Time-Optimal Online Trajectory Generator for Robotic Manipulators

Francisco Ramos, Mohanarajah Gajamohan, Nico Huebel and Raffaello D'Andrea

***Abstract*— This work presents a trajectory generator for time-optimal maneuvering of a robotic manipulator in joint-space. This generator allows to adapt these maneuvers depending on external events received by the robot (e.g. from sensors or human input). Closed-form equations for these trajectories are provided, resulting on a computationally light algorithm which can update the trajectory at any time taking into account the current state of the robot and the final desired state.**

The contribution of this paper is an implementation of the algorithm which is available under LGPL for the use of the community. This implementation has been designed to work with KUKA Fast Research Interface (FRI). As a test-bed, a KUKA Light Weight Robot with seven degrees-of-freedom was used, and the experimental results are summarized in this article. The source code of the trajectory generator is available at `https://github.com/IDSCETHZurich/trajectory-generator`

I. INTRODUCTION

Trajectory design is a topic of capital importance in industrial robotics and, as such, it has been developing since first robots were ever released and there are chapters devoted to the generation of trajectories in almost every textbook on robotics [1]. Their importance is tightly related to productivity in assembly chains (the faster the trajectories, the higher the production), but also to the durability of the robot elements, such as motors, or the avoidance of vibrations at the tip of the arm.

In this paper we deal with online generation of time-optimal trajectories. While traditional approaches design the trajectory offline [2], [3] and execute it afterwards ignoring any new information regarding the environment, the online trajectory generation approach attempts to generate trajectories that can react instantaneously to unforeseen external events. This problem is attracting considerable attention lately [4] as robots are beginning to operate outside of the factories where they performed repetitive tasks in structured environments. New applications demand from them the ability to interact with their environments, or at least, react to events that happen in them [5].

Therefore, the trajectories have to be modified during execution time, depending on external factors such as sensor measurements or human commands. In [6], [7], Kröger and Wahl show a general solution to this problem. Every time instant they calculate the next reference position based on the current state of the robot and the final desired state. While their work proposes a very general solution, it also requires a not-negligible amount of calculations at every time instant, and the classification stage grows exponentially with the number of derivatives accounted for in the robot state model. In this work we take a different approach that saves computation time. We only recalculate the trajectories, whenever a new event arrives to the trajectory generator, while we can get the position from evaluating a polynomial at every time step. Furthermore, the classification depends only on two parameters that can be calculated in a straightforward manner.

The article is structured as follows. First, Section II introduces the problem framework and the notation and provides a general structure for the solution. Then, the velocity profiles of a specific case are calculated in Section III, and Section IV states a procedure for synchronizing the trajectories of the joints in a maneuver. Finally, the experimental setup is described and the experimental results are analyzed in Section V before conclusions are drawn.

II. TIME-OPTIMAL TRAJECTORIES

We aim to find a family of trajectories that solve the problem of moving a robot manipulator with m joints from an initial state to a final state in a time-optimal fashion. We define the state of the system at instant t as

$$\mathbf{X}(t) = \left({}^1\mathbf{x}(t), {}^2\mathbf{x}(t), ..., {}^m\mathbf{x}(t)\right), \tag{1}$$

where

$${}^l\mathbf{x}(t) = \left({}^lp(t),\ {}^lv(t)\right) \tag{2}$$

represents the state of a single joint l at time t, which consists of position, ${}^lp(t)$ and velocity, ${}^lv(t)$, of the joint. In addition, we define the acceleration of joint l, ${}^la(t)$, as the control input to the system.

These variables are bounded by the following joint space and control input limits

$$\left.\begin{array}{l} {}^la_m \leq {}^la(t) \leq {}^la_M \\ {}^lv_m \leq {}^lv(t) \leq {}^lv_M \\ {}^lp_m \leq {}^lp(t) \leq {}^lp_M \end{array}\right\} \forall t \in (0, t_f)\,. \tag{3}$$

We only consider the kinematic model of the system in this work. Hence, the maneuver can be decoupled for each joint and calculated separately, so we will drop the l index and assume that we are calculating the time-optimal maneuver for a generic joint. We also assume that the robot cannot collide with itself as long as we are moving within the joint space, which is a common design requirement for industrial manipulators [8].

F. Ramos is with the School of Industrial Engineering, University of Castilla-La Mancha, 13071 Ciudad Real, Spain francisco.ramos@uclm.es

G. Mohanarajah, N. Huebel and R. D'Andrea are with the Institute for Dynamic Systems and Control, ETH Zurich, 8092 Zurich, Switzerland {gajan,nhuebel,rdandrea}@ethz.ch

A. Problem solution

The time optimal problem for each joint can be stated as:
Find the optimal input $a^*(t)$ that minimizes

$$\int_0^{t_f} dt$$

subject to the following system equations

$$v(t) = \frac{dp(t)}{dt} \quad a(t) = \frac{dv(t)}{dt} \tag{4}$$

the initial and final constraints

$$\begin{array}{l} \mathbf{x}_i = \mathbf{x}(0) = (p_i,\ v_i) \\ \mathbf{x}_f = \mathbf{x}(t_f) = (p_f,\ v_f)\,, \end{array} \tag{5}$$

and the state and input constraints given by equation (3).

This is a well-known control problem that can be solved using Pontryagin's Minimum Principle [9] and the direct adjoining approach to handle the constraints. Then, from corollary 5.2 in [10], it can be demonstrated that the the control input that provides a time-optimal solution is a piecewise bang-zero-bang profile. The state and control profiles are given by the following polynomial equations

$$p_j(t) = \tfrac{1}{2}a_{j0}(t-T_j)^2 + v_{j0}(t-T_j) + p_{j0} \tag{6}$$
$$v_j(t) = a_{j0}(t-T_j) + v_{j0} \tag{7}$$
$$a_j(t) = a_{j0}, \tag{8}$$

where p_{j0}, v_{j0} and a_{j0} are the initial values of the states and the control input in the j-th piece, and T_j is the initial time of the piece. Assuming that actuator limits are identical in both positive and negative directions, the constraints from (3) become

$$|a(t)| \le a_M,\ \ |v(t)| \le v_M,\ \ |p(t)| \le p_M,\ \forall t \in (0, t_f), \tag{9}$$

B. Note on the final velocities

The use of final velocities different from zero raises two additional problems, as not all states that satisfy (9) can be reached without implying a previous or posterior violation of those same constraints:

- *Non-decelerable final state*: these states are reachable from within the feasible region of the state-space, but they will conflict with the position constraints after the end of the trajectory. Assuming that our final velocity is v_f, the minimum distance covered when decelerating to zero would be

$$\Delta p_{dec} = \frac{v_f \cdot |v_f|}{2a_M}. \tag{10}$$

  As the position at rest must be within the limits given by (9), it follows that

$$|p_f + \Delta p_{dec}| \le p_M, \tag{11}$$

  or we would not be able to stop the movement without violating the position constraints.

- *Non-reachable final state*: these states are within the feasible region of the phase space defined by (9), but cannot be reached without leaving that feasible region previously. That is, the distance needed to accelerate to v_f from 0 is given by Eq. (10), with $|\Delta p_{acc}| = |\Delta p_{dec}|$. Therefore, for the final state to be reachable, the position, at rest, from where we approach it must fulfil

$$|p_f - \Delta p_{acc}| \le p_M, \tag{12}$$

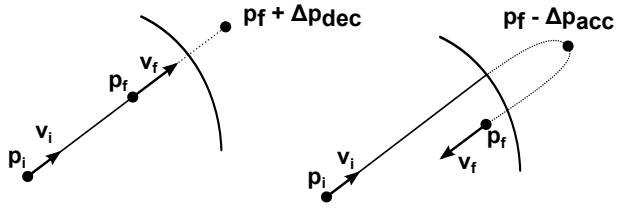


Fig. 1: Outline of non-decelerable and non-reachable final states due to final velocities

From conditions (11) and (12) the following additional feasibility condition can be derived for the desired final state of the trajectory can be joined in a single statement as

$$\max\left(|p_f + \Delta p_{dec}|\,, |p_f - \Delta p_{acc}|\right) \le p_M. \tag{13}$$

III. VELOCITY PROFILES

Depending on the values of the initial state and the final desired state, we will obtain different trajectory shapes. In this work we will use the velocity profile to classify the trajectories into three cases: critical, over-critical and under-critical. For simplification, both initial and final velocities of the trajectory will be considered positive in the analysis, but the results extend to any pair of velocities within the limits given by (9) and (13).

A. Critical profile

We define a critical profile that consists of a linear velocity profile between initial and final velocities at a constant acceleration a_M (see Fig. 2a). This profile defines a threshold between accelerating and decelerating from the initial state. Let us define $\Delta p = p_f - p_i$ and $\Delta v = v_f - v_i$. Then the duration of this profile is

$$t_f^{crit} = \frac{\tilde{s}_v \Delta v}{a_M} \tag{14}$$

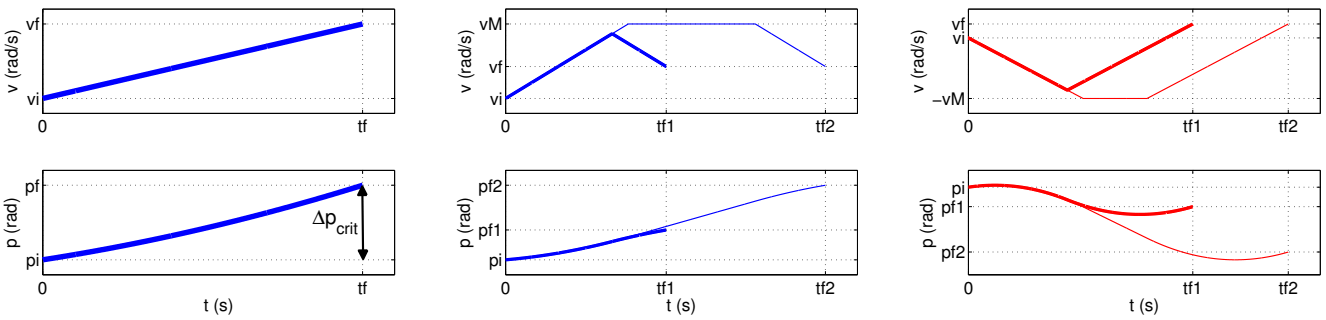
and the distance covered during the maneuver is given by

$$\Delta p^{crit} = \bar{v} \cdot t_f^{crit} = \frac{\tilde{s}_v \left(v_f^2 - v_i^2\right)}{2a_M}, \tag{15}$$

where $\tilde{s}_v = sign(\Delta v)$.

In the case $\Delta p = \Delta p^{crit}$, no further calculations are needed, and the piecewise polynomial solution consists of a single piece

$$\left.\begin{array}{lll} a_1(t) &=& \tilde{s}_v a_M \\ v_1(t) &=& \tilde{s}_v a_M t + v_i \\ p_1(t) &=& \frac{1}{2}\tilde{s}_v a_M t^2 + v_i t + p_i \end{array}\right\} \quad t \in \left[0, t_f^{crit}\right]. \tag{16}$$



(a) Critical velocity profile (b) Over-critical profiles (c) Under-critical profiles

Fig. 2: Velocity profiles classification depending on the comparison between Δp and Δp^{crit}.

The duration of a critical profile is the minimum time that a trajectory with such values of initial and final velocity can take. When the distance to cover is longer or even shorter than Δp^{crit}, the duration of the trajectory will be longer than the critical, as we will discuss in the following subsections.

B. Over-critical profile ($\Delta p > \Delta p^{crit}$)

In this case, as the distance to cover is longer than the critical, the maneuver will require more time to be accomplished. Let us first assume that the velocity profile is triangular and, hence, is divided into two parts: 1) acceleration, and 2) deceleration (see Fig. 2b). The peak velocity achieved during the motion is given by

$$v_{po} = \sqrt{\Delta p \cdot a_M + \frac{1}{2}\left(v_f^2 + v_i^2\right)}. \tag{17}$$

The triangular profile will be feasible if $v_{po} < v_M$. Otherwise, the system will follow a trapezoidal profile with three parts: 1) acceleration, 2) constant speed at v_M, and 3) deceleration (see Fig. 2b). Equations for both profiles will be treated in Section III-D.

C. Under-critical profile ($\Delta p < \Delta p^{crit}$)

This case requests a shorter position movement and, therefore, we may be inclined to think that its duration should be also shorter. However, the final velocity restriction causes the joint first to move backwards to reach point $p_i^{crit} = p_f - \Delta p^{crit}$ at velocity v_i, and then perform a critical maneuver in order to achieve the final desired state. Hence, the total trajectory will also be longer than the critical.

However, the profile shape is similar to that of the over-critical profile. Again, we determine the profile to be triangular or trapezoidal (see Fig. (2c))depending on the peak velocity, which, in this case, is given by

$$v_{pu} = -\sqrt{\frac{1}{2}\left(v_f^2 + v_i^2\right) - \Delta p \cdot a_M}. \tag{18}$$

D. Equations of a generic non-critical profile

Under and over-critical profiles can be expressed in a single formulation by defining the sign of the trajectory as

$$\tilde{s} = sign\left(\Delta p - \Delta p^{crit}\right) \tag{19}$$

and the peak speed (assuming a triangular profile) as

$$v_p = \sqrt{\frac{1}{2}\left(v_f^2 + v_i^2\right) + \tilde{s} \cdot \Delta p \cdot a_M}. \tag{20}$$

Equation (19) also gives the sign of the initial acceleration of the profile, while (20) provides a criterion to classify the profile.

1) Triangular profile ($v_p \leq v_M$): This is a bang-bang acceleration profile with two parts:

- First piece, $t \in (0, T_1)$, from initial velocity to peak velocity

$$\begin{array}{rcl} a_1(t) & = & \tilde{s} a_M \\ v_1(t) & = & \tilde{s} a_M t + v_i \\ p_1(t) & = & \frac{1}{2}\tilde{s} a_M t^2 + v_i t + p_i. \end{array} \tag{21}$$

- Second piece, $t \in (T_1, t_f]$, from peak velocity to final velocity

$$\begin{array}{rcl} a_2(t) & = & -\tilde{s} a_M \\ v_2(t) & = & -\tilde{s} a_M (t - T_1) + \tilde{s} v_p \\ p_2(t) & = & -\frac{1}{2}\tilde{s} a_M (t - T_1)^2 + \tilde{s} v_p (t - T_1) + p_1(T_1). \end{array} \tag{22}$$

- Switching times: to completely define this trajectory we only need to calculate T_1 and t_f, which are given by

$$\begin{array}{rcl} T_1 & = & \dfrac{\tilde{s} v_p - v_i}{\tilde{s} a_M} \\ t_f & = & T_1 + \dfrac{v_f - \tilde{s} v_p}{-\tilde{s} a_M}. \end{array} \tag{23}$$

2) Trapezoidal profile ($v_p > v_M$): In this case, the distance to cover is long enough such that the velocity reaches its maximum admissible value. Hence, the profile will have three parts:

- First piece, $t \in [0, T_1]$, from initial to maximum velocity

$$\begin{array}{rcl} a_1(t) & = & \tilde{s} a_M \\ v_1(t) & = & \tilde{s} a_M t + v_i \\ p_1(t) & = & \frac{1}{2}\tilde{s} a_M t^2 + v_i t + p_i. \end{array} \tag{24}$$

- Second piece $t \in (T_1, T_2]$, motion at constant velocity $\tilde{s} v_M$

$$\begin{array}{rcl} a_2(t) & = & 0 \\ v_2(t) & = & \tilde{s} v_M \\ p_2(t) & = & \tilde{s} v_M (t - T_1) + p_1(T_1). \end{array} \tag{25}$$

- Third piece $t \in (T_2, t_f]$, from maximum to final velocity

$$\begin{aligned} a_3(t) &= -\tilde{s}a_M \\ v_3(t) &= -\tilde{s}a_M(t-T_2) + \tilde{s}v_M \\ p_3(t) &= -\tfrac{1}{2}\tilde{s}a_M(t-T_2)^2 + \tilde{s}v_M(t-T_2) + p_2(T_2). \end{aligned} \tag{26}$$

- Switching times: T_1, T_2 and t_f can be calculated from

$$\begin{aligned} T_1 &= \frac{\tilde{s}v_M - v_i}{\tilde{s}a_M} \\ T_2 &= \frac{1}{v_M}\left[\frac{v_f^2 + v_i^2 - 2\tilde{s}v_M v_i}{2a_M} + \tilde{s}\Delta p\right] \\ t_f &= T_2 + \frac{v_f - \tilde{s}v_M}{\tilde{s}a_M}. \end{aligned} \tag{27}$$

This is a compact, straightforward formulation for the event-driven online trajectory generation problem that functions for any set of initial and final conditions (p_i, p_f, v_i and v_f) satisfying (9) and (13).

IV. TRAJECTORY SYNCHRONIZATION

Most maneuvers of the robot will have different durations for each joint trajectory and the shorter ones will already have finished their motions while the longer ones are still moving. Hence, there is no need for the shorter trajectories to be performed in a time-optimal way and they could use some of this extra time to make their profiles less demanding for the actuators, hence extending their lifetime. This procedure is what we will define as synchronization: a maneuver is synchronized if all joints arrive at their target positions at the same time instant.

To achieve this, new synchronized trajectory profiles that take the final time as an input have been developed.

The synchronization procedure consists of three steps

- Calculate the minimum time of every joint profile and choose the synchronizing time as the maximum of them

$$t_f^{sync} = \max {}^l t_f, l = 1, ..., m. \tag{28}$$

- Determine the shape of synchronized velocity profile for every joint.
- Calculate the parameters and coefficients of the profiles.

The first step is straightforward from the equations given in previous Section, and hence it will not be further detailed. However, the shape of the synchronized profile is a rich problem in terms of the choice of the design parameters. As we now relax the optimal-time requirement and only fix the final time of the trajectory, we have a new degree of freedom in the equations, which could be used in a number of ways (i.e. reducing the maximum demanded acceleration). In this paper we approach the problem by minimizing the value of the velocity at the constant velocity piece of each individual joint, v_c, such that all joints reach the final position at time t_f^{sync} while keeping the acceleration value at its maximum, a_M, during the acceleration and deceleration pieces. However, the duration of these pieces will be reduced.

With these constraints in mind, the different synchronized profiles have been outlined in Fig. 3. For the trajectory

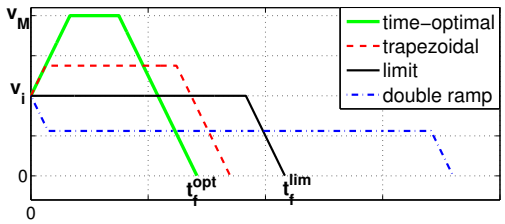


Fig. 3: Different shapes of the velocity profiles depending on the synchronization time

synchronization, we will assume that the final velocity is zero.

We can see that there is a limit case, which separates trapezoidal from double ramp profiles, when $v_c = v_i$. The duration of this limit case has to be calculated separately for each joint, yielding

$$t_f^{lim} = \frac{\Delta p}{v_i} + \frac{\tilde{s}\cdot v_i}{2a_M}. \tag{29}$$

Then, comparing this value with t_f^{sync}, we can determine the shape of the trajectory: trapezoidal profile if $t_f^{sync} < t_f^{lim}$ or double-ramp profile otherwise.

The expression of v_c will differ for each profile

- Trapezoidal profile:

$$v_c = \frac{1}{2}\left[b - \sqrt{b^2 - 4\tilde{s}\cdot a_M\Delta p - 2v_i^2}\right], \tag{30}$$

  where $b = a_M t_f^{sync} + \tilde{s}\cdot v_i$.
- Double-ramp profile:

$$v_c = \frac{\tilde{s}\cdot\Delta p - \dfrac{v_i^2}{2a_M}}{t_f^{sync} - \dfrac{v_i}{\tilde{s}\cdot a_M}}. \tag{31}$$

The equations of the synchronized trajectory will be very similar to those of the trapezoidal profile, only substituting v_M by v_c and including a new sign $\tilde{s}_t = sign(t_f^{lim} - t_f^{sync})$, that makes a general formulation valid for both trapezoidal and double-ramp profiles, yielding

- First piece, $t \in (0, T_1)$: initial to constant velocity.

$$\begin{aligned} a_1(t) &= \tilde{s}\tilde{s}_t a_M \\ v_1(t) &= \tilde{s}\tilde{s}_t a_M t + v_i \\ p_1(t) &= \tfrac{1}{2}\tilde{s}\tilde{s}_t a_M t^2 + v_i t + p_i. \end{aligned} \tag{32}$$

- Second piece, $t \in (T_1, T_2)$: constant velocity at $\tilde{s}v_M$.

$$\begin{aligned} a_2(t) &= 0 \\ v_2(t) &= \tilde{s}v_c \\ p_2(t) &= \tilde{s}v_c(t-T_1) + p_1(T_1). \end{aligned} \tag{33}$$

- Third piece, $t \in (T_2, t_f^{sync})$: from $\tilde{s}v_M$ to final velocity.

$$\begin{aligned} a_3(t) &= -\tilde{s}a_M \\ v_3(t) &= -\tilde{s}a_M(t-T_2) + \tilde{s}v_c \\ p_3(t) &= -\tfrac{1}{2}\tilde{s}a_M(t-T_2)^2 + \tilde{s}v_c(t-T_2) + p_2(T_2). \end{aligned} \tag{34}$$

Obviously, the switching times will be different for this profile than those of the optimal trajectory

$$\begin{aligned} T_1 &= \frac{\tilde{s}v_c - v_i}{\tilde{s}\tilde{s}_t a_M} \\ T_2 &= t_f^{sync} - \frac{v_c}{a_M}. \end{aligned} \tag{35}$$

V. EXPERIMENTS

To show the performance of our trajectory generator, experimental results will be presented in this section.

A. Experimental Setup

The trajectory generator has been tested on a KUKA Light Weight Robot (LWR) with seven degrees of freedom. For our software setup we were using the Robot Operating System (ROS) [11] and the Open Robot Control Software (OROCOS) [12]. We were using a ROS node to generate random joint positions at a rate of 1 s. These positions were sent to our trajectory generator OROCOS component. To couple our trajectory generator with the KUKA FRI the OROCOS LWR_FRI component was used. More information on this component can be found at `http://www.ros.org/wiki/lwr_fri`.

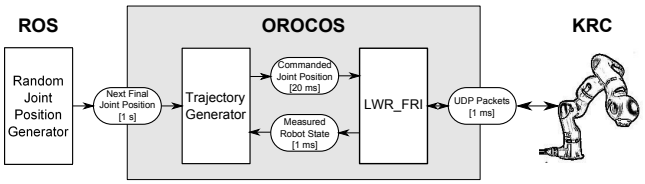
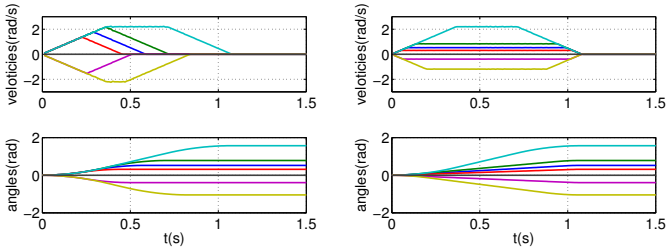


Fig. 4: Experimental setup

Whenever the trajectory generator component receives a new final desired position, it gets the current state of the robot from the LWR_FRI component and calculates the new trajectory to reach the new final position taking into account the current position and velocity of the robot. After calculating the trajectory polynomials, they can be evaluated at the current time whenever the FRI is requesting a new position. This event-based architecture was chosen to avoid synchronization problems between the FRI running on the KUKA Robot Controller (KRC) and the OROCOS and ROS components running on the remote computer.

B. Source Code of the Trajectory Generator

The trajectory generator is available at `https://github.com/IDSCETHZurich/trajectory-generator`, under LGPL. This repository contains all nodes and components required to run the setup described above (except for the FRI, which must be provided by KUKA). If a real KUKA LWR is not available or the behavior of newly developed code using the trajectory generator should first be tested without endangering the real robot, a script file offers to reroute the output to the ROS visualization environment RVIZ (`http://www.ros.org/wiki/rviz`). Please be aware that this is only a visualization of the kinematics, which does not take into account the dynamics of the robot. README files, explaining the content, setup, and use of each package, are provided as well. If you should encounter problems or have questions, feel free to contact the authors.



(a) Time-optimal (b) Synchronized

Fig. 5: Single maneuver of the robot

C. Results

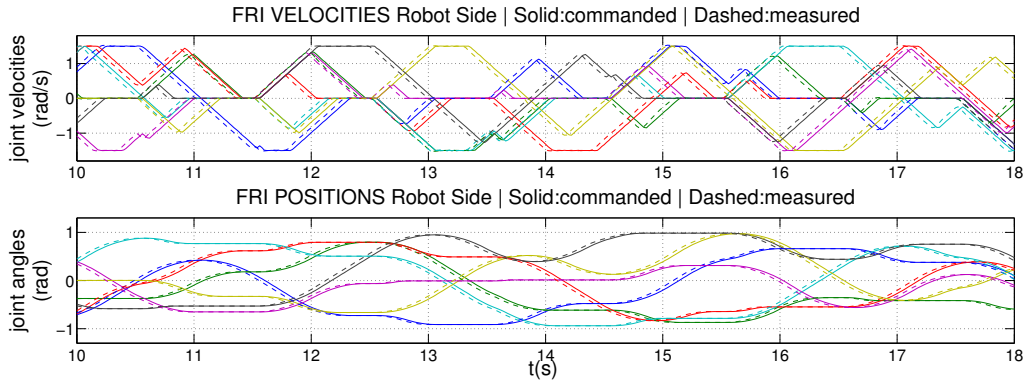
Figure 5 shows a single maneuver performed by the robot using time optimal trajectories for all joints (Fig. 5a) and using synchronized trajectories (Fig. 5b). Notice that when we use the synchronization procedure, the acceleration periods of each joint (except for the longest time-optimal trajectory) are reduced with respect to the set of original time-optimal profiles, while the overall maneuver duration remains the same.

The most interesting feature of this work is the ability to interrupt trajectories depending on unforeseen events (human or sensor driven). In Fig. 6, two excerpts of the profiles generated during longer runs of the robot are shown. Fig. 6a shows the joint velocity profiles when the time-optimal trajectories are used in the robot, while Fig. 6b gives an example of the velocity profiles when the joint trajectories are synchronized. The position profiles are smooth during the whole sequence of maneuvers even when they are interrupted by new events. This shows the advantage of considering the current state in the updating of the trajectories, which is one of the main goals of this work.

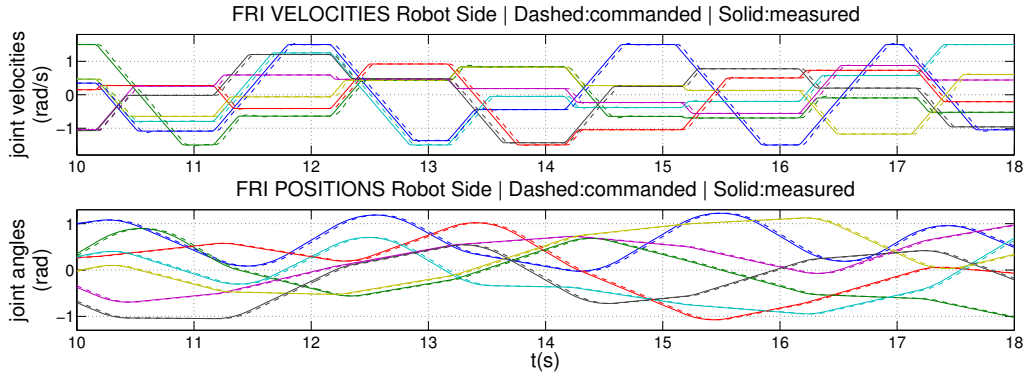
Maneuvers are clearly more abrupt when they are not synchronized, and there are also more rest periods for the joints, while the synchronized motions are smoother but there are no rests in the whole sequence of maneuvers. Velocities are also more steady in the synchronized maneuvers, while they are almost constantly changing in the time-optimal case, showing that the use of acceleration (and hence, motor torques) is reduced with synchronization.

VI. CONCLUSIONS

In this paper, a trajectory generator for robotic arms that takes into account the kinematics of the arm has been studied. The trajectory profiles have been calculated in a time-optimal manner, obtaining a compact, computationally light formulation which works for any set of initial and final states within the robot limits. Due to this, the trajectory generator can be event-driven and dynamically update the current trajectory of each joint whenever a new desired final state arrives. This is a convenient feature of a trajectory



(a) Non-synchronized maneuvers.



(b) Synchronized maneuvers.

Fig. 6: Continuous motion of each joint of the KUKA LWR. New targets state every second.

generator, as there are a number of applications in which we would like to modify the original trajectory based on sensor information, especially for service robotics, where the robot must dynamically react to changes in the environment.

The trajectory generator has been implemented to work with some of the most used software packages in the robotics community (ROS, OROCOS), and is available under LGPL in a public git repository at `https://github.com/IDSCETHZurich/trajectory-generator`. Even if no physical robot is available, the trajectory generator operation and performance can be checked in simulation following the instructions found in the code repository.

The same approach taken in this paper will be extended in future works to more general trajectories that include synchronization with non-zero final velocities, and higher order control inputs (jerk, snap).

VII. ACKNOWLEDGMENTS

This research was funded by the European Union Seventh Framework Programme FP7/2007-2013 under grant agreement no. 248942 RoboEarth.

REFERENCES

[1] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics. Modelling, Planning and Control*. Springer, 2009.

[2] K. G. Shin and N. D. McKay, "Minimum-time control of robotic manipulators with geometric path constraints," *Automatic Control, IEEE Transactions on*, vol. 30, no. 6, pp. 531 – 541, jun 1985.

[3] H. Geering, L. Guzzella, S. Hepner, and C. Onder, "Time-optimal motions of robots in assembly tasks," *Automatic Control, IEEE Transactions on*, vol. 31, no. 6, pp. 512 – 518, jun 1986.

[4] J. Kim, S.-R. Kim, S.-J. Kim, and D.-H. Kim, "A practical approach for minimum-time trajectory planning for industrial robots," *Industrial Robot: An International Journal*, vol. 37, no. 1, pp. 51–61, 2010.

[5] J. Lloyd and V. Hayward, "Trajectory generation for sensor-driven and time-varying tasks," *The International Journal of Robotics Research*, vol. 12, no. 4, pp. 380–393, 1993.

[6] T. Kröger, A. Tomiczek, and F. M. Wahl, "Towards on-line trajectory computation," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, China, oct 2006, pp. 736–741.

[7] T. Kröger and F. M. Wahl, "On-line trajectory generation: Basic concepts for instantaneous reactions to unforeseen events," *IEEE Trans. on Robotics*, vol. 26, no. 1, pp. 94–111, feb 2010.

[8] M. Shimizu, H. Kakuya, W. . Yoon, K. Kitagaki, and K. Kosuge, "Analytical inverse kinematic computation for 7-dof redundant manipulators with joint limits and its application to redundancy resolution," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1131–1142, 2008.

[9] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Athena Scientific, 2007.

[10] H. Maurer, "Optimal-control problems with bounded state variables and control appearing linearly," *SIAM Journal on Control and Optimization*, vol. 15, no. 3, pp. 345–362, 1977.

[11] Willow Garage, "Robot Operating System (ROS)," 2009, [Last visited 2011].

[12] H. Bruyninckx, "Open Robot COntrol Software," 2001, [Last visited 2011].