interaction between a humanoid robot and its environment.

The accompanying website gives many examples of robot tasks, each time explaining the task specification using constraints and showing a graphical simulation of the task execution. These tasks include: (i) combinations of geometric and dynamic constraints, including constraints specified w.r.t other robot links; (ii) robot systems consisting of single robots, cooperating robots or mobile manipulators; (iii) under- and overconstrained task specifications; (iv) implementations in velocity or acceleration resolved control schemes.

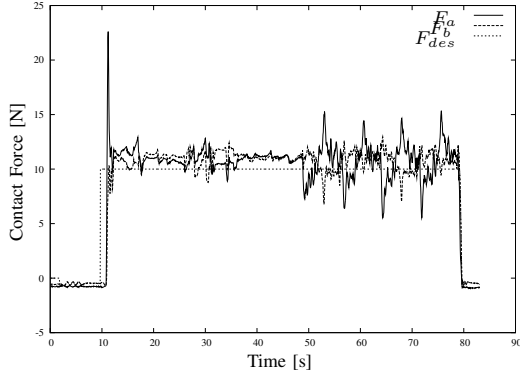


Fig. 4. Contact forces: F_a (contact point a) and F_b (contact point b)

VIII. CONCLUSIONS

A constraint based approach has been presented for task specification of complex sensor-based robot systems. Major contributions of this paper are the development of: (i) tools and procedures to specify the constraints, and (ii) numerical procedures to derive the control inputs for the actuators. The approach is compatible with the most common robot control approaches and benefits from known results in resolution of redundancy and conflicts in the task specification.

Further work includes development of (i) a user interface for machine operators, that masks much of the complexity of the formalism, and is integrated into a CAD-based robot task planning environment; (ii) a library of common tasks, with parameterized specification; (iii) extension to higher control levels.

APPENDIX

Consider two frames $\{i\}$ and $\{j\}$. The vector from the origin of $\{i\}$ to the origin of $\{j\}$ is $\mathbf{p}^{i,j}$ and the rotation matrix, transforming a vector from $\{j\}$ to $\{i\}$ is ${}^j_i\mathbf{R}$.

To change the reference frame of a twist from $\{j\}$ to $\{i\}$, the twist is multiplied by the *screw projection matrix* ${}^j_i\mathbf{P}$:

$${}^j_i\mathbf{P} = \begin{bmatrix} {}^j_i\mathbf{R} & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} & {}^j_i\mathbf{R} \end{bmatrix}.$$

To change the velocity reference point of a twist from the origin of $\{j\}$ to the origin of $\{i\}$, the twist is multiplied

by the *reference point transformation matrix* \mathbf{M}_i^j :

$$\mathbf{M}_i^j = \begin{bmatrix} \mathbf{I}_{3 \times 3} & [\mathbf{p}^{i,j} \times] \\ \mathbf{0}_{3 \times 3} & \mathbf{I}_{3 \times 3} \end{bmatrix},$$

in which $[\mathbf{p}^{i,j} \times]$ is the 3×3 skew-symmetric matrix representing the cross product with $\mathbf{p}^{i,j}$:

$$[\mathbf{p}^{i,j} \times] = \begin{bmatrix} 0 & -p_z^{i,j} & p_y^{i,j} \\ p_z^{i,j} & 0 & -p_x^{i,j} \\ -p_y^{i,j} & p_x^{i,j} & 0 \end{bmatrix}$$

ACKNOWLEDGMENTS

The authors gratefully acknowledge the financial support by K.U.Leuven's Concerted Research Action GOA/99/04 and thank Bart Demarsin for the graphical animations on the website.

REFERENCES

- [1] A. P. Ambler and R. J. Popplestone. Inferring the positions of bodies from specified spatial relationships. *Artificial Intelligence*, 6:157–174, 1975.
- [2] H. Bruyninckx. Open Robot Control Software. <http://www.orocos.org/>.
- [3] H. Bruyninckx. *Kinematic Models for Robot Compliant Motion with Identification of Uncertainties*. PhD thesis, Department of Mechanical Engineering, Katholieke Universiteit Leuven, Leuven, Belgium, 1995.
- [4] H. Bruyninckx and J. De Schutter. Specification of force-controlled actions in the ‘‘Task Frame Formalism’’: A survey. *IEEE Trans. Rob. Automation*, 12(5):581–589, 1996.
- [5] S. Chiaverini and L. Sciavicco. The parallel approach to force/position control manipulators. *IEEE Trans. Rob. Automation*, 9:289–293, 1993.
- [6] J. De Schutter and H. Bruyninckx. Force control of robot manipulators. In *The Control Handbook*, pages 1351–1358. CRC Press, 1996.
- [7] J. De Schutter, D. Torfs, S. Dutr e, and H. Bruyninckx. Invariant hybrid position/force control of a velocity controlled robot with compliant end effector using modal decoupling. *Int. J. Robotics Research*, 16(3):340–356, 1997.
- [8] K. L. Doty, C. Melchiorri, E. M. Schwartz, and C. Bonivento. Robot manipulability. *IEEE Trans. Rob. Automation*, 11(3):462–468, 1995.
- [9] N. Hogan. Impedance control: An approach to manipulation. Parts I-III. *Trans. ASME J. Dyn. Systems Meas. Control*, 107:1–24, 1985.
- [10] O. Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE J. Rob. Automation*, RA-3(1):43–53, 1987.
- [11] M. T. Mason. Compliance and force control for computer controlled manipulators. *IEEE Trans. on Systems, Man, and Cybernetics*, SMC-11(6):418–432, 1981.
- [12] Y. Nakamura. *Advanced robotics: redundancy and optimization*. Addison-Wesley, Reading, MA, 1991.
- [13] R. J. Popplestone, R. Weiss, and Y. Liu. Using characteristic invariants to infer new spatial relationships. In *Int. Conf. Robotics and Automation*, pages 1107–1112, Philadelphia, PA, 1988.
- [14] C. Samson, M. Le Borgne, and B. Espiau. *Robot Control, the Task Function Approach*. Clarendon Press, Oxford, England, 1991.