Extending the iTaSC Constraint-based Robot Task Specification Framework to Time-Independent Trajectories and User-Configurable Task Horizons

Wilm Decré, Herman Bruyninckx, and Joris De Schutter

***Abstract*— In constraint-based programming, robot tasks are specified and solved as optimization problems with sets of constraints and one or multiple objective functions. In our previous work, we presented (i) a generic modeling approach for geometrically complex robot tasks, including the modeling of parametric uncertainty, in order to allow the robot task programmer to specify the optimization problem without explicitly writing down the different (possibly numerous and involved) constraint equations, and (ii) methods for solving these optimization problem online in the instantaneous case (reactive control), and offline in the non-instantaneous case (trajectory planning). This paper has two contributions. First, it extends our framework to include task constraints (e.g. tracking a curve) that are not given as explicit functions of time. These constraints are highly relevant in practice, for example to facilitate time-optimal path planning combined with other constraints. Second, it extends our framework to user-configurable task horizons when solving the optimization problem, to allow task programmers to make a trade-off between computational speed and (global) task optimality. Both of these novel framework extensions are illustrated by a time-optimal laser tracing experiment.**

I. Introduction

In current commercial robot systems, robot motions are in essence specified in joint space or Cartesian space. In joint space, a motion trajectory is directly imposed on the individual robot joints. This case is mainly used for fast point-to-point motions, with a trapezoidal joint velocity profile being a typical motion profile segment.

In the Cartesian case, for example used for tool trajectory tracking, the robot motion is specified in either a tool center point (TCP) frame or a base frame and the robot joint motion is derived using the robot's inverse kinematic relation.

Constraint-based programming takes a conceptually different approach. It does not consider the robot joints or a single Cartesian frame as *centric* while specifying tasks. Instead, the core idea behind the approach is to describe a, possibly sensor-based, robot task as a **set of constraints** on a set of controllable variables of interest, denoted *output variables* $\boldsymbol{y}$, and **one or multiple objective functions**.

In the constraint-based programming *paradigm*, joint-based and Cartesian-based programming are simply two special cases of constraint-based programming, where the output variables are chosen as the joint and Cartesian frame coordinates, respectively. In the more general case, the output variables can for example consist of the position of an object in a camera image, the relative position and orientation of two workpieces that have to be assembled, ... Possible objective functions are the task execution time or the root-mean-square (RMS) actuator torques or joint velocities. On a higher discrete (state machine) level, the skill level [9], constraints and objective functions are added, removed of modified.

Early theoretical work on constraint-based programming of robot tasks was done by Ambler and Popplestone [1] and by Samson et al. [8]. Ambler and Popplestone develop a method to infer the positions of rigid-body objects based on imposed desired spatial relations between rigid bodies, motivated by automatic assembly applications. Also motion planning research in configuration space methods (see [5] for an overview of the literature) specifies the desired relative poses as the result of applying several (possibly conflicting) constraints between object features. Samson and coworkers introduce the task function approach, that models a robot task as the minimization of one or multiple given task functions.

Other related and more recent work on constraint-based programming includes the work of Mansard et al. [6], [7], that specifically focuses on solvers for fast reactive instantaneous control with inequality constraints and on efficient solvers that support task prioritization (stack of tasks), respectively.

Our own original task specification approach based on constraints was presented in [3]. This work focused mainly on (i) developing a **systematic modeling procedure** for (geometrically) complex robot tasks, including the modeling of parametric geometric uncertainty, and (ii) closing the loop from modeling to control and estimation. From a control perspective, our original work only supported instantaneous equality constraints[1] and instantaneous least-squares objective functions (such as the mean-square imposed joint velocity).

In more recent work [4], the method was denoted iTaSC[2], and the control law was generalized into a standard optimization form, that (i) allows formulating more general task objective functions (as opposed to only least-squares objective functions), (ii) supports inequality constraints in the task specification, that can be used to specify joint limits, limited sensor ranges, velocity limits, ... and can be non-instantaneous

All authors are with the Department of Mechanical Engineering, KU Leuven, Celestijnenlaan 300b, Box 02420, B-3001 Heverlee, Belgium, {wilm.decre, herman.bruyninckx, joris.deschutter} @mech.kuleuven.be

[1] inequality constraints can be *mimicked* by applying variable constraint activation or weighting, but are not supported in a generic way.

[2] **i**nstantaneous **Ta**sk **S**pecification using **C**onstraints.

as long as they are defined on independent outputs [3], and (iii) opens up the more general possibility of non-instantaneous task specification.

A highly relevant functionality that is missing in our previous work, is the support of time-independent trajectory constraints. This type of constraints typically arises in path tracking problems where the exact motion profile that is used to traverse the path is not imposed by the task programmer. **As a first contribution, this paper includes the motion profile as a variable in the optimization problem definition, allowing us to more fully follow the constraint-based programming *mindset*, which is: formulate the system limitations and the safety and task requirements (and only these) as constraints, and define one or multiple objective functions to resolve redundancy.**

Time-optimal tracking of a given path, with kinematic and dynamic constraints, is a well-studied topic [2]. In Verscheure et al. [10] this problem was reformulated as a convex constrained optimization problem. However, in contrast to this paper, here the path is fully defined on joint level, and hence redundancy of the robot system for the given task is not exploited.

As a second contribution, we extend our framework to user-configurable task horizons when solving the optimization problem. Previously, we focused on either instantaneous (where the horizon equals one time step) or on global task specification (where the horizon equals the total task execution time). A user-configurable task horizon allows task programmers to make a trade-off between computational speed and task optimality.

The paper is structured as follows. Section II introduces the theoretical background, section III discusses a time-optimal laser tracing setup. Subsequently section IV presents the main results, and finally section V summarizes the main conclusions and lists opportunities for future work.

II. Theoretical background

This section introduces the control scheme, the system variables, the underlying equations and the optimal control problem formulation.

A. Control scheme and system variables

Figure 1 depicts the general control scheme. The scheme consists of three blocks:

- plant block P consists of the robot system and its environment, with control inputs $\boldsymbol{u}$, disturbances $\boldsymbol{\chi_u}$, outputs $\boldsymbol{y}$ and measurements $\boldsymbol{z}$,
- control block C generates control signals $\boldsymbol{u}$ based on reference signals $\boldsymbol{y}_d$ and estimates of the outputs $\widehat{\boldsymbol{y}}$ and disturbances $\widehat{\boldsymbol{\chi}}_{\boldsymbol{u}}$,
- model update and estimation block $M + E$ generates estimates or *virtual measurements* $\widehat{\boldsymbol{y}}$ and $\widehat{\boldsymbol{\chi}}_{\boldsymbol{u}}$ of the outputs and disturbances, based on the observed variables $\boldsymbol{z}$ and control signals $\boldsymbol{u}$.

[3]for example: simultaneously constraining joint positions, velocities and accelerations is possible.

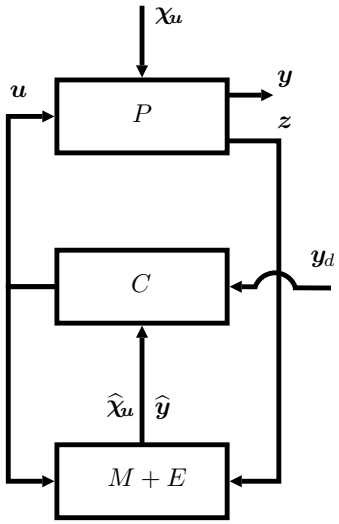


Fig. 1. General control scheme including plant *P*, controller *C*, and model update and estimation block *M*+*E*.

The different system variables are:

- $\boldsymbol{u}$: control inputs to the robot system. Depending on the particular control scheme, $\boldsymbol{u}$ represents desired joint positions, velocities, accelerations or actuation torques.
- $\boldsymbol{\chi_u}$: disturbances or uncertainties to the robot system or its environment. Here we focus on geometric uncertainty, where $\boldsymbol{\chi_u}$ can for example represent an uncertain position of a workpiece.
- $\boldsymbol{y}$: system outputs. $\boldsymbol{y}$ consists of all variables of control interest. Typical outputs are Cartesian positions, contact forces or positions in camera images.
- $\boldsymbol{z}$: measurements. All observed variables using proprioceptive or exteroceptive sensors. Typical measurements are distances, forces, joint angles and image coordinates.
- $\boldsymbol{y}_d$: reference values for $\boldsymbol{y}$. $\boldsymbol{y}_d$ can be either exogenous or include sensor or model feedback. If inequality constraints are imposed, limit values on $\boldsymbol{y}$ ($\boldsymbol{y}_{min}$ and $\boldsymbol{y}_{max}$) and/or its derivatives are added to these reference values.

B. Equations

The underlying equations are given by:

1) The *robot system equation* relates the rate of change of the robot system state to the control input:

$$\dot{\boldsymbol{x}} = \boldsymbol{s}\left(\boldsymbol{x}, \boldsymbol{u}\right), \tag{1}$$

with $\boldsymbol{x}$ typically chosen as $\boldsymbol{x} = \left(\boldsymbol{q}\ \dot{\boldsymbol{q}}\right)^T$, for rigid robot manipulators, with $\boldsymbol{q}$ the robot joint coordinates and $\boldsymbol{u}$ the control input. $\dot{x} \triangleq \dfrac{dx}{dt}$ denotes the time derivative of x.

2) The *output equation* relates the system output to the joint coordinates $\boldsymbol{q}$ and feature coordinates $\boldsymbol{\chi_f}$, which are auxiliary coordinates that facilitate the task modeling:

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

3) The system output consists of all signals of control interest. Hence, we define the *constraint equation*:

$$\boldsymbol{y} = \boldsymbol{y}_d, \tag{3}$$

that imposes desired values to the system output.

4) Finally, the *loop closure equation* expresses the relation between $\boldsymbol{q}$, $\boldsymbol{\chi_f}$ and $\boldsymbol{\chi_u}$.

$$\boldsymbol{l}\left(\boldsymbol{q}, \boldsymbol{\chi_f}, \boldsymbol{\chi_u}\right) = \boldsymbol{0}. \tag{4}$$

C. Optimal control problem

The control block generates control signals $\boldsymbol{u}$ based on estimates of the outputs and disturbances, and reference signals $\boldsymbol{y}_d$.

$\boldsymbol{u}$ is obtained by solving an optimal control problem, which for the case of a time-dependent constraint equation becomes[4]:

$$\begin{aligned} &\text{minimize} && r(\boldsymbol{\mu}(t), \boldsymbol{\nu}_1(t), \boldsymbol{\nu}_2(t), t_0, t_f) \\ &\text{subject to} && \end{aligned} \tag{5}$$

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{s}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \\ \boldsymbol{y}(t) &= \boldsymbol{f}(\boldsymbol{q}(t), \boldsymbol{\chi_f}(t)) \\ \boldsymbol{0} &= \boldsymbol{l}\left(\boldsymbol{q}(t), \boldsymbol{\chi_f}(t)\right) \\ \boldsymbol{y}(t) &= \boldsymbol{y}_d(t) + \boldsymbol{\mu}(t) \\ \boldsymbol{y}^{(i)}(t) &\leq \boldsymbol{y}_{i,max} + \boldsymbol{\nu}_1(t) \\ \boldsymbol{y}^{(i)}(t) &\geq \boldsymbol{y}_{i,min} + \boldsymbol{\nu}_2(t), \end{aligned} \tag{6}$$

$\forall t \in [t_0, t_f]$ with t_0 and t_f the start and end time of the considered task horizon. Hence, the horizon length equals $t_f - t_0$. The inequality constraints are stacked for all values of $i(\in \mathbb{Z})$ for which constraints are specified on $\boldsymbol{y}^{(i)} \triangleq \frac{d^i}{dt^i}\boldsymbol{y}$ and with $\boldsymbol{y}_{i,min}$ and $\boldsymbol{y}_{i,max}$ the imposed lower and upper limits on $\boldsymbol{y}^{(i)}$. $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$, which are also variables solved via the optimization problem, are residues that equal the errors on the constraints. r can consist of a combination of Mayer and Lagrange terms. Since violations of the constraints are allowed, the constraints are *soft*. For *hard* constraints, the corresponding ν or μ are omitted. If hard constraints are internally conflicting, (5) becomes infeasible, which can be detected by the solver. In the simplest case, an emergency stop can then be triggered. Alternatively, the values of the residues $\boldsymbol{\nu}$ and $\boldsymbol{\mu}$ can be monitored and appropriate actions can be taken if certain limits are exceeded. In problem (5), the constraint equations are written as trajectory constraints, that is, values are imposed on $\boldsymbol{y}$ for each time step of the task. This is a useful type of constraint, for example when tracing a trajectory in operational space, which is the case presented in [4] using a 7 degree-of-freedom robot manipulator. However, for example if the robot task involves picking up an object, specifying the entire trajectory is often unnecessary and can result in suboptimal solutions. In this case, we can also define constraint equations at discrete time steps, typically at the boundaries of the time window (initial and end constraints). State initial and end constraints are of the form:

$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0 \tag{7}$$

$$\boldsymbol{x}(t_f) = \boldsymbol{x}_f. \tag{8}$$

To strip the time-dependence from the constraint equation, we define the path coordinate p as the degree of advancement on the specified path. Although this is not mathematically required, we more specifically define p as the distance traveled along the path, which has the benefit that p has a straightforward physical interpretation.

The modified constraint equation becomes:

$$\boldsymbol{y}(t) = \boldsymbol{y}_d(p(t)), \tag{9}$$

indicating that the output should be *on* the path specified by $\boldsymbol{y}_d$, but the exact point on the path is not imposed. Additionally $p(t)$ can be included in the objective function and/or constraints can be imposed on $p(t)$. For example, (i) adding the constraint $p(t) \geq 0$ imposes that $\boldsymbol{y}_d$ should be traversed in the positive direction, disallowing moving backwards on the path, (ii) adding $-\int_{t_o}^{t_f} \dot{p} dt$ to the objective function would maximize the velocity along the path, and hence minimize the time required to traverse the path. The resulting optimization problem then becomes:

$$\begin{aligned} &\text{minimize} && r(\boldsymbol{\mu}(t), \boldsymbol{\nu}_1(t), \boldsymbol{\nu}_2(t), t_0, t_f, p(t)) \\ &\text{subject to} && \end{aligned} \tag{10}$$

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{s}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \\ \boldsymbol{y}(t) &= \boldsymbol{f}(\boldsymbol{q}(t), \boldsymbol{\chi_f}(t)) \\ \boldsymbol{0} &= \boldsymbol{l}\left(\boldsymbol{q}(t), \boldsymbol{\chi_f}(t)\right) \\ \boldsymbol{y}(t) &= \boldsymbol{y}_d(p(t)) + \boldsymbol{\mu}(t) \\ \boldsymbol{y}^{(i)}(t) &\leq \boldsymbol{y}_{i,max} + \boldsymbol{\nu}_1(t) \\ \boldsymbol{y}^{(i)}(t) &\geq \boldsymbol{y}_{i,min} + \boldsymbol{\nu}_2(t) \\ \boldsymbol{x}(t_0) &= \boldsymbol{x}_0 \\ p(t_0) &= p_0, \end{aligned} \tag{11}$$

where (11) is added to impose the initial position on the path.

III. Example - setup

As a validation application we consider a laser tracing task. Four cases using this setup, differing in the type and number of constraints, are presented in section IV.

The task is inspired by [3], [4]. As illustrated in figure 2 (a), a laser is mounted on the end effector of a KUKA LWR; a robot manipulator with seven revolute joints. The task goal is to trace a specified path with the laser on a plane. In this example the specified path is the Lissajous curve shown in figure 3.

The feature coordinates are defined as:

$$\boldsymbol{\chi_f} = \begin{pmatrix} x_l & y_l & \phi_l & \theta_l & \psi_l & d_l \end{pmatrix}^T, \tag{12}$$

[4]The contributions of this paper lie in new control possibilities. Hence, for clarity of the presentation, we only consider the case with no geometric uncertainty, that is, $\chi_{\boldsymbol{u}}$ is nonexisting. The methods however generalize to include the case with geometric uncertainty by using a modified loop closure equation constraint.

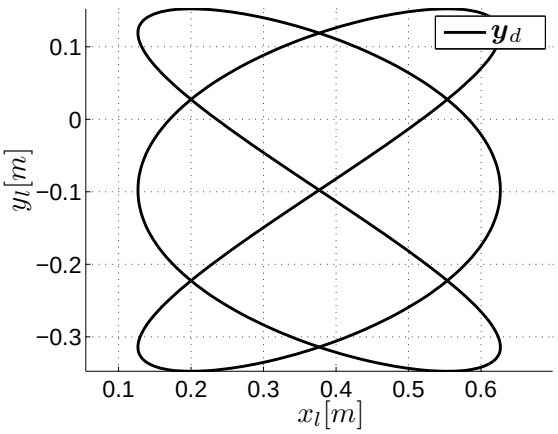


Fig. 3. Desired laser tracing trajectory.

with x_l and y_l the position of the laser dot on the plane; ϕ_l, θ_l and ψ_l the roll, pitch and yaw angles of the laser beam with respect to the plane; and d_l the distance between the laser and the plane, measured along the laser beam, as shown in figure 2 (b).

As the task goal is to trace a specified path with the laser on the plane, two outputs y_1 and y_2 representing the $x-$ and $y-$position of the laser dot on the plane are defined, which yields the output equation:

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{q}, \boldsymbol{\chi_f}) = \begin{pmatrix} x_l \\ y_l \end{pmatrix}. \tag{13}$$

The total task execution time is 5 [s], and the sampling time is chosen as 0.01 [s]

The orientation of the laser with respect to the plane, as well as the laser distance are not constrained. Their values depend on the objective function that is used to resolve redundancy. As the robot system has seven degrees of freedom and the number of output equations is only two, the robot system is highly redundant for the given task.

As objective function, we choose to trace the Lissajous curve as fast as possible. This is equivalent with maximizing $p(t_f)$ (minimizing $-p(t_f)$). Hence the objective function becomes:

$$r = -p(t_f). \tag{14}$$

IV. Example - results

This sections considers four case studies, each based on the setup of section III, with increasing complexity:

- case 1: time-optimal laser tracing with joint velocity limits and different task horizon settings,
- case 2: time-optimal laser tracing with joint and path velocity limits,
- case 3: time-optimal laser tracing with joint and path velocity and joint acceleration limits,
- case 4: time-optimal laser tracing with joint and path velocity and joint acceleration limits, and laser distance control.

These four cases are motivated as follows: case 1 exemplifies the influence of the considered task horizon on the objective function value. Case 2 extends the task to include an inequality constraint on a task coordinate, namely the velocity along the path. Case 3 extends case 2 to include joint constraints on acceleration level. Finally case 4 adds an additional equality constraint in task space (on the laser distance).

A. *case 1: joint velocity limits*

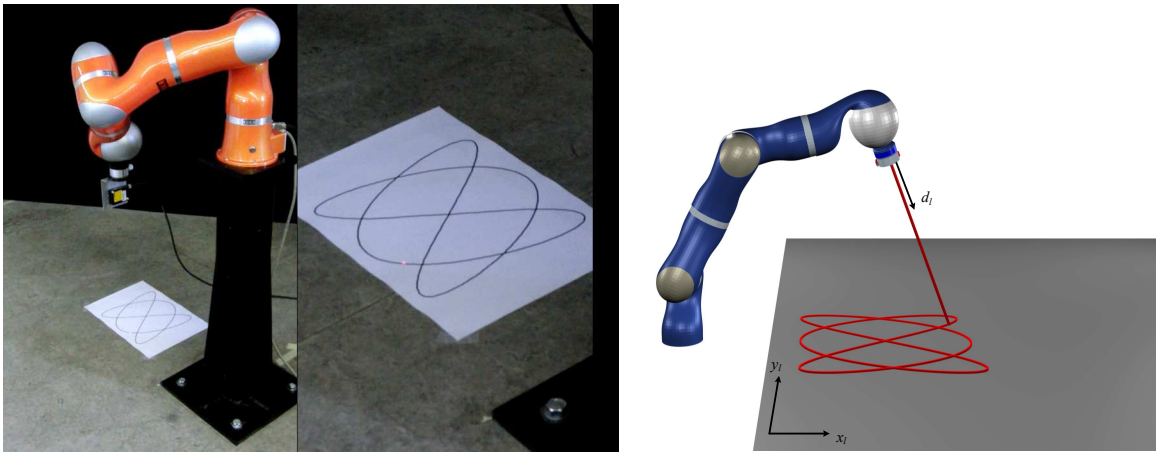
Joint velocity limits are imposed to the laser tracing task as:

$$-\pi/6\,[\text{rad/s}] \leq \dot{\boldsymbol{q}} \leq \pi/6\,[\text{rad/s}]\,. \tag{15}$$

Figure 4 depicts the results for a task horizon of one time step (0.01 [s]), showing the resulting (a) joint motion, (b) joint velocity, (c) joint acceleration, (d) feature coordinate motion, consisting of the x_l and y_l motion of the laser dot on the plane and the laser distance d_l; and (e) velocity along the Lissajous path. Because the laser is mounted along the axis of joint 7, this joint does not contribute to motion of the laser dot on the plane and is (by the solver, due to initialization) held at a constant position. Because the objective function is simply $-p(t_f)$ joint 7 is kept at a constant position -only- due to the solver initial estimate, and its motion is hence extremely sensitive to algorithm initialization. To avoid undesired and unnecessary joint motions in the 'locus' of the optimization problem, a regularization term, such as the root-mean-square control input, can be added to the objective function.

Table I illustrates the effect of the horizon length: the number of time steps that is looked into the future in the optimization problem. The computation times are obtained in a Matlab script implementation on a laptop with a $2.8GHz$ Intel Core i7 processor and $4GB$ of RAM memory. A computational speed increase with a factor of 30 to 100 can certainly be expected in an improved implementation (pre-compiled code, a real-time OS, better performing hardware).

In general there is a (modest) trend towards a longer distance traveled along the path (the laser moves farther along the Lissajous curve in the given 5 [s] task execution time) with increasing horizon length, which can intuitively be expected since the solver can look further ahead and hence has more information to plan the motion. Although this can be expected as a general trend, this can however not be guaranteed formally, since the horizon is only finite, and not covering the full motion from start to goal. The computational cost however also increases significantly, from 0.05 [s] for an horizon of one time step (0.01 [s]) to 0.8546 [s] for an horizon of 8 time steps (0.08 [s]). For the normal Lissajous the performance increase is very limited, with the traveled distance on the path going from 4.8458 [m] to 4.8837 [m]. Since it can also be expected, as a general trend, that the performance increase will be more significant if the task is more challenging for the robot system, Table I also depicts the results for a Lissajous curve scaled by a factor 2.5 (denoted 'Large Lissajous'). In this latter example, the performance increase is more significant, with the traveled distance on the path going from 5.7147 [m] to 6.0835 [m].



(a) Overview of the setup

(b) Model of the setup with feature coordinates: $\{x_l, y_l, d_l\}$

Fig. 2. Laser tracing setup with the KUKA LWR.

TABLE I

EFFECT OF HORIZON LENGTH (NUMBER OF TIME STEPS) ON THE AVERAGE COMPUTATION TIME PER STEP AND THE TOTAL DISTANCE TRAVELED ALONG THE LISSAJOUS PATH.

	horizon [# steps]	1	2	4	8
Normal	comp. time $[s]$	0.05	0.0891	0.2476	0.8546
Lissajous	distance $[m]$	4.8458	4.8458	4.8469	4.8837
Large	comp. time $[s]$	0.0540	0.1035	0.4338	1.2543
Lissajous	distance $[m]$	5.7147	5.6949	5.7204	6.0835

As underlying optimization technique we use nonlinear programming with an active set method for handling the inequality constraints. Because the underlying optimization problem is non-convex (due to the nonlinear kinematic constraints) and the applied technique is derivative-based, and hence searches for a local optimizer, the result of the optimization algorithm furthermore does not necessarily equal the global optimum.

For clarity and conciseness of the remaining discussion, in the following cases 2-4, the task horizon is fixed at one time step. These cases can however easily be considered for other horizon lengths.

B. *case 2: joint and path velocity limits*

As a second example case, we extend case 1 to include a velocity limit along the path:

$$\dot{p} \leq 0.25\,[\text{m/s}]\,. \tag{16}$$

This constraint has practical relevance if the tracing task (being a model for for example painting, welding or gluing) imposes constraints on the maximum speed at which the path can be traversed.

Figure 5 again depicts the resulting (a) joint motion, (b) joint velocity, (c) joint acceleration, (d) feature coordinate motion, consisting of the x_l and y_l motion of the laser dot on the plane and the laser distance d_l; and (e) velocity along the Lissajous path.

From the results, it is clear that for this example the imposed velocity limit along the path of $25\,[\text{cm/s}]$ is the limiting constraint during the entire motion.

C. *case 3: joint and path velocity and joint acceleration limits*

As a third example case, we extend case 2 to include joint acceleration limits. In the previous example the joint accelerations were (too) high, even over $100\,\left[\text{rad/s}^2\right]$, which is in practice of course not desirable/realizable. Joint acceleration limits are therefore added as:

$$-2\pi\,\left[\text{rad/s}^2\right] \leq \ddot{\boldsymbol{q}} \leq 2\pi\,\left[\text{rad/s}^2\right]\,. \tag{17}$$

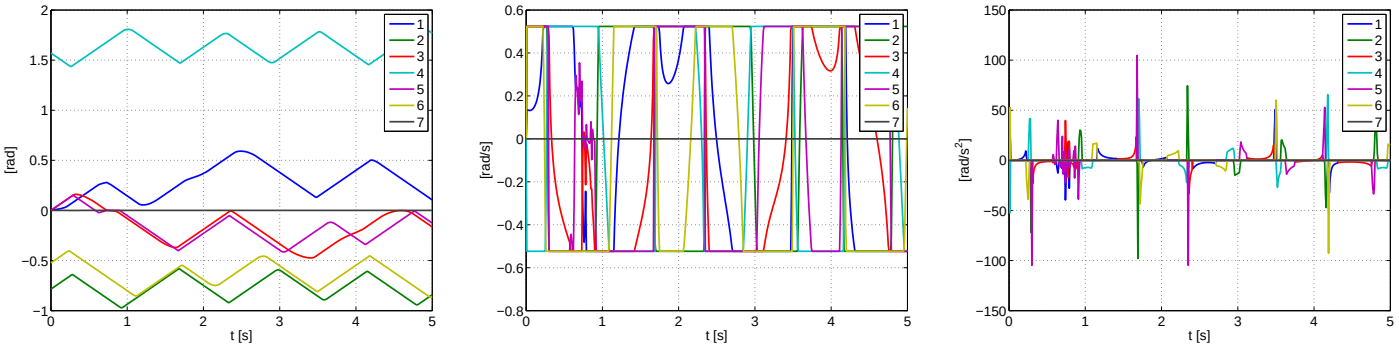
Figure 6 again depicts the resulting (a) joint motion, (b) joint velocity, (c) joint acceleration, (d) feature coordinate motion, consisting of the x_l and y_l motion of the laser dot on the plane and the laser distance d_l; and (e) velocity along the Lissajous path. Note for this example that, although the velocity limit along the path is still at its limit during the entire motion and hence the distance traveled along the Lissajous curve is identical to case 2, the joint velocities and accelerations are kept within the imposed limits.

D. *case 4: joint and path velocity and joint acceleration limits and laser distance control*

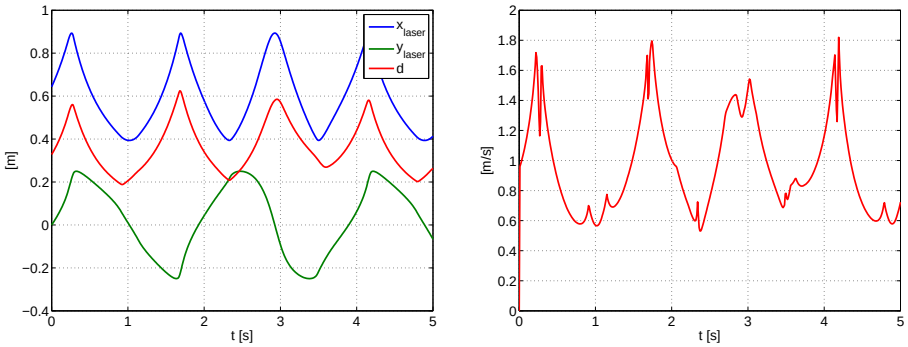
As a final case, we add an equality constraint to the laser distance to keep it fixed at its initial value. This type of constraint is often also task-inspired when the process (welding, painting) requires a certain distance of the tool with respect to the workpiece. Similarly constraints can be imposed on the angle of the laser with respect to the plane. If there is a tolerance of the laser distance, a set of inequality constraints instead of an equality constraint can of course be used. The additional constraint equation is given by:

$$d_l = d_l(0). \tag{18}$$

Figure 7 again depicts the resulting (a) joint motion, (b) joint velocity, (c) joint acceleration, (d) feature coordinate

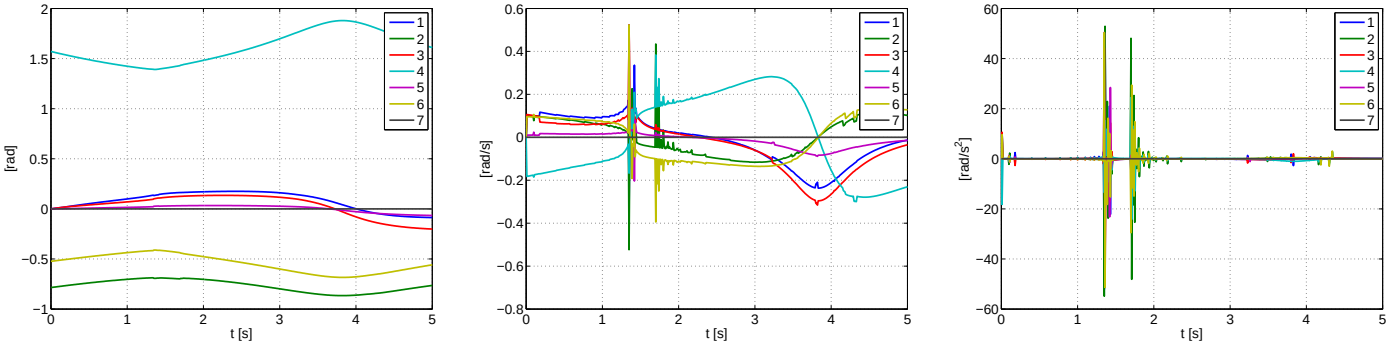


(a) joint positions, $\boldsymbol{q}$ (b) joint velocities, $\dot{\boldsymbol{q}}$ (c) joint accelerations, $\ddot{\boldsymbol{q}}$

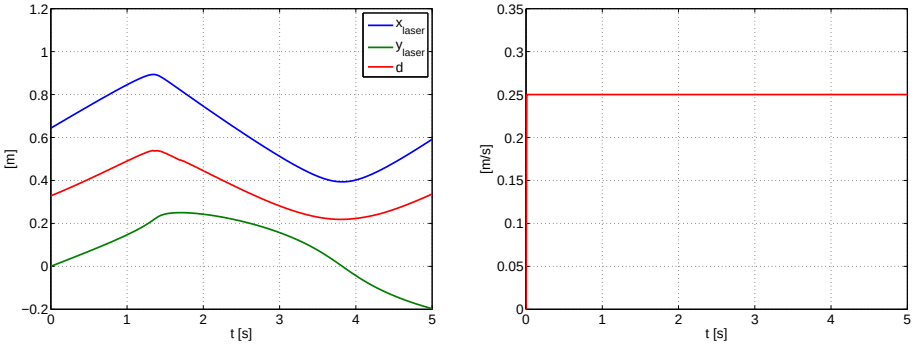


(d) feature coordinates, $\{x_l, y_l, d_l\}$ (e) velocity along the path, $\dot{p}$

Fig. 4. Case 1: time-optimal laser tracing with joint velocity inequality constraints.

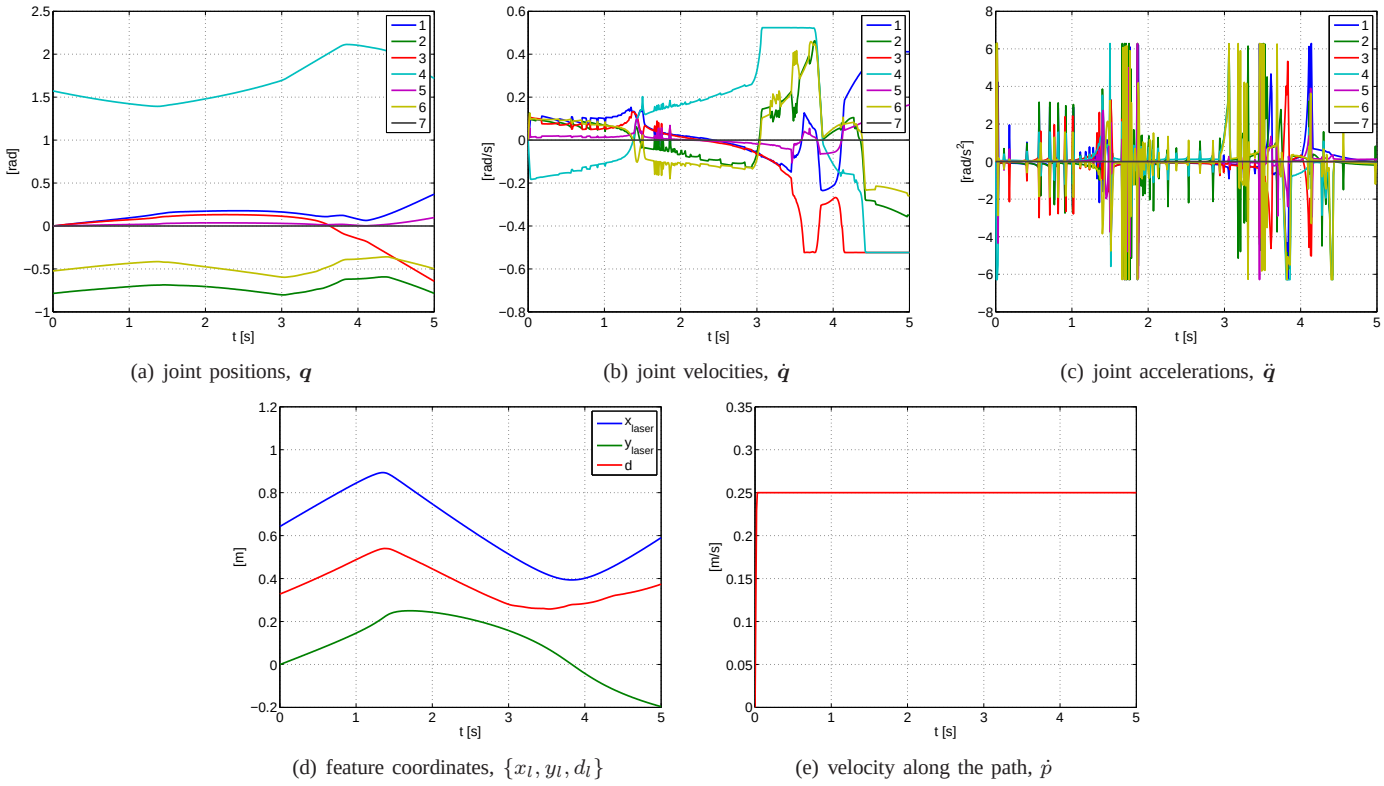


(a) joint positions, $\boldsymbol{q}$ (b) joint velocities, $\dot{\boldsymbol{q}}$ (c) joint accelerations, $\ddot{\boldsymbol{q}}$



(d) feature coordinates, $\{x_l, y_l, d_l\}$ (e) velocity along the path, $\dot{p}$

Fig. 5. Case 2: time-optimal laser tracing with joint and path velocity inequality constraints.



(a) joint positions, $\boldsymbol{q}$ (b) joint velocities, $\dot{\boldsymbol{q}}$ (c) joint accelerations, $\ddot{\boldsymbol{q}}$

(d) feature coordinates, $\{x_l, y_l, d_l\}$ (e) velocity along the path, $\dot{p}$

Fig. 6. Case 3: time-optimal laser tracing with joint and path velocity and joint acceleration inequality constraints.

motion, consisting of the x_l and y_l motion of the laser dot on the plane and the laser distance d_l; and (e) velocity along the Lissajous path. Figure 7 (d) clearly depicts that the laser distance is kept at a constant value. Note that, in this case, the velocity along the path is not the limiting factor around 1.4 [s] (figure 7 (e)).

V. Conclusion and Future Work

In this paper we extended our iTasc constraint-based robot task specification framework to time-independent trajectories and user-configurable task horizons. We illustrated the new framework extensions by a laser tracing application. Case 1 of the results showed that a larger task horizon as a general trend leads to improved overall objective function values, however at the cost of increased computation time. As practical guidelines we can conclude: (i) for online use the horizon is limited by the imposed timing constraints, (ii) for offline use different horizons can be tested and a trade-off curve can be made to balance computation time with the overall objective function value. As mentioned in section IV, the applied optimization technique is derivative-based, and hence searches for a local minimum. As future work, global path planning approaches can be coupled to our approach to provide a better initialization of the solver, in order to improve the quality of the solution. Cases 2 to 4 further illustrate the versatility of the approach, by showing a time-optimal laser tracing application with joint velocity limits, joint acceleration limits, and task-space equality and inequality constraints.

VI. Acknowledgements

The authors gratefully acknowledge the financial support by K.U. Leuven's Concerted Research Action GOA/2010/011 *Global real-time optimal control of autonomous robots and mechatronic systems* and by the European Integrated Project (European Seventh Framework Programme) Rosetta. This work also benefits from KU Leuven-BOF PFV/10/002 Center-of-Excellence Optimization in Engineering (OPTEC).

References

[1] A. P. Ambler and R. J. Popplestone. Inferring the positions of bodies from specified spatial relationships. *AI*, 6:157–174, 1975.

[2] J. E. Bobrow, S. Dubowsky, and J. S. Gibson. Time-optimal control of robotic manipulators along specified paths. *The International Journal of Robotics Research*, 4(3):3–17, 1985.

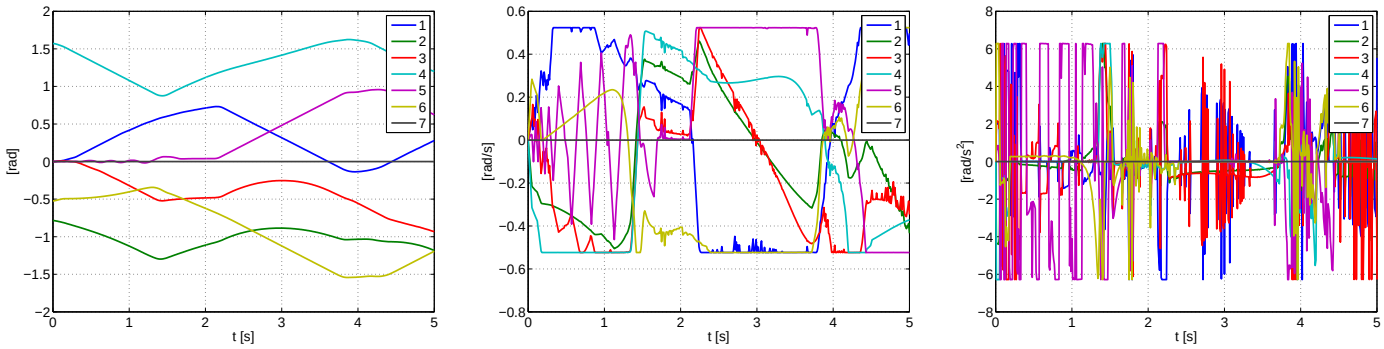
[3] J. De Schutter, T. De Laet, J. Rutgeerts, W. Decré, R. Smits, E. Aertbeliën, K. Claes, and H. Bruyninckx. Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty. *The International Journal of Robotics Research*, 26(5):433–455, 2007.

[4] W. Decré, R. Smits, H. Bruyninckx, and J. De Schutter. Extending iTaSC to support inequality constraints and non-instantaneous task specification. In *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*, pages 964–971, Kobe, Japan, 2009.

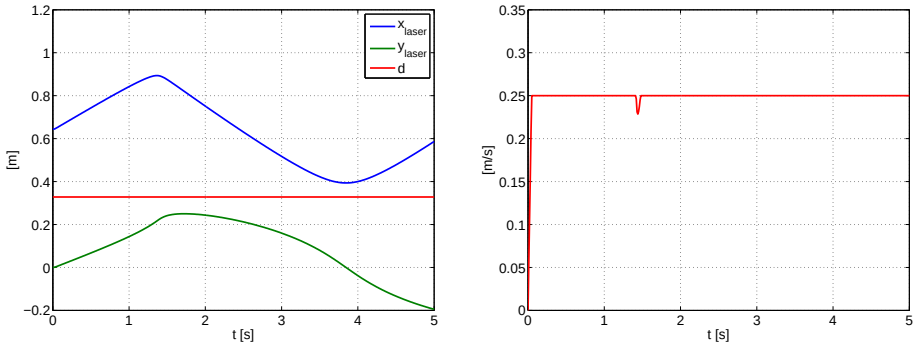
[5] J. C. Latombe. *Robot motion planning*, volume 124 of *Int. Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, MA, 1991.

[6] N. Mansard and O. Khatib. Continuous control law from unilateral constraints. In *Proceedings of the 2008 IEEE International Conference on Robotics and Automation*, Pasadena, California, U.S.A., 2008.

[7] N. Mansard, O. Stasse, P. Evrard, and A. Kheddar. A versatile generalized inverted kinematics implementation for collaborative working humanoid robots: The stack of tasks. In *Proceedings of the 2009 International Conference on Advanced Robotics*, Munich, Germany, 2009.



(a) joint positions, $\boldsymbol{q}$ (b) joint velocities, $\dot{\boldsymbol{q}}$ (c) joint accelerations, $\ddot{\boldsymbol{q}}$



(d) feature coordinates, $\{x_l, y_l, d_l\}$ (e) velocity along the path, $\dot{p}$

Fig. 7. Case 4: time-optimal laser tracing with joint and path velocity, and joint acceleration inequality constraints and laser distance equality constraint.

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

[9] R. Smits. *Robot skills: design of a constraint-based methodology and software support*. PhD thesis, Department of Mechanical Engineering, Katholieke Universiteit Leuven, Belgium, May 2010.

[10] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl. Time-optimal path tracking for robots: A convex optimization approach. *Automatic Control, IEEE Transactions on*, 54(10):2318 –2327, oct. 2009.