Constraint-Based Task Specification and Estimation for Sensor-Based Robot Systems in the Presence of Geometric Uncertainty

Joris De Schutter, Tinne De Laet, Johan Rutgeerts, Wilm Decré
Ruben Smits, Erwin Aertbeliën, Kasper Claes and Herman Bruyninckx
Department of Mechanical Engineering, Katholieke Universiteit Leuven
Celestijnenlaan 300B, B3001 Leuven (Heverlee), Belgium
Email: Joris.DeSchutter@mech.kuleuven.be

June 30, 2008

Abstract

This paper introduces a systematic constraint-based approach to specify complex tasks of general sensor-based robot systems consisting of rigid links and joints. The approach integrates both instantaneous task specification and estimation of geometric uncertainty in a unified framework. Major components are the use of feature coordinates, defined with respect to object and feature frames, which facilitate the task specification, and the introduction of uncertainty coordinates to model geometric uncertainty. While the focus of the paper is on task specification, an existing velocity based control scheme is reformulated in terms of these feature and uncertainty coordinates. This control scheme compensates for the effect of time varying uncertainty coordinates. Constraint weighting results in an invariant robot behavior in case of conflicting constraints with heterogeneous units.

The approach applies to a large variety of robot systems (mobile robots, multiple robot systems, dynamic human-robot interaction, etc.), various sensor systems, and different robot tasks. Ample simulation and experimental results are presented.

Keywords: constraint-based programming, task specification, estimation, geometric uncertainty

1 Introduction

Robotic tasks of limited complexity, such as simple positioning tasks, trajectory following or pick-and-place applications in well structured environments, are straightforward to program. For these kinds of tasks extensive programming support is available, as the specification primitives for these tasks are present in current commercial robot control software.

While these robot capabilities already fulfil some industrial needs, research focuses on specification and execution of much more complex tasks. The goal of this research is to open up new robot applications in industrial as well as domestic and service environments. Examples of complex tasks include sensor-based navigation and 3D manipulation in partially or completely unknown environments, using redundant robotic systems such as mobile manipulator arms, cooperating robots, robotic hands or humanoid robots, and using multiple sensors such as vision, force, torque, tactile and distance sensors. Little programming support is available for these kinds of tasks, as there is no general, systematic approach to specify them in an easy way while at the same time dealing properly with geometric uncertainty. As a result, the task programmer has to rely on extensive knowledge in multiple fields such as spatial kinematics, 3D modeling of objects, geometric uncertainty and sensor systems, dynamics and control, estimation, as well as resolution of redundancy and of conflicting constraints.

The goal of our research is to fill this gap. We want to develop programming support for the implementation of complex, sensor-based robotic tasks in the presence of geometric uncertainty. The foundation for this

programming support is a generic and systematic approach, presented in this paper, to specify and control a task while dealing properly with geometric uncertainty. This approach includes a step-by-step guide to obtain an adequate task description. Due to its generic nature, application of the approach does not lead automatically to the most efficient real-time control implementation for a particular application. This is however not the aim of the present paper. Rather we want to present an approach that, with minor adaptations, provides a solution to a large variety of tasks, robot systems and task environments. Software support is being developed based on this approach to facilitate computer assisted instantaneous task specification and to obtain efficient control implementations.

A key idea behind the approach is that a robot task always consists of accomplishing *relative motion* and/or *controlled dynamic interaction* between objects. This paper introduces a set of reference frames and corresponding coordinates *to express* these relative motions and dynamic interactions in an easy, that is, task directed way. The task is then specified by imposing *constraints* on the modeled relative motions and dynamic interactions. The latter is known as *the task function approach* [36] or *constraint-based task programming*. An additional set of coordinates models geometric uncertainty in a systematic and easy way. This uncertainty may be due to modeling errors (for example calibration errors), uncontrolled degrees of freedom in the robotic system or geometric disturbances in the robot environment. When these uncertainty coordinates are included as states in the dynamic model of the robot system, they can be estimated using the sensor measurements and using standard estimation techniques. Subsequently, these state estimates are introduced in the expressions of the task constraints to compensate for the effect of modeling errors, uncontrolled degrees of freedom and geometric disturbances.

Relation to previous work. Previous work on specification of sensor-based robot tasks, such as force controlled manipulation [11,18,19,21] or force controlled compliant motion combined with visual servoing [2], was based on the concept of the *compliance frame* [30] or *task frame* [4]. 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. These setpoints are known as *artificial constraints* [30]. At the controller level, implementation of different control modes in the task frame directions is known as *hybrid control* [35]. The task frame concept has proven to be very useful for the specification of a variety of practical robot tasks. For that reason, several research groups have based their programming and control software to support sensor-based manipulation on this concept [24,39]. However, the drawback of the task frame approach is that it only applies to task geometries with limited complexity, that is, task geometries for which separate control modes can be assigned independently to three pure translational and three pure rotational directions along the axes of a *single* frame.

A more general approach is to assign control modes and corresponding constraints to *arbitrary* directions in the six dimensional manipulation space. This approach, known as *constraint-based programming*, opens up new applications involving a much more complex geometry (for example compliant motion with two- or three-point contacts) and/or involving multiple sensors that control different directions in space simultaneously. In a sense, the *task frame* is replaced by and extended to multiple *feature frames*, as shown in this paper. Each of the feature frames allows us to model *part of* the task geometry in much the same way as in the task frame approach using translational and rotational directions along the axes of a frame. *Part of* the constraints is specified in each of the feature frames. On the other hand, the *total* model consists of a collection of the partial descriptions expressed in the individual feature frames, while the total set of constraints is the collection of all the constraints expressed in the individual feature frames.

Seminal theoretical work on constraint-based programming of robot tasks was done by Ambler and Popplestone [1] and by Samson and coworkers [36]. Also motion planning research in configuration space methods (see [25] for an overview of the literature) specifies the desired relative poses as the result of applying several (possibly conflicting) constraints between object features.

Ambler and Popplestone specify geometric relations (that is, geometric constraints) between features of

two objects. The goal is to infer the desired relative *pose* of these objects from the specified geometric relations between the features. In turn this desired relative pose is used to determine the goal pose of a robot who has to assemble both objects. The geometric relations are based on pose closure equations and they are solved using symbolic methods. To support the programmer with the specification, feature frames and object frames are introduced, as well as suitable local coordinates to express the relative pose between these frames.

Samson and coworkers [36] have put a major step forward in the area of constraint-based programming. They introduce the *task function approach*. The basic idea behind the approach is that many robot tasks may be reduced to a problem of positioning, and that the control problem boils down to regulating a vector function, known as the task function, which characterizes the task. The variables of this function are the joint positions and time. Their work already contains the most important and relevant aspects of constraint-based programming. Based on the task function they propose a general non-linear proportional and derivative control scheme of which the main robotic control schemes are special cases. In addition they analyze the stability and the robustness of this control scheme. They also consider task functions involving feedback from exteroceptive sensors, they recognize the analogy with the kinematics of contact and they treat redundancy in the task specification as a problem of constrained minimization. Finally, they derive the task function for a variety of example applications using a systematic approach. Espiau and coworkers [15] and Mezouar and Chaumette [31] apply this approach to visual servoing and show how this task can be extended to a hybrid task consisting of visual servoing in combination with other constraints, for example, trajectory tracking. Very recent work by Samson [16] presents a framework for systematic and integrated control of *mobile manipulation*, for both holonomic and non-holonomic robots; the scope of that framework is much more focused on only feedback control, while this paper also integrates the instantaneous specification and the estimation of sensor-based tasks.

In addition to constraints on *position* level, constraints on velocity and acceleration level have traditionally been coped with, for example in mobile robotics and multibody simulation. These constraints are expressed using velocity and acceleration closure equations, and they are solved for the robot position, velocity or acceleration using numerical methods that are standard in multibody dynamics. Basically, these numerical methods are of either the *reduction* type (identifying and eliminating the dependent coordinates, see [6] for an overview and further reference) or the *projection* type (doing model updates in a non-minimal set of coordinates and then projecting the result on the allowable motion manifold, see [7] for an overview). The latter approach is better known in robotics as *(weighted) pseudo-inverse control* of redundant robots, [12, 23, 33].

Contributions. This paper proposes a similar approach as in [36], but with a number of extensions and elaborations. First, a set of auxiliary coordinates, denoted as feature coordinates, are introduced to simplify the expression of the task function. Inspired by [1], we introduce feature frames and object frames and we define the auxiliary feature coordinates with respect to these frames in a systematic, standardized way. Whereas Samson relies on the intrinsic robustness of sensor-based control for dealing with uncertainty due to sensors and due to the environment, our approach explicitly models geometric uncertainty. To this end an additional set of uncertainty coordinates is introduced. A generic scheme is proposed to estimate these coordinates and to take them into account in the control loop. This approach results in a better quality of the task execution (higher success rate in case of geometric disturbances as well as smaller tracking errors). The proposed estimation scheme is a generalization of [9] where the scheme was proposed to account for the unknown motion of a contacting surface in 1D force control, and of our work on estimation of geometric uncertainties [10, 26, 27]. The paper also contains the link between constraint-based task specification and real-time task execution control. To this end a velocity based control law [11, 38] is proposed in this paper. However, the development of other control approaches, such as acceleration based control [**?**, 22], without and with rigid constraints [29, 32], proceeds along the same lines. Our approach also considers both redundant and overconstrained task specifications, and uses well-known approaches to deal with these situations [12, 33]. Finally, the paper contains simulation and experimental results of several example applications.

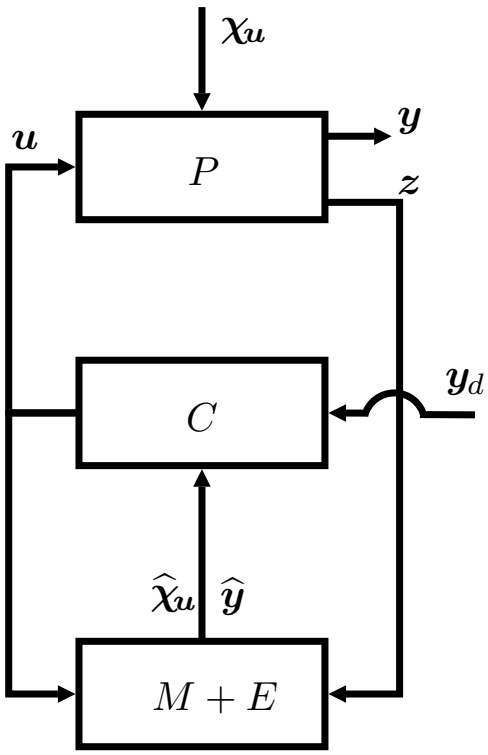


Figure 1: General control scheme including plant P, controller C, and model update and estimation block M+E.

Paper overview. Section 2 proposes a generic overall control and estimation scheme, states the assumptions, defines the variables and introduces the notation. Subsequently, Section 3 introduces a systematic procedure to define feature and uncertainty coordinates. An example illustrates the application of these coordinates throughout the successive steps in this section. Subsequently, Section 4 details the control and estimation scheme for the velocity based control approach. Example applications are presented in Section 5, which also includes simulation and experimental results. Each example illustrates one or more of the main contributions of the proposed framework. Section 6 discusses the proposed approach and points to future work. Finally, Section 7 summarizes the main conclusions.

2 General control and estimation scheme

Figure 1 shows the general control and estimation scheme used throughout this paper. This scheme includes the plant P, the controller C, and the model update and estimation block M+E. The plant P represents both the robot system and the environment. We consider a *single* global robot system. This system however may consist of several separate devices: mobile robots, manipulator arms, conveyors, humanoids, etc. In general the robot system may comprise both holonomic and nonholonomic subsystems. However, for simplicity, the general derivations in Sections 2-4 only consider holonomic systems. A slight modification suffices to apply the general procedures to nonholonomic systems, as explained in footnotes as well as in Section 5.2 for a mobile robot application. The robot system is assumed to be rigid, that is, it consists of rigid links and joints. On the other hand, the environment is either *soft* or *rigid*. In case of contact with a soft environment, the environment deforms under the action of the robot system. In case of contact with a rigid environment, the robot system cannot move in the direction of the constraint. Rigid constraints are known as *natural constraints* [30].

2.1 System variables

The *control input* to the plant is $\boldsymbol{u}$. Depending on the particular control scheme, $\boldsymbol{u}$ represents desired joint velocities, desired joint accelerations, or joint torques.

The *system output* is $\boldsymbol{y}$, which represents the controlled variables. Typical examples of controlled variables are Cartesian or joint positions, distances measured in the real space or in camera images, and contact forces or moments. Task specification consists of imposing constraints to the system output $\boldsymbol{y}$. These constraints take the form of desired values $\boldsymbol{y}_d(t)$, which are in general time dependent.[1] These desired values correspond to the *artificial constraints* defined in [30].

The plant is observed through measurements $\boldsymbol{z}$ which include both proprioceptive and exteroceptive sensor measurements. Note that $\boldsymbol{y}$ and $\boldsymbol{z}$ are not mutually exclusive, since in many cases the plant output is directly measured. For example, if we want to control a distance that is measured using a laser sensor, this distance is contained both in $\boldsymbol{y}$ and $\boldsymbol{z}$. However, not all system outputs are directly measured, and an estimator is needed to generate estimates $\widehat{\boldsymbol{y}}$ (see Section 4.4). These estimates are needed in the control law C.

In general the plant is disturbed by various disturbance inputs. However, in order to avoid overloaded mathematical derivations, this paper only focuses on *geometric disturbances*, represented by coordinates $\boldsymbol{\chi_u}$. These coordinates represent modeling errors, uncontrolled degrees of freedom in the robot system or geometric disturbances in the robot environment. As with system outputs, not all these disturbances can be measured directly, but they can be estimated by including a disturbance observer in the estimator block $M + E$ (see Section 4.4). The control and estimation scheme can be extended in a straightforward manner to include other disturbances, such as disturbance joint torques, output and measurement noise, wheel slip, and so on, see for example 5.2.

Finally, a representation of the *internal state* of the robot system is needed. For a holonomic system, a natural choice for the system state consists of the joint coordinates $\boldsymbol{q}$ and their first time derivatives. However, this paper introduces additional task related coordinates, denoted as *feature coordinates* $\boldsymbol{\chi_f}$, to facilitate the modeling of outputs and measurements by the user. Obviously, a dependency relation exists between the joint coordinates $\boldsymbol{q}$ and feature coordinates $\boldsymbol{\chi_f}$. In this paper we assume all robot joints can be controlled.

2.2 Equations

The *robot system equation* relates the control input $\boldsymbol{u}$ to the rate of change of the robot system state:

$$\frac{d}{dt}\begin{pmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \end{pmatrix} = \boldsymbol{s}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}). \tag{1}$$

On the other hand, the *output equation* relates the position based outputs $\boldsymbol{y}$ to the joint and feature coordinates:

$$\boldsymbol{f}\left(\boldsymbol{q}, \boldsymbol{\chi_f}\right) = \boldsymbol{y}, \tag{2}$$

where $\boldsymbol{f}\left(\right)$ is a vector-valued function. Similarly, the *measurement equation* relates the position based measurements $\boldsymbol{z}$ to the joint and feature coordinates:

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

The *artificial constraints* used to specify the task are expressed as:

$$\boldsymbol{y} = \boldsymbol{y}_d. \tag{4}$$

Furthermore, in case of a rigid environment, the *natural constraints* are expressed as constraints on the joint and feature coordinates:

$$\boldsymbol{g}\left(\boldsymbol{q}, \boldsymbol{\chi_f}\right) = \boldsymbol{0}, \tag{5}$$

[1] In order to avoid overloaded notation we omit time dependency in the remainder of the paper.

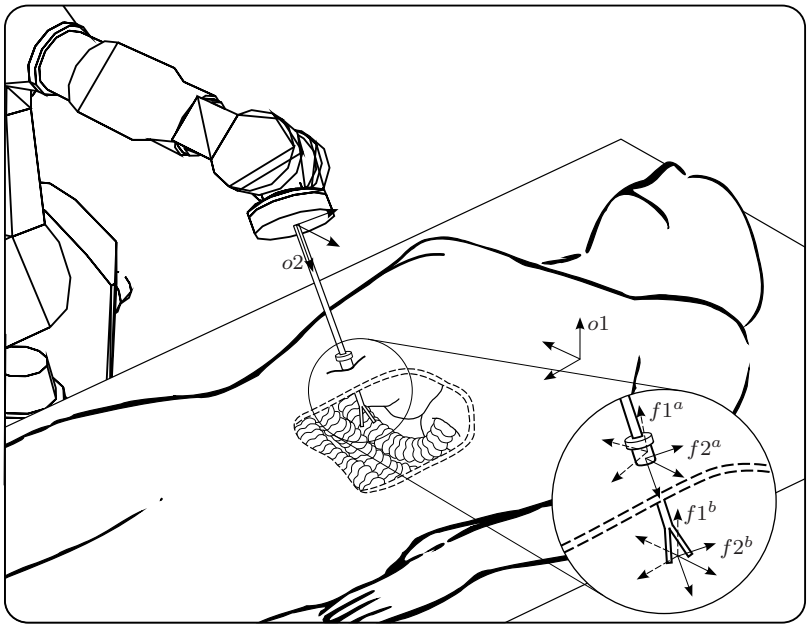


Figure 2: The object and feature frames for a minimally invasive surgery task.

and are therefore a special case of the artificial constraints with $\boldsymbol{y}_d = 0$.

The dependency relation between $\boldsymbol{q}$ and $\boldsymbol{\chi_f}$ is perturbed by the uncertainty coordinates $\boldsymbol{\chi_u}$, and is expressed as:

$$\boldsymbol{l}\left(\boldsymbol{q}, \boldsymbol{\chi_f}, \boldsymbol{\chi_u}\right) = \boldsymbol{0}. \tag{6}$$

Remark that for nonholonomic systems this equation does not exist. In that case, the joint coordinates $\boldsymbol{q}$ are replaced in (6) by operational coordinates $\boldsymbol{\chi_q}$, while a dependency relation exists between $\dot{\boldsymbol{q}}$ and $\dot{\boldsymbol{\chi}}_{\boldsymbol{q}}$. This dependency relation is known as the *nonholonomic constraint*. See also Section 5.2 for more details.

This relation is derived using the position closure equations, that is, by expressing kinematic *loop constraints* and therefore is called *position loop constraint* further on. The benefit of introducing feature coordinates $\boldsymbol{\chi_f}$ is that they can be chosen according to the specific task at hand, such that equations (2)–(5) can much be simplified. A similar freedom of choice exists for the uncertainty coordinates in equation (6). While Section 3 introduces a particular set of feature and uncertainty coordinates to this end, the analyses in Section 4 are more generally valid for *any* set of feature and uncertainty coordinates, provided that the feature coordinates can unambiguously be solved from loop closure equations (6) for given values of the independent variables $\boldsymbol{q}$ and $\boldsymbol{\chi_u}$. This condition requires that the number of independent loop closure equations equals the number of feature coordinates.

3 Task coordinates

This section introduces systematic definitions for feature coordinates (Section 3.2) and uncertainty coordinates (Section 3.3). These coordinates are defined in *object* frames and *feature* frames (Section 3.1) that can be chosen by the task programmer in a way that simplifies the specification of the task at hand. The systematic procedure is illustrated by a (simplified) *minimally invasive surgery* task (Figure 2), in which a robot holds a laparoscopic tool inserted into a patient's body through a hole, known as the trocar. The endpoint of the tool has to move along a specified trajectory in the patient's body.

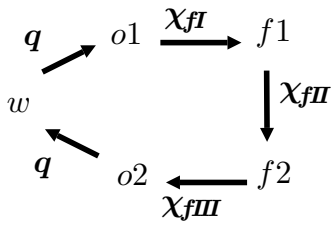


Figure 3: Object and feature frames and feature coordinates.

3.1 Object and feature frames

A typical robot task accomplishes a relative motion and/or controlled dynamic interaction between objects. This section presents a systematic procedure to introduce a set of reference frames in which to express these relative motions and dynamic interactions in an easy way.

The first step is *to identify the objects and features* that are relevant for the task, and to assign reference frames to them. An *object* can be any rigid object in the robot system (for example a robot end effector or a robot link) or in the robot environment. A *feature* is linked to an *object*, and indicates a *physical entity* on that object (such as a vertex, edge, face, surface), or an *abstract geometric property* of a physical entity (such as the symmetry axis of a hollow cylinder, or the reference frame of a sensor connected to the object, for instance a camera). The relative motion or force between two objects is *specified by imposing constraints* on the relative motion or force between one feature on the first object and a corresponding feature on the second object. Each such constraint 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).

Below four rules are given for choosing the features and for assigning object and feature frames.

R1. $o1$ and $o2$ are rigidly attached to the corresponding object.

R2. $f1$ and $f2$ are linked to, but not necessarily rigidly attached to $o1$ and $o2$, respectively.

For an application in 3D space, there are in general six degrees of freedom between $o1$ and $o2$. The connection $o1 \rightarrow f1 \rightarrow f2 \rightarrow o2$ forms a *kinematic chain*, that is, the degrees of freedom between $o1$ and $o2$ are distributed over three submotions: the relative motion of $f1$ with respect to $o1$, the relative motion of $f2$ with respect to $f1$, and the relative motion of $o2$ with respect to $f2$, Figure 3[2]. Rule 3 simplifies the mathematical representation of the submotions:

R3. the origin and the orientation of the frames are chosen as much as possible such that the submotions correspond to motions along or about the axes of one of the two frames involved in the submotion.

These rules do not result in a unique frame assignment, but allow for optimal task-specific choices by the task programmer. Finally, the "world" reference frame is denoted by w throughout the paper.

For the surgery task, a natural task description imposes two motion constraints: (i) the trocar point maintains its desired position, and (ii) the endpoint of the tool performs a specified motion relative to the patient's body. This suggests the use of two feature relationships: one at the trocar point (feature a) and one at the endpoint of the tool (feature b). For both feature relationships, the corresponding objects are the laparoscopic tool attached to the robot mounting plate, and the patient, Figures 2 and 4. The object and feature frames are chosen as follows:

- $o1$ is fixed to the patient.
- $o2$ is fixed to the mounting plate of the robot, with the Z-axis along the laparoscopic tool.

[2]Figure 3 does not hold at position level for nonholonomic systems.

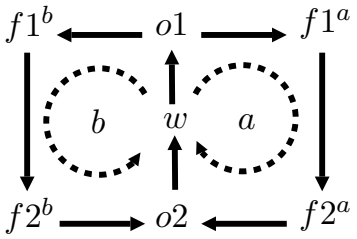


Figure 4: Object and feature frames for minimally invasive surgery task.

- $f1^a$ is rigidly attached to $o1$, with its origin at the constant *desired* position for the trocar point, and its Z-axis normal to the patient.
- $f2^a$ is parallel to $o2$, and has its origin at the *actual* trocar point.
- both $f1^b$ and $f2^b$ have their origin at the endpoint of the tool; $f1^b$ is parallel to $o1$, and $f2^b$ is parallel to $o2$.

Similarly as for the specification of constraints, objects and feature frames are introduced to model sensor measurements, that is, to derive expressions (3). So, in general:

R4. every system output and every measurement is modeled using a feature relationship. However, a single feature might be used to model multiple system outputs and/or measurements. Obviously, if outputs or measurements are expressed using only *joint coordinates*, that is $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{q})$ or $\boldsymbol{z} = \boldsymbol{h}(\boldsymbol{q})$, there is no need for defining such feature relationship.

3.2 Feature coordinates

Objects may or may not be moved by the robot system. For a single global and holonomic robot system with joint position vector $\boldsymbol{q}$, the pose of $o1$ and $o2$ depends in general on $\boldsymbol{q}$, Figure 3[3].

Owing to rule Rule 3, for a given feature relationship the relative positions between respectively $o1$ and $f1$, $f1$ and $f2$, and $f2$ and $o2$, can be represented by position coordinates along the axes of either frame, or by Euler angles expressed in either frame. In case of multiple feature relationships, all these coordinates are collected into a single vector of feature coordinates, $\boldsymbol{\chi_f}$.

In many practical cases every feature submotion can be represented by a minimal set of position coordinates, that is, as many coordinates as there are instantaneous degrees of freedom in the submotion. Hence, for every feature relationship, the relative position between $o1$ and $o2$ is represented by six coordinates, while the total number of feature coordinates is six times the number of feature relationships:

$$\dim(\boldsymbol{\chi_f}) = 6n_f, \tag{7}$$

where n_f is the number of feature relationships.

For every feature relationship a kinematic loop exists, as shown in Figure 3. Expressing position loop closure results in six scalar equations. Hence, the number of feature coordinates equals the number of closure equations:

$$\dim[\boldsymbol{l}(\boldsymbol{q}, \boldsymbol{\chi_f}, \boldsymbol{\chi_u})] = \dim(\boldsymbol{\chi_f}). \tag{8}$$

Consequently, a set of feature coordinates can always be found such that $\boldsymbol{J_f} = \frac{\partial \boldsymbol{l}}{\partial \boldsymbol{\chi_f}}$ is a square and invertible matrix, which is a requirement in the derivation of the control law in Section 4.

If no minimal set of coordinates exists to represent one or more feature submotions, $\boldsymbol{\chi_f}$ will not be minimal. However, a minimal set of coordinates always exists at velocity level, while a dependency relation exists between

[3]For nonholonomic robots, the pose of $o1$ and $o2$ depends on $\boldsymbol{\chi_q}$.

these velocity coordinates and the time derivatives of the nonminimal set of position coordinates. In such case $\boldsymbol{J_f}$ is defined in relation to this minimal set of velocity coordinates. This approach again ensures that $\boldsymbol{J_f}$ is a square and invertible matrix, see Section 5.3 for an example.

For every feature $\boldsymbol{\chi_f}$ can be partitioned as in Figure 3:

$$\boldsymbol{\chi_f} = \begin{pmatrix} \boldsymbol{\chi_{fI}}^T & \boldsymbol{\chi_{fII}}^T & \boldsymbol{\chi_{fIII}}^T \end{pmatrix}^T, \tag{9}$$

in which:

- $\boldsymbol{\chi_{fI}}$ represents the relative motion of $f1$ with respect to $o1$,
- $\boldsymbol{\chi_{fII}}$ represents the relative motion of $f2$ with respect to $f1$, and
- $\boldsymbol{\chi_{fIII}}$ represents the relative motion of $o2$ with respect to $f2$.

The surgery task distributes the six degrees of freedom between $o1$ and $o2$ as follows:

- for feature a:

$$\boldsymbol{\chi_{fI}}^a = (-), \tag{10}$$
$$\boldsymbol{\chi_{fII}}^a = \begin{pmatrix} x^a & y^a & \phi^a & \theta^a & \psi^a \end{pmatrix}^T, \tag{11}$$
$$\boldsymbol{\chi_{fIII}}^a = (z^a). \tag{12}$$

  x^a and y^a represent the x- and y-position of the origin of $f2^a$ with respect to $f1^a$, expressed in $f1^a$, while z^a represents the z-position of the origin of $f2^a$ with respect to $o2^a$, expressed in $o2^a$. ϕ^a, θ^a and ψ^a are ZYX-Euler angles between $f2^a$ and $f1^a$, expressed in either frame.

- for feature b:

$$\boldsymbol{\chi_{fI}}^b = \begin{pmatrix} x^b & y^b & z^b \end{pmatrix}^T, \tag{13}$$
$$\boldsymbol{\chi_{fII}}^b = \begin{pmatrix} \phi^b & \theta^b & \psi^b \end{pmatrix}^T, \tag{14}$$
$$\boldsymbol{\chi_{fIII}}^b = (-). \tag{15}$$

  x^b, y^b and z^b represent the x-, y- and z-position of the origin of $f1^b$ with respect to $o1^b$, expressed in $o1^b$. ϕ^b, θ^b and ψ^b are ZYX-Euler angles between $f2^b$ and $f1^b$, expressed in either frame.

3.3 Uncertainty coordinates

This paper focuses on two types of geometric uncertainty that are typical for an industrial robot with a fixed base executing a task on a workpiece, that is: (i) uncertainty on the pose of an object, and (ii) uncertainty on the pose of a feature with respect to its corresponding object. This second type corresponds to uncertainty in the geometric model of the object. Other types of uncertainty can be handled in a similar way.

Uncertainty *coordinates* are introduced *to represent* the pose uncertainty of a real frame with respect to a modeled frame:

$$\boldsymbol{\chi_u} = \begin{pmatrix} \boldsymbol{\chi_{uI}}^T & \boldsymbol{\chi_{uII}}^T & \boldsymbol{\chi_{uIII}}^T & \boldsymbol{\chi_{uIV}}^T \end{pmatrix}^T, \tag{16}$$

in which (Figure 5):

- $\boldsymbol{\chi_{uI}}$ represents the pose uncertainty of $o1$,
- $\boldsymbol{\chi_{uII}}$ represents the pose uncertainty of $f1$ with respect to $o1$,
- $\boldsymbol{\chi_{uIII}}$ represents the pose uncertainty of $f2$ with respect to $o2$, and

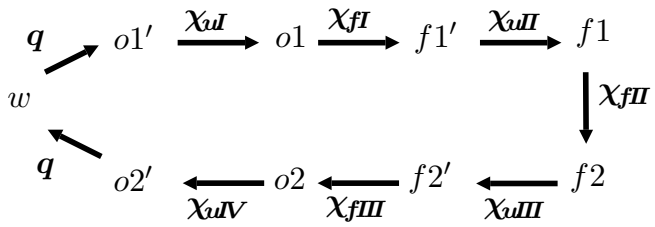


Figure 5: Feature and uncertainty coordinates. The primed frames represent the modelled frame poses while the others are the actual ones.

- χ_{uIV} represents the pose uncertainty of $o2$.

For the surgery task the following uncertainty coordinates might be considered:

- $\chi_{uI}{}^a = \chi_{uI}{}^b = \chi_{uI}$ represents the uncertainty in the position of the patient,
- $\chi_{uIV}{}^a = \chi_{uIV}{}^b = \chi_{uIV}$ represents the uncertainty in the robot model and the fixation of the tool to the robot end effector;

furthermore, for feature a:

- $\chi_{uII}{}^a$ represents the motion of the trocar with respect to the patient,
- $\chi_{uIII}{}^a$ represents the uncertainty in the shape of the tool;

and for feature b:

- $\chi_{uII}{}^b$ represents the motion of the organs with respect to the patient, and
- $\chi_{uIII}{}^b$ represents the uncertainty in the shape of the tool.

3.4 Task specification

A task is easily specified using the task coordinates χ_f and χ_u as defined in Sections 3.2 and 3.3, as shown here for the surgery task. The goal of this task is threefold: (i) the tool has to go through the trocar, (ii) three translations between the laparoscopic tool and the patient are specified, and (iii) if possible, two supplementary rotations may be specified.

Output equations (2) The outputs to be considered for this task are:

$$\begin{array}{llll} y_1 = x^a, & y_2 = y^a, & y_3 = x^b, & \\ y_4 = y^b, & y_5 = z^b, & y_6 = \phi^b, \text{ and} & y_7 = \theta^b. \end{array} \tag{17}$$

Constraint equations (4) The artificial constraints used to specify the task are:

$$y_{1d}(t) = 0, \quad y_{2d}(t) = 0, \tag{18}$$

furthermore $y_{3d}(t)$, $y_{4d}(t)$, $y_{5d}(t)$ and possibly $y_{6d}(t)$ and $y_{7d}(t)$ also need to be specified. In case of a rigid trocar point, x^a and y^a must be identically zero. Hence, the first two artificial constraints (18) represent in fact *natural* constraints.

Position loop constraints (6) There are two position loop constraints, one for each feature relationship. The position loop constraints are easily expressed using transformation matrices based on Figure 5. For instance for feature a this results in:

$$\begin{array}{c} \boldsymbol{T}_{w}^{o1'}(\boldsymbol{q})\ \boldsymbol{T}_{o1'}^{o1}(\boldsymbol{\chi_{uI}})\ \boldsymbol{T}_{o1}^{f1'}(\boldsymbol{\chi_{fI}}^{a})\cdots \\ \boldsymbol{T}_{f1'}^{f1}(\boldsymbol{\chi_{uII}}^{a})\ \boldsymbol{T}_{f1}^{f2}(\boldsymbol{\chi_{fII}}^{a})\ \boldsymbol{T}_{f2}^{f2'}(\boldsymbol{\chi_{uIII}}^{a})\cdots \\ \boldsymbol{T}_{f2'}^{o2}(\boldsymbol{\chi_{fIII}}^{a})\ \boldsymbol{T}_{o2}^{o2'}(\boldsymbol{\chi_{uIV}})\ \boldsymbol{T}_{o2'}^{w}(\boldsymbol{q}) = \mathbf{1}_4, \end{array} \tag{19}$$

where $\boldsymbol{T}_{w}^{o1'}(\boldsymbol{q})$ and $\boldsymbol{T}_{o2'}^{w}(\boldsymbol{q})$ contain the forward kinematics of the robot, while all other transformation matrices are easily written using the definitions of the feature coordinates (Section 3.2) and of the uncertainty coordinates (Section 3.3).

Measurement equations (3) Suppose that the x- and y-position of the patient with respect to the world are measured, for instance with a camera and markers attached to the patient. In this case the measurements are:

$$z_1 = x_u \quad \text{and} \quad z_2 = y_u, \tag{20}$$

where x_u and y_u are two components of $\boldsymbol{\chi_{uI}}$ (Section 3.3).

In another setting x and y forces could be measured at the trocar. If the modeled stiffness at the trocar is k_s, the measurement equations are then written as:

$$z_1 = k_s x^a \quad \text{and} \quad z_2 = k_s y^a. \tag{21}$$

As both types of measurements are easily expressed using the coordinates defined for features a and b, no additional feature has to be considered to model these measurements.

4 Control and estimation

This section works out the details for controller C and model update and estimation block M+E in Figure 1 for the case of velocity based control. The development of other control approaches, such as acceleration based control in case of a soft or rigid environment proceeds along the same lines [8].

First this section derives a closed form expression for the control input $\boldsymbol{u}$. Next, the closed loop behavior is briefly discussed. Then, an approach is outlined for constraint weighting and for smooth transition between successive subtasks with different constraints. Finally, an approach for model update and estimation is given.

4.1 Control law

Output equation (2) is differentiated with respect to time to obtain an output equation at velocity level:

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\chi_f}}\dot{\boldsymbol{\chi}}_{\boldsymbol{f}} = \dot{\boldsymbol{y}}, \tag{22}$$

which can be written as:

$$\boldsymbol{C_q}\dot{\boldsymbol{q}} + \boldsymbol{C_f}\dot{\boldsymbol{\chi}}_{\boldsymbol{f}} = \dot{\boldsymbol{y}}, \tag{23}$$

with $\boldsymbol{C_q} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{q}}$ and $\boldsymbol{C_f} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\chi_f}}$. On the other hand, the relationship between the nonminimal set of variables $\dot{\boldsymbol{q}}$, $\dot{\boldsymbol{\chi}}_{\boldsymbol{f}}$ and $\dot{\boldsymbol{\chi}}_{\boldsymbol{u}}$ is given by the velocity loop constraint which is obtained by differentiating the position loop constraint (6):

$$\frac{\partial \boldsymbol{l}}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} + \frac{\partial \boldsymbol{l}}{\partial \boldsymbol{\chi_f}}\dot{\boldsymbol{\chi}}_{\boldsymbol{f}} + \frac{\partial \boldsymbol{l}}{\partial \boldsymbol{\chi_u}}\dot{\boldsymbol{\chi}}_{\boldsymbol{u}} = \mathbf{0}, \tag{24}$$

or:

$$\boldsymbol{J_q}\dot{\boldsymbol{q}} + \boldsymbol{J_f}\dot{\boldsymbol{\chi}}_{\boldsymbol{f}} + \boldsymbol{J_u}\dot{\boldsymbol{\chi}}_{\boldsymbol{u}} = \mathbf{0}, \tag{25}$$

with $\boldsymbol{J_q} = \frac{\partial \boldsymbol{l}}{\partial \boldsymbol{q}}$, $\boldsymbol{J_f} = \frac{\partial \boldsymbol{l}}{\partial \boldsymbol{\chi_f}}$ and $\boldsymbol{J_u} = \frac{\partial \boldsymbol{l}}{\partial \boldsymbol{\chi_u}}$. $\dot{\boldsymbol{\chi}}_{\boldsymbol{f}}$ is solved from (25), yielding[4]:

$$\dot{\boldsymbol{\chi}}_{\boldsymbol{f}} = -\boldsymbol{J_f}^{-1} \left(\boldsymbol{J_q} \dot{\boldsymbol{q}} + \boldsymbol{J_u} \dot{\boldsymbol{\chi}}_{\boldsymbol{u}} \right). \tag{26}$$

This operation requires that $\boldsymbol{J_f}$ is invertible, that is, the number of feature coordinates must equal the number of independent loop equations. The feature coordinates introduced in Section 3.2 satisfy this requirement. Substituting (26) into (23) yields the modified output equation:

$$\boldsymbol{A} \dot{\boldsymbol{q}} = \dot{\boldsymbol{y}} + \boldsymbol{B} \dot{\boldsymbol{\chi}}_{\boldsymbol{u}}, \tag{27}$$

where $\boldsymbol{A} = \boldsymbol{C_q} - \boldsymbol{C_f} \boldsymbol{J_f}^{-1} \boldsymbol{J_q}$ and $\boldsymbol{B} = \boldsymbol{C_f} \boldsymbol{J_f}^{-1} \boldsymbol{J_u}$ are introduced for simplicity of notation.

In the velocity based approach the plant is assumed to be an ideal velocity controlled system, that is, the robot dynamics are neglected, and the system equation is given by:

$$\dot{\boldsymbol{q}} = \boldsymbol{u} = \dot{\boldsymbol{q}}_d, \tag{28}$$

where the control input $\boldsymbol{u}$ corresponds to the desired joint velocities $\dot{\boldsymbol{q}}_d$.

Constraint equation (4) is also expressed at velocity level. As a result, the constraint equation has to include feedback at position level to compensate for integration errors, modeling errors and disturbances:

$$\dot{\boldsymbol{y}} = \underbrace{\dot{\boldsymbol{y}}_d + \boldsymbol{K}_p \left(\boldsymbol{y}_d - \boldsymbol{y} \right)}_{\dot{\boldsymbol{y}}_d^\circ}, \tag{29}$$

with $\boldsymbol{K}_p$ a matrix of feedback constants (typically, but necessarily, a diagonal matrix with positive constants k_{pi}) and $\dot{\boldsymbol{y}}_d^\circ$ the modified constraint at velocity level. If $\boldsymbol{y}$ cannot be measured directly, it has to be replaced in (29) by its estimate $\widehat{\boldsymbol{y}}$ provided by the estimator.

Applying constraint (29) to (27), while also substituting system equation (28) and replacing $\dot{\boldsymbol{\chi}}_{\boldsymbol{u}}$ by its estimate $\widehat{\dot{\boldsymbol{\chi}}}_{\boldsymbol{u}}$ (since its real value may be unknown during the task execution), results in:

$$\boldsymbol{A} \dot{\boldsymbol{q}}_d = \dot{\boldsymbol{y}}_d^\circ + \boldsymbol{B} \widehat{\dot{\boldsymbol{\chi}}}_{\boldsymbol{u}}. \tag{30}$$

Solving for the control input $\dot{\boldsymbol{q}}_d$ yields:

$$\dot{\boldsymbol{q}}_d = \boldsymbol{A}_{\boldsymbol{W}}^{\#} \left(\dot{\boldsymbol{y}}_d^\circ + \boldsymbol{B} \widehat{\dot{\boldsymbol{\chi}}}_{\boldsymbol{u}} \right), \tag{31}$$

where $\overset{\#}{\boldsymbol{W}}$ denotes the weighted pseudoinverse [12, 33] with weighting matrix $\boldsymbol{W}$. The pseudoinverse allows us to handle over-, under- and fully constrained cases as explained in Section 4.3.

4.2 Closed loop behavior

Substituting control input (31) in system equation (28) and then in output equation (27), and solving for $\dot{\boldsymbol{y}}$, results in:

$$\dot{\boldsymbol{y}} = \boldsymbol{A} \boldsymbol{A}_{\boldsymbol{W}}^{\#} \dot{\boldsymbol{y}}_d^\circ + \left(\boldsymbol{A} \boldsymbol{A}_{\boldsymbol{W}}^{\#} - \boldsymbol{1} \right) \boldsymbol{B} \dot{\boldsymbol{\chi}}_{\boldsymbol{u}} + \boldsymbol{A} \boldsymbol{A}_{\boldsymbol{W}}^{\#} \boldsymbol{B} \left(\widehat{\dot{\boldsymbol{\chi}}}_{\boldsymbol{u}} - \dot{\boldsymbol{\chi}}_{\boldsymbol{u}} \right). \tag{32}$$

This equation reveals the individual contributions of the control law ($\dot{\boldsymbol{y}}_d^\circ$), the geometric disturbances ($\dot{\boldsymbol{\chi}}_{\boldsymbol{u}}$) and the estimation errors ($\widehat{\dot{\boldsymbol{\chi}}}_{\boldsymbol{u}} - \dot{\boldsymbol{\chi}}_{\boldsymbol{u}}$). In the absence of estimation errors and in the case of a fully or underconstrained system ($\boldsymbol{A} \boldsymbol{A}_{\boldsymbol{W}}^{\#} = \boldsymbol{1}$), the closed loop behavior reduces to (29). This result proves that the desired output will effectively be realized with closed loop dynamics governed by the applied control law. In the case of an overconstrained system, only the projection of the desired outputs according to projection matrix $\boldsymbol{A} \boldsymbol{A}_{\boldsymbol{W}}^{\#}$ can be realized. In addition, the effect of the uncertainty coordinates is instantaneously only compensated within the range of $\boldsymbol{A} \boldsymbol{A}_{\boldsymbol{W}}^{\#}$, which in general changes with time.

[4] These equations are similar to the ones used in multibody systems (MBS). However, the elimination of the auxiliary variables is not done in most MBS algorithms; instead, the closure is explicitly added as extra constraints.

4.3 Invariant constraint weighting

In general, the set of control constraints can be over– or underconstrained, or a combination of both. This paper proposes to use the minimum weighted norm solution ("pseudo-inverse approach"), both in joint space (for underconstrained systems) and in constraint space (for overconstrained systems). Many different norms can be used.

In joint space, the mass matrix of the robot is a possible and well-known weighting matrix. The corresponding norm is then proportional to the kinetic energy of the robot.

On the other hand, in constraint space a possible solution consists of choosing the weighting matrices in (31) as $\boldsymbol{W} = diag(w_i^2)$, with:

$$w_i = \alpha \frac{1}{\Delta_{pi} k_{pi}} \quad \text{or} \quad w_i = \alpha \frac{1}{\Delta_{vi}}. \tag{33}$$

In these expressions Δ_{pi} is a measure for the tolerance allowed on constraint i in terms of deviation from y_{di}, and Δ_{vi} is a measure for the tolerance allowed on constraint i in terms of deviation from $\dot{y}_{di}$. $k_{pi}[s^{-1}]$ is the feedback constant for constraint i, defined in (29). α is an activation coefficient which allows for a smooth transition between subtasks. Since each subtask has, in general, a different set of constraints, a transition phase is required in order to avoid abrupt transition dynamics. The suggested approach consists of two parts: (i) to *simultaneously* activate *both* sets of constraints during the transition phase, and (ii) to vary the activation coefficient α from 1 to 0 for the "old" subtask and from 0 to 1 for the "new" subtask. The smoothness of the time-variation of this coefficient can be adapted by the task programmer to the dynamic requirements of the task.

Using these weights makes the solution for the robot actuation, and hence also the resulting closed loop robot behavior, invariant with respect to the units that are used to express the constraints.

4.4 Model update and estimation

The goal of model update and estimation is threefold: (i) to provide an estimate for the system outputs $\boldsymbol{y}$ to be used in the feedback terms of constraint equations (29), (ii) to provide an estimate for the uncertainty coordinates $\boldsymbol{\chi_u}$ to be used in the control input (31), and (iii) to maintain the consistency between the joint and feature coordinates $\boldsymbol{q}$ and $\boldsymbol{\chi_f}$ based on the loop constraints[5]. Model update and estimation is based on an extended system model:

$$\frac{d}{dt}\begin{pmatrix} \boldsymbol{q} \\ \boldsymbol{\chi_f} \\ \boldsymbol{\chi_u} \\ \dot{\boldsymbol{\chi}}_{\boldsymbol{u}} \\ \ddot{\boldsymbol{\chi}}_{\boldsymbol{u}} \end{pmatrix} = \begin{pmatrix} 0\,0\,0 & 0 & 0 \\ 0\,0\,0 & -\boldsymbol{J_f}^{-1}\boldsymbol{J_u} & 0 \\ 0\,0\,0 & 1 & 0 \\ 0\,0\,0 & 0 & 1 \\ 0\,0\,0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{q} \\ \boldsymbol{\chi_f} \\ \boldsymbol{\chi_u} \\ \dot{\boldsymbol{\chi}}_{\boldsymbol{u}} \\ \ddot{\boldsymbol{\chi}}_{\boldsymbol{u}} \end{pmatrix} + \begin{pmatrix} 1 \\ -\boldsymbol{J_f}^{-1}\boldsymbol{J_q} \\ 0 \\ 0 \\ 0 \end{pmatrix} \dot{\boldsymbol{q}}_d. \tag{34}$$

This model consists of three parts: the first line corresponds to the system model (28), the second line corresponds to the velocity loop constraint (26), and the last three lines correspond to the dynamic model for the uncertainty coordinates. While in (34) a constant acceleration model is used for the uncertainty coordinates $\boldsymbol{\chi_u}$, other dynamic models governing the uncertainty coordinates could be used as well. If the joint coordinates are measured, which is usually the case, the first line in (34) is omitted.

A prediction-correction procedure is proposed here. The prediction consists of two steps. The first step generates predicted estimates based on (34). A second step eliminates any inconsistencies between these predicted state estimates: the dependent variables $\widetilde{\boldsymbol{\chi_f}}$ are made consistent with the other estimates ($\widetilde{\boldsymbol{q}}$, $\widetilde{\boldsymbol{\chi_u}}$) by iteratively solving the position loop constraints. Since in this step no extra information on the geometric uncertainties is available, and since there is no physical motion of the robot involved, both $\widetilde{\boldsymbol{\chi_u}}$ and $\widetilde{\boldsymbol{q}}$ are kept constant. Adapting $\widetilde{\boldsymbol{\chi_f}}$ is sufficient to close any opening in the position loops, since $\boldsymbol{\chi_f}$ can be solved unambiguously from these position loops. This step can also be used at the beginning of an application task. Usually, only initial values for $\boldsymbol{q}$ and $\widehat{\boldsymbol{\chi}}_{\boldsymbol{u}}$ are available. Hence this first step generates starting values for $\boldsymbol{\chi_f}$ that are consistent with $\boldsymbol{q}$ and $\widehat{\boldsymbol{\chi}}_{\boldsymbol{u}}$.

[5]For a nonholonomic system consistency with the operational robot coordinates $\boldsymbol{\chi_q}$ has to be maintained as well.

Similarly, the correction consists of two steps. The first step generates updated estimates based on the predicted estimates and on the information contained in the sensor measurements. In some cases the uncertainty coordinates can be measured directly. Since the uncertainty coordinates appear as states in the extended system model, the expression for the measurement equation then simplifies by generalizing (3) to:

$$\boldsymbol{h}\left(\boldsymbol{q}, \boldsymbol{\chi_f}, \boldsymbol{\chi_u}\right) = \boldsymbol{z}. \tag{35}$$

Since these measurement equations can be expressed in any of the dependent coordinates $\boldsymbol{\chi_u}$, $\boldsymbol{q}$ and $\boldsymbol{\chi_f}$, the position loop constraints have to be included in this step to provide the relationship with the other coordinates. Different estimation techniques can be used to obtain optimal estimates of the geometric uncertainties. Extended Kalman filtering is an obvious choice, but more advanced techniques are also applicable. Finally, since the correction procedure may re-introduce inconsistencies between the updated estimates, the second step of the prediction is repeated, but now applied to the updated state estimates.

After prediction and correction of the states, estimates for the system outputs $\widehat{\boldsymbol{y}}$ follow from (2).

Given the extended system model and the conceptual procedure discussed above, detailed numerical procedures can be obtained by applying state of the art estimation techniques [3, 13, 17, 20, 37].

5 Example applications

This section illustrates the paper's main contributions by several example applications. The focus is on *task specification* (Section 3), as the detailed procedures for control and estimation can be derived subsequently by straightforward application of Section 4. Hence, for the sake of brevity, numerical details such as control gains, noise levels, constraints weights, etc. are omitted in the presentation of the simulation and experimental results. In each example, the ease of the task specification is reflected by the fact that only single scalars are needed to specify the desired tasks, even for the most complex ones.

Some of the examples have completely or partially been solved before, however using dedicated approaches. Hence, we do not claim that we have solved all of these applications for the first time, but rather we illustrate how control solutions can be generated for each application using the systematic approach presented in this paper.

The section contains both simulations and experiments. The range of considered robot systems is very wide, involving mobile robots, multiple robot systems as well as human-robot co-manipulation. All simulations and experiments use velocity based control and Extended Kalman Filters for model update and estimation.

5.1 Laser tracing

This section shows application of the methodology to a geometrically complex task involving an underconstrained specification as well as estimation of uncertain geometric parameters. The goal is to trace simultaneously a path on a plane as well as on a cylindrical barrel with known radius using two lasers which are rigidly attached to the robot end effector, Figure 6. The lasers also measure the distance to the surface. The position and orientation of the plane and the position of the barrel are uncertain at the beginning of the task. Furthermore, this section shows how the lasers' positions and orientations with respect to the robot end effector can be calibrated.

Object and feature frames The use of two lasers beams suggests the use of two feature relationships, one for the laser-plane combination, feature a, and one for the laser-barrel combination, feature b. Figure 6 shows the object and feature frames. For the laser-plane feature:

- frame $o1^a$ fixed to the plane,
- frame $o2^a$ fixed to the first laser on the robot end effector and with its z-axis along the laser beam,

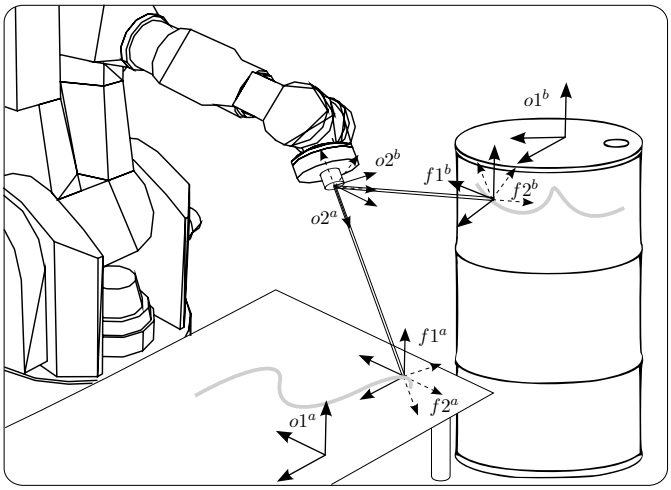


Figure 6: The object and feature frames for simultaneous laser tracing on a plane and a barrel.

- frame $f1^a$ with the same orientation as $o1^a$, and located at the intersection of the laser with the plane,
- frame $f2^a$ with the same position as $f1^a$ and the same orientation as $o2^a$,

and for the laser-barrel feature:

- frame $o1^b$ fixed to the barrel and with its x-axis along the axis of the barrel,
- frame $o2^b$ fixed to the second laser on the robot end effector and with its z-axis along the laser beam,
- frame $f1^b$ located at the intersection of the laser with the barrel, z-axis perpendicular to the barrel surface and x-axis parallel to the barrel axis,
- frame $f2^b$ with the same position as $f1^b$ and the same orientation as $o2^b$.

Feature coordinates For both features a minimal set of position coordinates exists representing the six degrees of freedom between the two objects. For the laser-plane feature:

$$\boldsymbol{\chi_{fI}}^a = \begin{pmatrix} x^a & y^a \end{pmatrix}^T, \tag{36}$$

$$\boldsymbol{\chi_{fII}}^a = \begin{pmatrix} \phi^a & \theta^a & \psi^a \end{pmatrix}^T, \tag{37}$$

$$\boldsymbol{\chi_{fIII}}^a = \begin{pmatrix} z^a \end{pmatrix}, \tag{38}$$

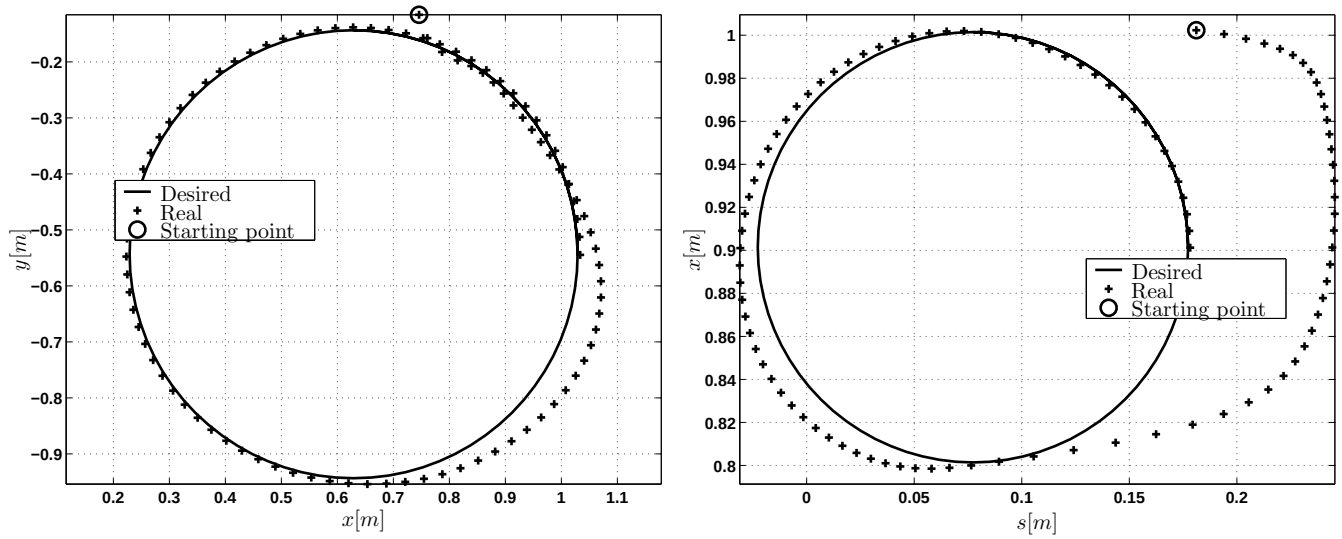
where x^a and y^a are expressed in $o1^a$ and represent the position of the laser dot on the plane, while z^a is expressed in $o2^a$ and represents the distance of the robot to the plane along the laser beam. ϕ^a, θ^a, ψ^a represent Euler angles between $f1^a$ and $f2^a$. For the laser-barrel feature:

$$\boldsymbol{\chi_{fI}}^b = \begin{pmatrix} x^b & \alpha^b \end{pmatrix}^T, \tag{39}$$

$$\boldsymbol{\chi_{fII}}^b = \begin{pmatrix} \phi^b & \theta^b & \psi^b \end{pmatrix}^T, \tag{40}$$

$$\boldsymbol{\chi_{fIII}}^b = \begin{pmatrix} z^b \end{pmatrix}, \tag{41}$$

where x^b and α^b are cylindrical coordinates expressed in $o1^b$ representing the position of the laser dot on the barrel, while z^b is expressed in $o2^b$ and represents the distance of the robot to the plane along the laser beam.



(a) Laser spot on plane.

(b) Laser spot on barrel surface, with $s = R\alpha^b$ the circumferential arc length and x the distance along the cylinder axis.

Figure 7: Laser tracing on plane (a) and on barrel (b) (simulation).

Uncertainty coordinates The unknown position and orientation of the (infinite) plane is modeled by:

$$\boldsymbol{\chi_{uI}}^a = \begin{pmatrix} h^a & \alpha^a & \beta^a \end{pmatrix}^T, \tag{42}$$

with h^a the z-position of a reference point on the plane with respect to the world, and α^a and β^a the Y and X Euler angles which determine the orientation of the plane with respect to the world. The unknown position of the barrel is modeled by:

$$\boldsymbol{\chi_{uI}}^b = \begin{pmatrix} x_u^b & y_u^b \end{pmatrix}^T, \tag{43}$$

with x_u^b and y_u^b the x- and y-position of the barrel with respect to the world. If the laser position and orientation with respect to the end effector is also unknown, for example during the calibration phase, additional uncertainty coordinates are introduced:

$$\boldsymbol{\chi_{uIV}} = \begin{pmatrix} x_l & y_l & z_l & \phi_l & \theta_l \end{pmatrix}^T. \tag{44}$$

Task specification To generate the desired path on the plane, constraints have to be specified on the system outputs:

$$y_1 = x^a \quad \text{and} \quad y_2 = y^a, \tag{45}$$

while for the path on the barrel constraints have to be specified on:

$$y_3 = x^b \quad \text{and} \quad y_4 = \alpha^b. \tag{46}$$

In this example circles are specified for both paths, yielding $y_{id}(t)$, for $i = 1, \ldots, 4$. The measurement equations are easily expressed using the coordinates of features a and b:

$$z_1 = z^a \quad \text{and} \quad z_2 = z^b. \tag{47}$$

Results Simulations are shown in Figure 7. The initial estimation errors for the plane are $0.40m$ for the z-position of the reference point on the plane, 20° for Euler angle α^a and 30° for Euler angle β^a. The initial estimation errors for the barrel are $0.4m$ in the x-direction and $0.1m$ in the y- direction. An extended Kalman filter is used for the estimation. As soon as the locations of the plane and the barrel are estimated correctly, after approximately one circle ($8s$), the circles traced by the laser beams equal the desired ones.

5.2 Mobile robot

This section shows application of the methodology to the localization and path tracking of a nonholonomic system. The goal is to move a mobile robot with two differentially driven wheels on a desired path, while the robot's position in the world is uncertain due to friction, slip of the wheels, or other disturbances. The mobile robot is equipped with two sensors: an ultrasonic sensor measuring a distance to a wall, and a range finder measuring the distance and angle with respect to a beacon. Both the positions of beacon and wall are known.

Object and feature frames Two objects are relevant for the task description: $o1$ fixed to the wall, and $o2$ attached to the mobile robot. The use of the two sensors suggests the use of two feature relationships: feature a for the ultrasonic sensor, and feature b for the range finder. A third feature, feature c, is required to specify the desired path of the robot. Figure 8 shows the frames, assigned to these objects and features:

- frame $o1$, fixed to the wall, with its x-axis along the wall,
- frame $o2$, fixed to the mobile robot,
- frame $f1^a$, with the same orientation as $o1$ and only able to move in the x direction of $o1$,
- frame $f2^a$, fixed to frame $o2$,
- frame $f1^b$, representing the beacon location and fixed to frame $o1$,
- frame $f2^b$, of which the x-axis represents the beam of the range finder hitting the beacon,
- frame $f1^c$, coinciding with $o1$, and
- frame $f2^c$, coinciding with $o2$.

Feature coordinates For each of the three features a minimal set of position coordinates exists representing the three degrees of freedom of the planar motion between the objects $o1$ and $o2$:

- for feature a (ultrasonic sensor):

$$\boldsymbol{\chi_{fI}}^a = \begin{pmatrix} x^a \end{pmatrix}, \tag{48}$$

$$\boldsymbol{\chi_{fII}}^a = \begin{pmatrix} y^a & \theta^a \end{pmatrix}^T, \tag{49}$$

$$\boldsymbol{\chi_{fIII}}^a = \begin{pmatrix} - \end{pmatrix}, \tag{50}$$

- for feature b (range finder):

$$\boldsymbol{\chi_{fI}}^b = \begin{pmatrix} - \end{pmatrix}, \tag{51}$$

$$\boldsymbol{\chi_{fII}}^b = \begin{pmatrix} x^b & \theta^b \end{pmatrix}^T, \tag{52}$$

$$\boldsymbol{\chi_{fIII}}^b = \begin{pmatrix} \phi^b \end{pmatrix}, \tag{53}$$

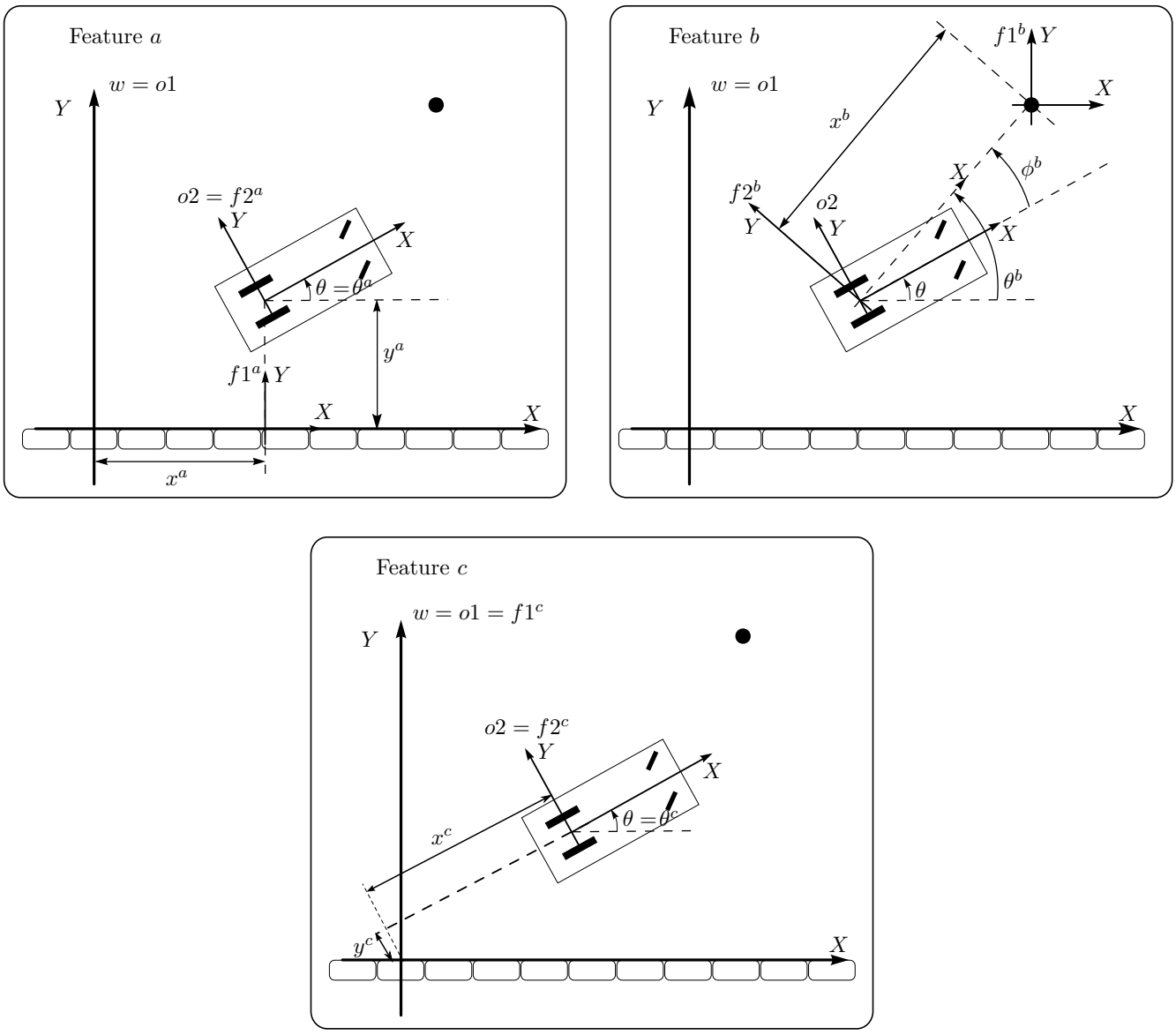


Figure 8: The feature coordinates and object and feature frames for the mobile robot example: left for feature a, ultrasonic sensor; right for feature b, range finder; bottom for feature c, robot trajectory.

- for feature c (path tracking):

$$\boldsymbol{\chi_{fI}}^c = \left(- \right), \tag{54}$$

$$\boldsymbol{\chi_{fII}}^c = \left(\begin{array}{ccc} x^c & y^c & \theta^c \end{array} \right)^T, \tag{55}$$

$$\boldsymbol{\chi_{fIII}}^c = \left(- \right). \tag{56}$$

These feature coordinates are defined in Figure 8. Obviously, (x^a, y^a, θ^a) can also be used to model path tracking feature c. At first sight this choice would even be more natural. The reason for defining the path tracking coordinates (x^c, y^c, θ^c) with respect to $o2 = f2^c$ is that this choice results in a path tracking controller with dynamics which are defined in the robot frame $o2$. The relation between the two sets of coordinates is derived using inversion of a homogeneous transformation matrix:

$$\boldsymbol{T}_{o2}^{o1}(x^c, y^c, \theta^c) = \left(\boldsymbol{T}_{o1}^{o2}(x^a, y^a, \theta^a) \right)^{-1} \tag{57}$$

Operational space robot coordinates In case of a nonholonomic robot, the position loop constraints (19), and more particularly the relative pose between $o2$ and w, cannot be written in terms of $\boldsymbol{q}$. Instead, operational space robot coordinates $\boldsymbol{\chi_q}$ are defined. In this case, a natural choice is $\boldsymbol{\chi_q} = \boldsymbol{\chi_f}^c$, because the dependency relation between $\dot{\boldsymbol{\chi}}_{\boldsymbol{q}}$ and $\dot{\boldsymbol{q}}$ is very simple:

$$\dot{\boldsymbol{\chi}}_{\boldsymbol{q}} = \begin{pmatrix} \dot{x}^c \\ \dot{y}^c \\ \dot{\theta}^c \end{pmatrix} \tag{58}$$

$$= \begin{pmatrix} \frac{R}{2} & \frac{R}{2} \\ 0 & 0 \\ -\frac{R}{2B} & \frac{R}{2B} \end{pmatrix} \begin{pmatrix} \dot{q}_l \\ \dot{q}_r \end{pmatrix} \tag{59}$$

$$= \boldsymbol{J_r}\dot{\boldsymbol{q}}, \tag{60}$$

where $\dot{q}_l$ and $\dot{q}_r$ denote the velocity of the left and right wheel, respectively, while R denotes the radius of both wheels and B denotes the wheel base. $\boldsymbol{J_r}$ represents the robot jacobian and is expressed in $o2$.

Replacing $\boldsymbol{q}$ in (2) and (6) by $\boldsymbol{\chi_q}$, and following the analysis in Section 4.1 results in:

$$\boldsymbol{C_q} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\chi_q}} \boldsymbol{J_r}; \quad \boldsymbol{J_q} = \frac{\partial \boldsymbol{l}}{\partial \boldsymbol{\chi_q}} \boldsymbol{J_r}, \tag{61}$$

after substituting (58).

Uncertainty coordinates Equation (58) represents the nonholonomic constraint which may be disturbed by wheel slip:

$$\dot{\boldsymbol{\chi}}_{\boldsymbol{q}} = \boldsymbol{J_r} \left(\dot{\boldsymbol{q}} + \dot{\boldsymbol{q}}_{slip} \right), \tag{62}$$

where $\dot{\boldsymbol{q}}_{slip} = s\dot{\boldsymbol{q}}$, with s the estimated slip rate. Hence, $\boldsymbol{\chi_{uIV}} = \boldsymbol{q}_{slip}$, while it is obvious from (25) that $\boldsymbol{J_u} = \boldsymbol{J_q}$.

Task specification Since the control is specified in the operational space of the robot, the system outputs are:

$$y_1 = x^c, \quad y_2 = y^c, \quad y_3 = \theta^c. \tag{63}$$

If the desired path is given in terms of x^a, y^a and θ^a, the desired values $y_{1d}(t)$, $y_{2d}(t)$ and $y_{3d}(t)$ are obtained using (57). The measurement equations are very simple due to the chosen set of feature coordinates:

$$z_1 = y^a, \quad z_2 = x^b, \quad z_3 = \theta^b, \tag{64}$$

where z_1 represents the ultrasonic measurement, while z_2 and z_3 represent the range finder measurements.

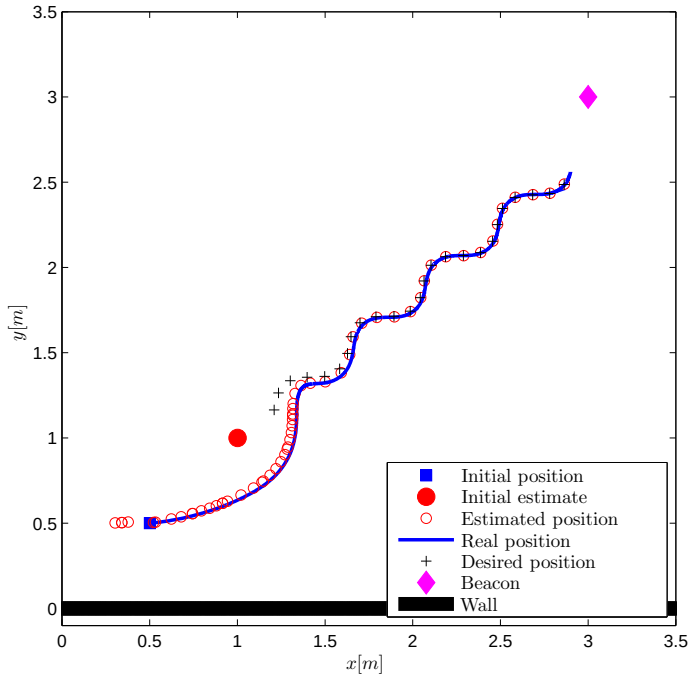


Figure 9: Localization and path tracking control of a mobile robot.

Feedback control The path controller is implemented in operation space, by applying constraints (29) with

$$\boldsymbol{K_p} = \begin{pmatrix} k_p & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \frac{{k_p}^2}{2sign(\dot{x}_c)} & k_p \end{pmatrix}, \tag{65}$$

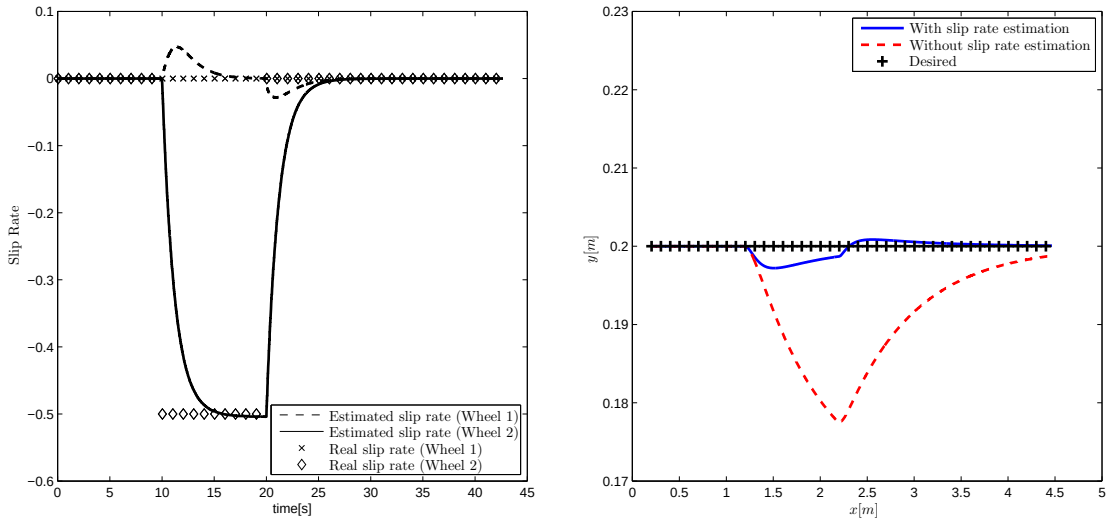
and k_p a feedback constant. Note that in this case $\boldsymbol{K_p}$ is not diagonal: in order to eliminate a lateral error, it has to be transformed into an orientation error.

Results In a first simulation both sensor measurements are used while the robot is controlled to follow the desired path. The desired path starts at the estimated initial position of the robot and consists of a constant translational velocity and a sinusoidal rotational velocity. The estimated, real and desired position of the robot are shown in Figure 9.

In a second simulation one of the wheels slips with a slip rate of -0.5 for 10 seconds, while the desired path of the robot consists of a constant translational velocity and no rotational velocity. Figure 10(a) shows the estimated slip rate. A constant slip rate model is used in the estimation. Figure 10(b) shows the benefit of estimating the slip. When no slip is estimated the controller cannot reduce the positioning error as long as slip occurs; the positioning error is even increasing. If the slip is estimated and $\widehat{\boldsymbol{\chi}}_{\boldsymbol{su}} = \hat{s}\dot{\boldsymbol{q}}$ is used in (31), the trajectory error can be reduced during slip.

5.3 Contour tracking

This section shows application of the methodology to a two dimensional task involving estimation of time-varying uncertain geometric parameters. The task consists of tracking an unknown contour with a tool mounted on the robot. The contact force (magnitude and orientation) between the contour and the robot is



(a) Estimation of the slip rate on both wheels.

(b) Trajectory of mobile robot.

Figure 10: Localization and path tracking control of a mobile robot: effect of estimation and feedforward of wheel slip on the trajectory.

measured. Furthermore, a desired contact force and orientation between the robot end effector and the contour is specified. Two dynamic models of the uncertain geometric parameters which represent the unknown contour are compared: a 'constant tangent' model and a 'constant curvature' model.

Object and feature frames One feature relationship is sufficient for the specification of the contour tracking task. Figure 11 shows the frames, assigned to the object and features:

- frame $o1$ is fixed to the workpiece to which the contour belongs,
- frame $o2$ is fixed to the robot end effector,
- frame $f1$ is located on the real contour, and its y-axis is perpendicular to it,
- frame $f1'$ (defined in Figure 5) is located on the estimated contour, and its y-axis is perpendicular to it,
- frame $f2$ is fixed to the robot end effector, and has the same orientation as $o2$.

Feature coordinates In the case of a *known* contour, $f1 = f1'$, and a minimal set of position coordinates exists representing the three degrees of freedom between $o1$ and $o2$:

$$\boldsymbol{\chi}_{f\mathrm{I}} = \begin{pmatrix} s \end{pmatrix}, \tag{66}$$

$$\boldsymbol{\chi}_{f\mathrm{II}} = \begin{pmatrix} y & \theta \end{pmatrix}^T, \tag{67}$$

$$\boldsymbol{\chi}_{f\mathrm{III}} = \begin{pmatrix} - \end{pmatrix}, \tag{68}$$

where s is the arc length along the known contour, y is expressed in $f1$ and represents the distance of the robot end effector to the contour, and θ is expressed in $f1$ with reference point in the origin of $f2$ and represents the orientation of the robot end effector with respect to the contour. Using the arc length parameter s and the contour model, the pose of frame $f1$ with respect to $o1$ is determined by:

$$\begin{cases} x_c = f_x(s) \\ y_c = f_y(s) \\ \theta_c = f_\theta(s) \end{cases}, \tag{69}$$

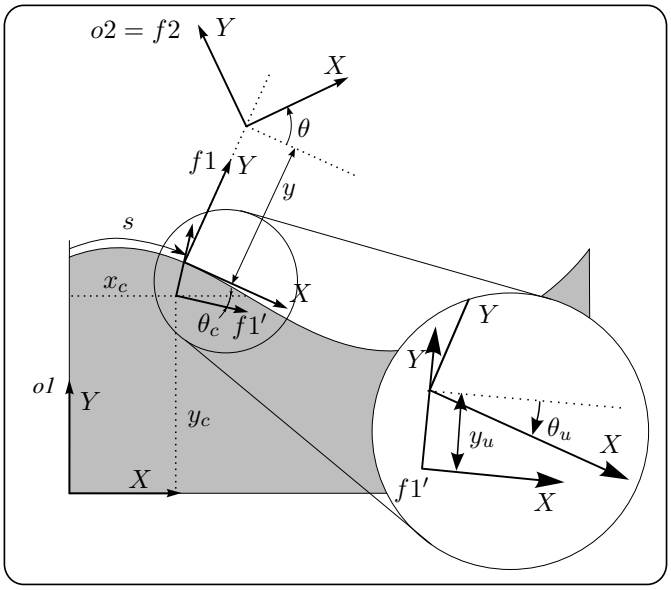


Figure 11: The feature coordinates and object and feature frames for the contour tracking example.

where x_c and y_c represent the position of $f1$ on the contour with respect to $o1$, while θ_c represents its orientation.

In the case of an *unknown* contour however, $f1 \neq f1'$, and no set of minimal position coordinates exists. $\boldsymbol{\chi_{fI}}$ has to be replaced by

$$\boldsymbol{\chi_{fI}}_{nm} = \begin{pmatrix} x_c & y_c & \theta_c \end{pmatrix}^T, \tag{70}$$

where subscript nm indicates the nonminimal nature, and $\boldsymbol{\chi_{fI}}_{nm}$ now represents the submotion between $o1$ and $f1'$. $\boldsymbol{\chi_{fII}}$ and $\boldsymbol{\chi_{fIII}}$ remain the same as in the case of the known contour. However, as discussed in Section 3.2, a minimal set of coordinates always exists at velocity level. In this case, the first submotion can still be modelled by a single velocity coordinate $\dot{s}$, which expresses that $f1'$ can move only tangential to the contour with a velocity $\dot{s}$. The relationship between the nonminimal set of position coordinates and the minimal set of velocity coordinates is easily expressed as:

$$\dot{\boldsymbol{\chi}}_{\boldsymbol{f}nm} = \begin{pmatrix} \dot{x}_c \\ \dot{y}_c \\ \dot{\theta}_c \\ \dot{y} \\ \dot{\theta} \end{pmatrix} \tag{71}$$

$$= \underbrace{\begin{pmatrix} \cos(\theta_c) & 0 & 0 \\ \sin(\theta_c) & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{\boldsymbol{J_{nm}}} \begin{pmatrix} \dot{s} \\ \dot{y} \\ \dot{\theta} \end{pmatrix}. \tag{72}$$

Starting from these nonminimal coordinates the feature jacobian used in the analysis in Section 4 becomes:

$$\boldsymbol{J_f} = \frac{\partial \boldsymbol{l}}{\partial \boldsymbol{\chi_f}_{nm}} \boldsymbol{J_{nm}}. \tag{73}$$

Notice the parallel between this example and the mobile robot example (Section 5.2) where a set of nonminimal coordinates is introduced on the level of the robot coordinates $\boldsymbol{\chi_q}$. Notice also that in (71) $\dot{\theta}_c$ is taken equal to 0 since, for an unknown contour, nothing can be said about the change of the tangent of the model contour.

Uncertainty coordinates Since the real contour is not known, the model contour frame $f1'$ may deviate from the real contour frame $f1$. Therefore, uncertainty coordinates are introduced:

$$\boldsymbol{\chi_{uII}} = \begin{pmatrix} y_u & \theta_u \end{pmatrix}^T, \tag{74}$$

with y_u the distance between the modelled and the real contour expressed in $f1'$, and θ_u the orientation between these two, and also expressed in $f1'$.

Task specification To follow the contour with a desired contact force and with a desired orientation, constraints have to be specified on the system outputs:

$$y_1 = \dot{s}, \quad y_2 = F_y = -k_y y \quad \text{and} \quad y_3 = \theta, \tag{75}$$

where F_y represents the contact force between robot and contour, and k_y is the modeled contact stiffness.

The desired values for these outputs are specified as:

$$y_{1d} = \dot{s_d}, \quad y_{2d} = F_{yd} \quad \text{and} \quad y_{3d} = \theta_d. \tag{76}$$

The measurement equations for the magnitude and the orientation of the measured contact forces are easily expressed using the feature coordinates:

$$z_1 = -k_y y \quad \text{and} \quad z_2 = \theta. \tag{77}$$

Results Simulations have been carried out using different dynamic models for the uncertainty coordinates. First, a 'constant tangent' model is used:

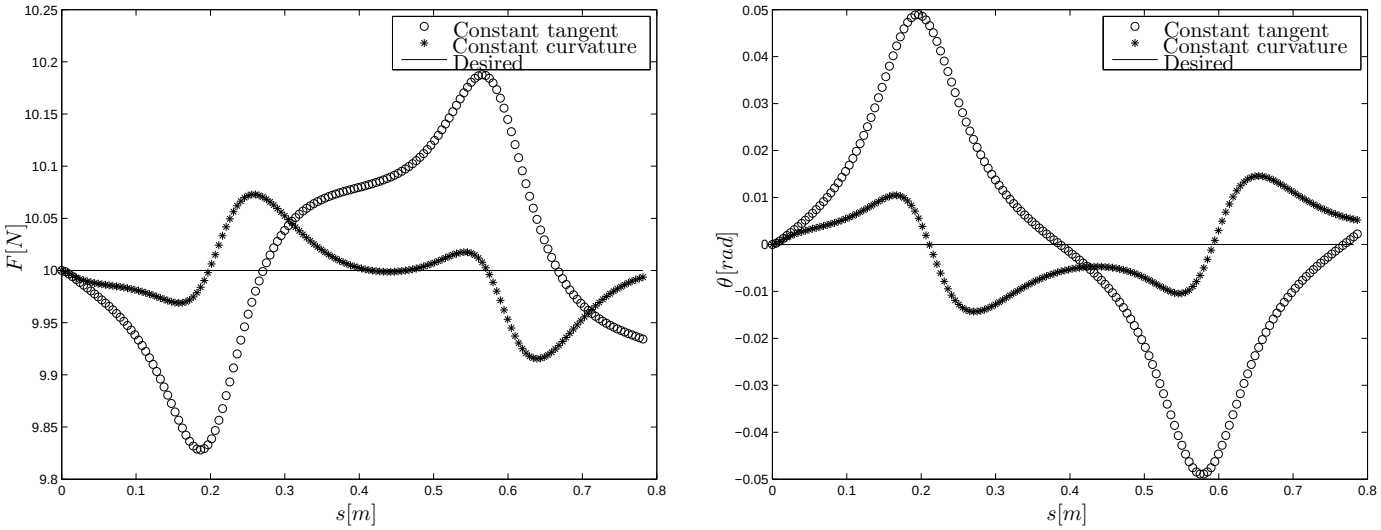
$$\frac{d}{dt}\begin{bmatrix} y_u \\ \theta_u \end{bmatrix} = \boldsymbol{0}_{2\times 1}. \tag{78}$$

Second, a 'constant curvature' model is used. Assuming a constant tangential velocity, this model contains $\dot{\theta}_u$ as an extra state variable and assumes it is constant[6]:

$$\frac{d}{dt}\begin{bmatrix} y_u \\ \theta_u \\ \dot{\theta}_u \end{bmatrix} = \begin{bmatrix} 0 \\ \dot{\theta}_u \\ 0 \end{bmatrix}. \tag{79}$$

In the simulation, the contour consists of a sine $y = A\sin(\omega x)$ with amplitude $A = 0.10m$ and $\omega = 100\frac{1}{m}$. The contour is tracked with a constant velocity of $0.0197\frac{m}{s}$. The desired and actual contact force and orientation between the robot end effector and the contour are shown in Figure 12. The more complex dynamic model with constant curvature results in a better contour tracking, which is most apparent in the orientation.

[6] An alternative consists of estimating the curvature κ, assuming it is is constant. In that case, the model contains the equations $\dot{\theta}_u = \kappa\dot{s}$ and $\frac{d\kappa}{dt} = 0$.



(a) Contact force between robot end effector and contour.

(b) Orientation between robot end effector and contour.

Figure 12: Contact force and orientation between robot end effector and contour for 'constant tangent' and 'constant curvature' model (simulation).

5.4 Human-robot co-manipulation

This section shows application of the methodology to an overconstrained task involving human-robot co-manipulation (Figure 13). A robot assists an operator to carry and fine position an object on a support beneath the object. A force/torque sensor is attached to the robot wrist. The force/torque sensor allows the operator to interact with the robot by exerting forces on the manipulated object. Using these interaction forces, the operator aligns one side of the object with its desired location on the support. A camera provides information of the position of the other side of the object relative to its desired position. These measurements are used by the robot to position this other side of the object. Hence, task control is shared between the human and the robot. The robot carries the weight of the object, generates a downward motion to realize a contact force between the object and its support, and at the same time positions one side of the object based on the camera measurements. At the same time the human aligns the other side of the object while maintaining overall control of the task.

Object and feature frames The use of two sensors suggests two feature relationships: feature a for the visual servoing and feature b for the force control. For feature a the relevant objects are the robot environment, $o1^a$ and the manipulated object, $o2$. For feature b the relevant objects are again the manipulated object $o2$, and an external object which is the source of the force interaction. This object could be either the human, or the support beneath the manipulated object. In fact, the force/torque sensor cannot distinguish between forces applied by the human and forces resulting from contact between the manipulated object and the support.

Figure 13 shows the object and feature frames, while Figure 14 shows the feature relationships:

- Frame $o1^a$ is fixed to the robot environment. The camera frame is a possible choice for this frame, as the camera is fixed in the environment.
- Frame $o2$ is chosen at the center of the object.
- $o1b$ is fixed to $o2$ by a compliance; $o1^b$ coincides with $o2$ when no forces are applied to the object. However, when forces are applied to the object, $o1^b$ and $o2$ may deviate from each other due to the passive compliance in the system.

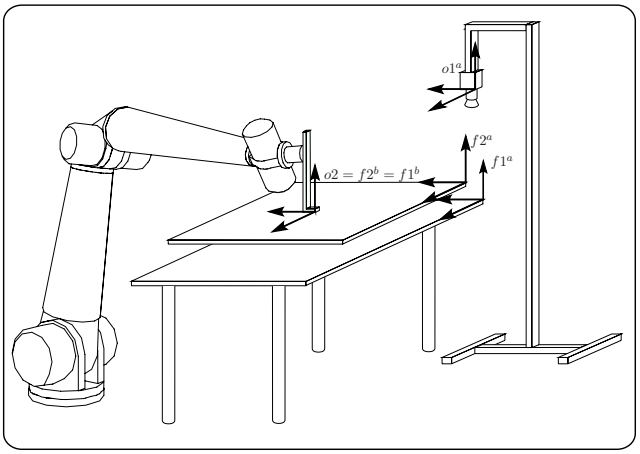


Figure 13: The object and feature frames for a human-robot co-manipulation task.

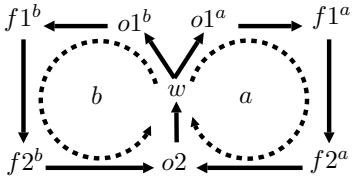


Figure 14: Object and feature frames for human-robot co-manipulation.

- Frame $f1^a$ is chosen at the reference pose on the support, with which the object should be aligned using the camera measurements.
- Frame $f2^a$ is fixed to the object. While the pose of $f2^a$ with respect to $o2$ is known, the relative pose between $f2^a$ and $f1^a$ is obtained from the camera measurements.
- If no forces are applied to the object, frames $f1^b$ and $f2^b$ coincide with $o2$, which is chosen as the reference frame for force control . However, frame $f2^b$ is rigidly attached to $o2$, while frame $f1^b$ is rigidly attached to $o1^b$. Hence, when forces are applied to the object, $f1^b$ and $f2^b$ deviate from each other. The corresponding passive stiffness is modelled as $diag(k_x, k_y, k_z, k_{\alpha x}, k_{\alpha y}, k_{\alpha z})$.

Feature coordinates As $f1^a$ is rigidly attached to $o1^a$, and $f2^a$ is rigidly attached to $o2$, all six degrees of freedom for feature a are located between $f1^a$ and $f2^a$. Similarly, as $f1^b$ is rigidly attached to $o1^b$, and $f2^b$ is rigidly attached to $o2$, all six degrees of freedom for feature b are located between $f1^b$ and $f2^b$:

- for feature a:

$$\boldsymbol{\chi_{fI}}^a = \left(\; - \; \right), \tag{80}$$

$$\boldsymbol{\chi_{fII}}^a = \left(\; x^a \quad y^a \quad z^a \quad \phi^a \quad \theta^a \quad \psi^a \; \right)^T, \tag{81}$$

$$\boldsymbol{\chi_{fIII}}^a = \left(\; - \; \right), \tag{82}$$

where ϕ^a, θ^a and ψ^a represent Euler angles,

- for feature b:

$$\boldsymbol{\chi_{fI}}^b = \left(- \right), \tag{83}$$

$$\boldsymbol{\chi_{fII}}^b = \left(\begin{array}{cccccc} x^b & y^b & z^b & \phi^b & \theta^b & \psi^b \end{array} \right)^T, \tag{84}$$

$$\boldsymbol{\chi_{fIII}}^b = \left(- \right), \tag{85}$$

where ϕ^b, θ^b and ψ^b represent small deformation angles about the frame axes.

Task specification The x and y positions of $f2^a$ with respect to $f1^a$ are controlled from the camera measurements[7]. Hence, two system outputs are:

$$y_1 = x^a, \quad y_2 = y^a. \tag{86}$$

Furthermore, when the robot establishes a contact between the object and its support, we want to control the contact force and two contact torques. Hence, three more outputs are:

$$\begin{array}{c} y_3 = F_z = k_z x^b, \quad y_4 = T_x = k_{\alpha x}\phi^b, \quad \text{and} \\ y_5 = T_y = k_{\alpha y}\theta^b, \end{array} \tag{87}$$

where F_i and T_i represent forces and torques expressed in $f2^b$. The operator can interact with the robot to align the object in six degrees of freedom, yielding another three system outputs in addition to y_3, y_4 and y_5:

$$\begin{array}{c} y_6 = F_x = k_x x^b, \quad y_7 = F_y = k_y y^b, \quad \text{and} \\ y_8 = T_z = k_{\alpha z}\psi^b. \end{array} \tag{88}$$

In the constraint equations the desired values for these outputs are specified as:

$$\begin{array}{c} y_{1d} = 0\text{mm}, \quad y_{2d} = 60\text{mm}, \\ y_{3d} = F_{zd}, \quad y_{4d} = 0, \quad y_{5d} = 0, \\ y_{6d} = y_{7d} = y_{8d} = 0. \end{array} \tag{89}$$

The first two constraints express the desired alignment of the object at the camera side, while the last six constraints express the desired force interaction. When the human applies forces, the robot will move so as to regulate the forces to the desired values. In addition, in contact with its support, the object will apply a contact force F_{zd} in vertical direction while aligning itself with the surface of the support. Obviously, constraints 1 and 2 conflict with constraints 6 and 7, since they all control translational motion in the horizontal plane. Hence, constraint weights have to be specified.

In this example all the outputs can be measured. Hence

$$z_i = y_i \quad \text{for } i = 1, \ldots, 8. \tag{90}$$

Results The experiment has been carried out on a Kuka 361 industrial robot equipped with a wrist mounted JR3 force/torque sensor. Figure 15 shows the setup. The manipulated object consists of a rectangular plate, which is to be placed on a table. Colored sheets are attached to the plate and the table as markers, to facilitate recognition of the object and the support in the camera images. Figure 16 shows some results. The left plot shows the F_x- and F_y-forces, exerted by the operator. The right plot shows x^a and y^a, as measured by the camera. As the task is overconstrained, the forces exerted by the operator conflict with the aligning motion from the visual servoing, yielding a spring-like behavior. The relative weights of the force and visual servoing constraints determine the spring constant of this behavior. A video of the experiment is found in Extension 1.

[7]Using the camera measurements it is also possible to control the orientation of the object, however this was not done in the experiment.

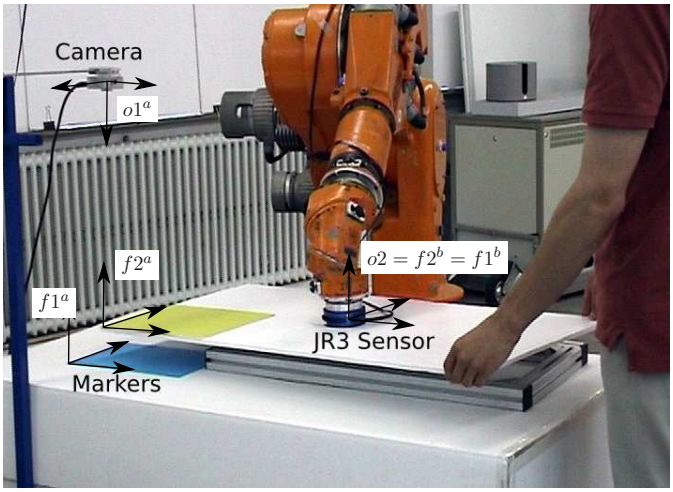


Figure 15: The experimental setup for the human-robot co-manipulation task.

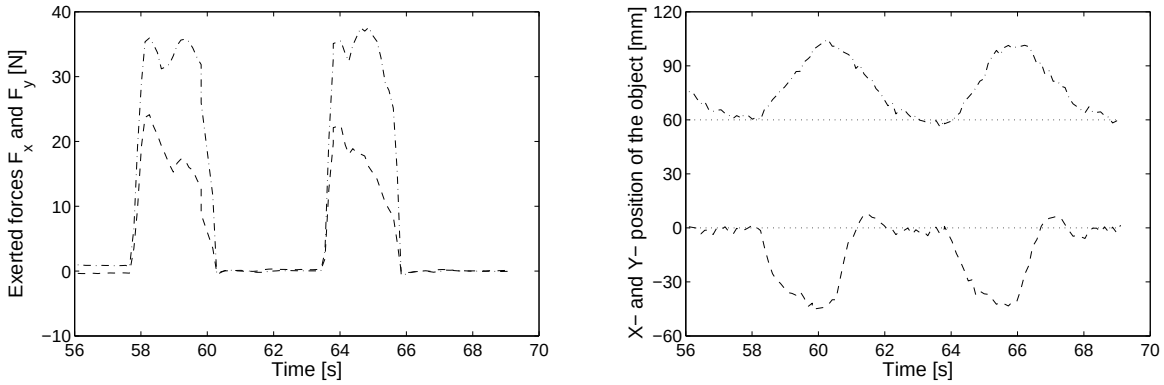


Figure 16: The left plot shows the forces F_x and F_y, exerted by the operator during the co-manipulation task. The right plot shows the alignment errors x^a and y^a as measured by the camera.

6 Discussion

6.1 Feature and joint coordinates

System outputs, natural constraints and measurements (2)-(5) are expressed both in terms of feature coordinates and joint coordinates. Both sets of coordinates are carried along throughout the analysis in Section 4. As shown, feature coordinates $\boldsymbol{\chi_f}$ facilitate the task specification in operational space. On the other hand, joint coordinates $\boldsymbol{q}$ allow us to include joint constraints and joint measurements. Typical examples of joint constraints are motion specification in joint space, or blocking of a defective joint.

6.2 Invariance

[12,14,28] have pointed out the importance of invariant descriptions when dealing with twists and wrenches to represent 3D velocity of a rigid body and 3D force interaction, respectively. Non-invariant descriptions result in a different robot behavior when changing the reference frame in which twists and wrenches are expressed, when changing the reference point for expressing the translational velocity or the moment, or when changing units. Such non-invariance is highly undesirable in practical applications as it compromises the predictability of the resulting robot behavior.

In this paper special care has been taken to obtain invariant solutions. In particular, nowhere pseudo-inverses appear, except in equation (31). These pseudo-inverses represent minimum weighted norm solutions while the weighting matrices have a clear physical meaning, as discussed in Section 4.3.

Due to this approach the robot behavior is completely defined by following user inputs, regardless of the particular numerical representation or implementation: (i) definition of outputs $\boldsymbol{y}$ and measurements $\boldsymbol{z}$; (ii) specification of natural constraints $\boldsymbol{g}$ and artificial constraints $\boldsymbol{y}_d$; (iii) tolerances Δ_{pi} and Δ_{vi}; (iv) feedback control gains $\boldsymbol{K}_p$; (v) parameters used for model update and estimation, for example process and measurement noise in case of Kalman filtering. While each of these inputs has a clear physical meaning, no ad-hoc parameters are introduced throughout the entire approach.

6.3 Task specification support

Even though the approach developed in this paper is very systematic, complete application down to the control and estimation level is quite involved in case of complex tasks. In addition, the task specification approach does not impose a unique definition of objects and feature frames, feature relationships, etc., but leaves room for task specific optimization. For both reasons, a software environment is needed to support the user. Evidently, the calculations for control and estimation can easily be automated once the task has completely been specified; the similarity with simulation of multibody systems is quite obvious and inspiring here. Hence, the challenge is to design a software environment to support the task specification as presented in Section 3 and illustrated in Section 5. This is subject of future work. Again, inspiration comes from the similarity with multibody systems. The goal is to specify the virtual kinematic chains, as in Figures 3 and 5, indicate which of the joints in these chains represent robot joints, feature coordinates or uncertainty coordinates, and define the artificial constraints ("drivers") as well as the measurements. Typical cases of (partial) task descriptions could be stored as templates in a library to simplify and speed up the interaction with the user. Clearly, these *topological* data have to be supplemented with *geometrical* as well as *dynamic* data. In order to relieve the user from this burden, a model of the robot system and its environment must be available in order to extract these data automatically.

6.4 Other applications

This section briefly discusses additional applications that can be tackled using the approach presented in this paper.

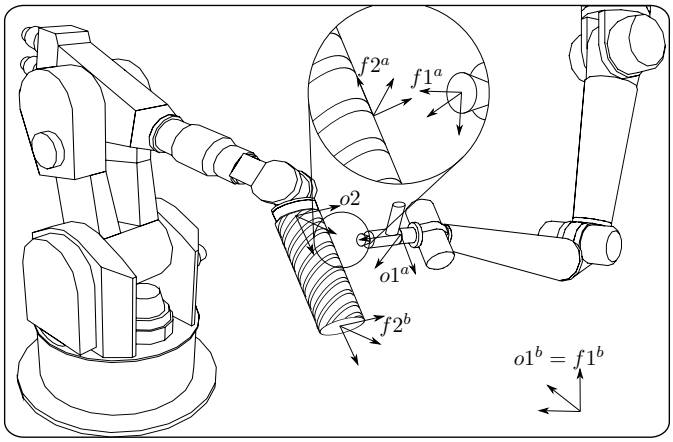


Figure 17: Two robots performing simultaneous pick-and-place and painting operations on a single work piece.

Multiple robots with simultaneous tasks (Extension 2) An example task consists of a pick-and-place operation of a compressor screw combined with a "painting" job using two cooperating robots, Figure 17. One robot holds the compressor screw while the other robot sprays paint. For each subtask feature relationships are defined.
Multimedia Extension 2 contains a movie of an experiment performed by a Kuka 361 and a Kuka160 industrial robot, with three subtasks: (i) picking the screw, (ii) the gross motion combined with the painting operation, and (iii) placing the screw. During the picking and placing of the screw constraints are applied to the pose of the screw. Hence, the motion of the robot holding the screw is fully specified. The second robot is not moving during these subtasks. In the second subtask, constraints are applied to the relative motion between the painting gun and the compressor screw as well as to the cartesian translation of the screw during the gross motion. We assume the paint cone is symmetric. Hence, the angle around the painting axis is arbitrary and does not need to be specified. As a result, eight constraints are specified, five for the relative motion and three for the gross translational motion. Since the robot system has twelve degrees of freedom the task is underconstrained.

Multi-link multi-contact force control This task is discussed in [34]. A serial robot arm has to perform simultaneous and independent control of two contact forces with the environment. One contact point is located at the robot end effector, while the other contact is located at one of its links. Applying the approach presented in this paper, both the end effector and the considered robot link are modelled as objects. Each object has a contact feature with corresponding loop constraint. Obviously, the loop constraints at velocity level, contain different robot jacobians $\boldsymbol{J_q}$, since the motion of the link is not affected by the last robot joint(s). Manipulation of objects by multi-finger grippers or whole arm manipulation can be modelled in the same way.

Systems with local flexibility Consider the spindle-assembly task from [5].

This task requires the insertion of a compressible spindle between two fixed supports. In the contact configuration shown in Figure 18, there are two point contacts. Each contact is modelled by a feature with corresponding loop constraint. This problem is reduced to the multi-link, multi-contact application discussed above by considering the compression coordinate δ as an extra joint of the robot system. In this case the extra joint is passive, as it is actuated by a spring. In the velocity based approach, the control input is still given by (31), but now $\dot{\boldsymbol{q}}$ includes the extra passive joint. While the desired joint velocities are applied to the real robot joints, the desired velocity of the compressible joint follows automatically from the passive spring action.

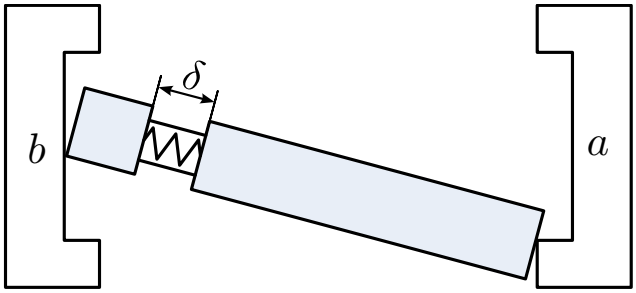


Figure 18: Spindle assembly task

7 Conclusion

This paper introduces a systematic constraint-based approach to specify the instantaneous motion specification in complex tasks for general sensor-based robot systems consisting of rigid links and joints. Complex tasks involve multiple task constraints and/or multiple sensors in the presence of uncertain geometric parameters. The approach integrates both task specification and estimation of uncertain geometric parameters in a unified framework. Major contributions are the use of feature coordinates, defined with respect to object and feature frames, which facilitate the task specification, and the introduction of uncertainty coordinates to model geometric uncertainty. To provide the link with real-time task execution a velocity based control scheme has been worked out that accepts the task specification as its input. This control scheme includes compensation for the effect of, possibly time varying, uncertain geometric parameters which are observed by means of a state estimator. Dimensionless constraint weighting results in an invariant robot behavior in case of conflicting constraints with heterogeneous units.

By following the approach developed in this paper the task programmer or application designer is able to obtain a solution for task specification and real-time control of complex tasks. This is a major advantage compared to existing constraint-based approaches. Existing approaches do not take into account uncertain geometric parameters in a systematic way, are limited to simple geometric configurations, for example configurations that can be modelled using a single task frame, or do not provide adequate tools to facilitate the expression of the task constraints in applications involving more complex geometry.

The equations governing the control law and estimators follow automatically from the task specification. Yet, dependent on the application, particular control laws and estimators can be chosen from state of the art methods.

The generic nature of the task specification and of the conversion to a task controller and estimator enables the development of programming support, which is needed to enhance the penetration of sensor based robots in industrial as well as domestic and service environments.

In this paper the constraint-based approach has been applied to a large variety of robot systems (mobile robots, multiple robot systems and dynamic human-robot interaction), various sensor systems, and different robot tasks. Both simulation and experimental results have been presented. Other potential applications include manipulation of objects using multi-finger grippers and whole arm manipulation.

Ongoing work includes (i) the development of a user friendly interface to support the task specification, (ii) extension to identification of unknown dynamic parameters such as contact stiffness and friction, and (iii) extension to systems with flexible robot links and joints as well as flexible objects.

A Index to multimedia Extensions

The multimedia extensions to this article can be found online by following the hyperlinks from www.ijrr.org.

Extension	Media type	Description
1	Video	Experiment of human-robot co-manipulation task.
2	Video	Experiment of pick-and-place operation of a compressor screw combined with a painting task.

Table 1: Table of Multimedia extensions

Acknowledgment

All authors gratefully acknowledge the financial support by K.U.Leuven's Concerted Research Action GOA/05/10. Tinne De Laet is a Doctoral Fellow of the Fund for Scientific Research–Flanders (F.W.O.) in Belgium. The authors would like to thank Paul Fisette, Jean-Claude Samin and Tony Postiau from the Université Catholique de Louvain, department of mechanical engineering, for stimulating discussions on the link between robot task specification and multibody dynamics.

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

[3] Y. Bar-Shalom and X. Li. *Estimation and Tracking, Principles, Techniques, and Software*. Artech House, Norwood, MA, 1993.

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

[5] J. Chen and A. Zelinsky. Programing by demonstration: Coping with suboptimal teaching actions. *The International Journal of Robotics Research*, 22(5):299–319, 2003.

[6] J. Critchley and K. S. Anderson. A generalized recursive coordinate reduction method for multibody system dynamics. *International Journal for Multiscale Computational Engineering*, 1(2–3):181–199, 2003.

[7] J. G. de Jalón and E. Bayo. *Kinematic and Dynamic Simulation of Multibody Systems—The Real Time Challenge*. Springer, 1993.

[8] T. De Laet and J. De Schutter. Control schemes for constraint-based task specification in the presence of geometric uncertainty using auxiliary coordinates. Internal report 07RP001, Department of Mechanical Engineering, Katholieke Universiteit Leuven, Belgium, 2007.

[9] J. De Schutter. Improved force control laws for advanced tracking applications. In *Proceedings of the 1988 IEEE International Conference on Robotics and Automation*, pages 1497–1502, Philadelphia, PA, 1988. ICRA88.

[10] J. De Schutter, H. Bruyninckx, S. Dutré, J. De Geeter, J. Katupitiya, S. Demey, and T. Lefebvre. Estimating first-order geometric parameters and monitoring contact transitions during force-controlled compliant motions. *The International Journal of Robotics Research*, 18(12):1161–1184, Dec 1999.

[11] J. De Schutter and H. Van Brussel. Compliant Motion I, II. *The International Journal of Robotics Research*, 7(4):3–33, Aug 1988.

[12] K. L. Doty, C. Melchiorri, and C. Bonivento. A theory of generalized inverses applied to robotics. *The International Journal of Robotics Research*, 12(1):1–19, 1993.

[13] A. Doucet and V. B. Tadic. Parameter Estimation in General State-Space Models using Particle Methods. *Annals of the Institute of Statistical Mathematics*, 55(2):409–422, 2003.

[14] J. Duffy. The fallacy of modern hybrid control theory that is based on "orthogonal complements" of twist and wrench spaces. *Journal of Robotic Systems*, 7(2):139–144, 1990.

[15] B. Espiau, F. Chaumette, and P. Rives. A new approach to visual servoing in robotics. *IEEE Transactions on Robotics and Automation*, 8(3):313–326, 1992.

[16] M. Fruchard, P. Morin, and C. Samson. A framework for the control of nonholonomic mobile manipulators. *The International Journal of Robotics Research*, 25(8):745–780, 2006.

[17] A. E. Gelb. *Optimal Estimation*. MIT Press, Cambridge, MA, 3rd edition, 1978.

[18] N. Hogan. Impedance control: An approach to manipulation. Parts I-III. *Transactions of the ASME, Journal of Dynamic Systems, Measurement, and Control*, 107:1–24, 1985.

[19] N. Hogan. Stable execution of contact tasks using impedance control. In *Proceedings of the 1987 IEEE International Conference on Robotics and Automation*, pages 1047–1054, Raleigh, NC, 1987.

[20] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME, Journal of Basic Engineering*, 82:34–45, 1960.

[21] H. Kazerooni. On the robot compliant motion control. *Transactions of the ASME, Journal of Dynamic Systems, Measurement, and Control*, 111:416–425, 1989.

[22] O. Khatib. Dynamic control of manipulators in operational space. In *6th IFtoMM Congr. on Theory of Machines and Mechanisms*, pages 15–20, 1983.

[23] C. A. Klein and C. H. Huang. Review of pseudoinverse control for use with kinematically redundant manipulators. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:245–250, 1983.

[24] T. Kröger, B. Finkemeyer, and F. M. Wahl. A task frame formalism for practical implementations. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, pages 5218–5223, New Orleans, U.S.A., 2004. ICRA2004.

[25] J. C. Latombe. *Robot motion planning*, volume 124 of *Int. Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, MA, 1991.

[26] T. Lefebvre, H. Bruyninckx, and J. De Schutter. Polyhedral contact formation modeling and identification for autonomous compliant motion. *IEEE Transactions on Robotics and Automation*, 19(1):26–41, 2003.

[27] T. Lefebvre, H. Bruyninckx, and J. De Schutter. Polyhedral contact formation identification for autonomous compliant motion: Exact nonlinear Bayesian filtering. *IEEE Transactions on Robotics and Automation*, 21(1):124–129, 2005.

[28] H. Lipkin and J. Duffy. Hybrid twist and wrench control for a robotic manipulator. *Transactions of the ASME, Journal of Mechanisms, Transmissions, and Automation in Design*, 110:138–144, 1988.

[29] G. Liu and Z. Li. A unified geometric approach to modeling and control of constrained mechanical systems. *IEEE Transactions on Robotics and Automation*, 18(4):574–587, 2002.

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

[31] Y. Mezouar and F. Chaumette. Path planning for robust image-based control. *IEEE Transactions on Robotics and Automation*, 18(4):534–549, 2002.

[32] J. K. Mills and A. A. Goldenberg. Force and position control of manipulators during constrained motion tasks. *IEEE Transactions on Robotics and Automation*, 5(1):30–46, 1989.

[33] Y. Nakamura. *Advanced robotics: redundancy and optimization*. Addison-Wesley, Reading, MA, 1991.

[34] J. Park and O. Khatib. Multi-link multi-contact force control for manipulators. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 3624–3629, Barcelona, Spain, 2005. ICRA2005.

[35] M. Raibert and J. J. Craig. Hybrid position/force control of manipulators. *Transactions of the ASME, Journal of Dynamic Systems, Measurement, and Control*, 102:126–133, 1981.

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

[37] H. Tanizaki. *Nonlinear Filters. Estimation and Applications*. Springer-Verlag, 1996.

[38] D. E. Whitney. Resolved motion rate control of manipulators and human prostheses. *IEEE Transactions on Man-Machine Systems*, 10(2):47–53, 1969.

[39] W. Witvrouw, J. Najéra, J. De Schutter, and C. Laugier. Integrating assembly planning with compliant control. In *WAC'96*, volume 6, pages 523–528, Montpellier, France, 1996.