Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 50-1 (2017) 4929–4934

Near Time-Optimal Path Tracking Method for Waiter Motion Problem

Gábor Csorvási [*] Ákos Nagy [**] István Vajk [***]

[*] Budapest University of Technology and Economics, Department of Automation and Applied Informatics, Budapest, Hungary (e-mail: gabor.csorvasi@aut.bme.hu).
[**] Budapest University of Technology and Economics, Department of Automation and Applied Informatics, Budapest, Hungary (e-mail: akos.nagy@aut.bme.hu).
[***] Budapest University of Technology and Economics Department of Automation and Applied Informatics, MTA-BME Control Engineering Research Group, Budapest, Hungary (e-mail: istvan.vajk@aut.bme.hu).

Abstract: Time-optimal motion planning is one of the most essential problems in robotic theories. The presented direct transcription approach uses convex optimisation method to obtain a minimum-time control for robot manipulators. One subset of motion planning is the waiter motion problem that consists of moving a tablet from one place to another as fast as possible, avoiding the slip of the object that is placed upon it. In this paper the direct transcription method is presented in order to solve the waiter problem. This path tracking problem is also approximated by sequential two dimensional second-order cone programs (SOCP) using nonlinear change of variables. The computational benefit of the sequential method compared to the direct transcription approach is presented. The waiter motion problem is validated by real-life experimental results with a 6-DoF robotic arm.

© 2017, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.

Keywords: Minimum-time control, time-optimal control, convex optimisation, robot control, motion planning, second-order cone programming

1. INTRODUCTION

Minimum-time motion planning is one of the most fundamental areas of robotic researches. Time-optimal methods can not only reduce energy consumption but also increase productivity. The motion planning problem can be solved by two different ways, using either a direct approach or a two stage method.

Direct approaches (Choset et al., 2005) or one step methods solve the entire problem in one step, however, generally this is a highly complex task (LaValle, 2006; Siciliano and Khatib, 2008). As an alternative to direct approach, two stage methods also exist for motion planning. These methods are called decoupled approach (LaValle, 2006). First, a geometry path planner determines a collision-free path for the robot. Dynamics of the robot is omitted in the first stage. In the next stage, a velocity profile for a predefined path is generated (path tracking), where all constraints of the robot (including dynamics of the robot) are applied for the fixed path. Decoupled approach is preferred to direct approach for its lower computational time.

Let us focus on the second stage of the decoupled approach, the velocity profile generation. In this stage the desired path of the robot is already defined, and fixed during path tracking. Therefore a scalar path coordinate ($\theta(t)$) can be used to represent robot position on the path (Shin and Mckay, 1985; Verscheure et al., 2009; Lipp and Boyd, 2014; Hauser, 2013). The major advantage of the scalar path coordinate is that the high dimensional state-space model of the robotic system can be reduced.

Generally there are three typical approaches to solve the decoupled minimum-time control problem. Dynamic programming methods divide the state space into a discrete grid, and the optimal solution is found in this plane (Oberherber et al., 2015). The next approach, so called indirect methods are based on the Pontryagin Maximum Principle: the optimal bang-bang control can be found by determining switching points, where the active constraints can be changed (Bobrow et al., 1985; Flores and Kecskeméthy, 2013). These points can be determined by forward and backward numerical integration. Direct transcription is the third velocity profile generation approach, where the optimal control problem is converted to a convex optimisation problem (Verscheure et al., 2009; Ardeshiri et al., 2011; Max and Lantos, 2014), namely a Second-Order Cone Program (SOCP) problem. Existing efficient convex optimisation solvers can be used in these methods in order to get minimum-time solution.

In this paper a direct transcription formulation for the time-optimal waiter problem is presented. In the waiter motion problem a tablet in the gripper of the robotic arm is moved along a predefined path. We have to ensure that objects placed on the tablet do not slide during the movement. Furthermore a reasonable approximation method for the convex, SOCP problem is presented in the paper. The computational time of the introduced method is lower compared to the original SOCP case.

The paper is organized as follows. In Section 2 the general time-optimal control problem is presented using SOCP formulation. The SOCP formulation is based on the work of Verscheure et al. (2009); Lipp and Boyd (2014). In Section 3 the waiter motion problem is introduced, and is shown how the SOCP approach can be extended with sliding constraints for waiter task. A sequential algorithm is presented in Section 4. Runtime of our method is compared with a state-of-art SOCP solver. Real-life experimental results are presented in Section 5 with a 6-DoF robot manipulator to validate our approach.

2. GENERAL SOCP CONTROL PROBLEM

In this section, we present the time-optimal SOCP control problem based on the work of Verscheure et al. (2009); Lipp and Boyd (2014).

In the approach $\boldsymbol{q}(t)$ is a p-dimensional generalized coordinate vector representing the configuration of the robot, and $\boldsymbol{r}(t) \in \mathrm{SE}(3)$ is the generalized position vector of the end-effector in the operational space. A fixed path traversed by the robot is given by one of these forms. The conversion between the joint space and the operational space coordinates can be obtained by the following formula

$$\boldsymbol{q}(t) \underset{\varphi^{-1}}{\overset{\varphi}{\rightleftharpoons}} \boldsymbol{r}(t), \tag{1}$$

where $\varphi(\boldsymbol{q})$ and $\varphi^{-1}(\boldsymbol{r})$ are direct and inverse kinematics of the robot respectively.

Using chain rule, path, velocities, and accelerations of the p-DoF robotic manipulator are defined in the following way:

$$\boldsymbol{q}(t) = \boldsymbol{s}(\theta(t)) \tag{2}$$

$$\dot{\boldsymbol{q}}(t) = \boldsymbol{s}'(\theta(t))\dot{\theta}(t) \tag{3}$$

$$\ddot{\boldsymbol{q}}(t) = \boldsymbol{s}'(\theta(t))\ddot{\theta}(t) + \boldsymbol{s}''(\theta(t))\dot{\theta}(t)^2 \qquad t \in [0,T], \tag{4}$$

where $\boldsymbol{s}(\theta) : [0,1] \to \mathbb{R}^p$ is the fixed path, $\theta(t) : [0,T] \to [0,1]$ is the scalar path coordinate, $\dot{(\cdot)}$ indicates time derivative, $\boldsymbol{s}'(\theta) = \frac{d\boldsymbol{s}(\theta)}{d\theta}$, $\boldsymbol{s}''(\theta) = \frac{d^2\boldsymbol{s}(\theta)}{d\theta^2}$, and T is the time when the robot reaches the end of the path. These equations are analogous for the operational space coordinates as well:

$$\boldsymbol{r}(t) = \boldsymbol{p}(\theta(t)) \tag{5}$$

$$\dot{\boldsymbol{r}}(t) = \boldsymbol{p}'(\theta(t))\dot{\theta}(t) \tag{6}$$

$$\ddot{\boldsymbol{r}}(t) = \boldsymbol{p}'(\theta(t))\ddot{\theta}(t) + \boldsymbol{p}''(\theta(t))\dot{\theta}(t)^2 \tag{7}$$

Let us suppose that $\dot{\theta}(t) \geq 0$ holds everywhere, being a general assumption in time-optimal control.

The optimisation variables will be $\dot{\theta}(t)$ and $\ddot{\theta}(t)$. In order to represent these, two new variables are introduced (from now on time dependency is omitted):

$$a(\theta) := \ddot{\theta} \tag{8}$$

$$b(\theta) := \dot{\theta}^2 \tag{9}$$

To form a finite dimensional problem, the scalar path coordinate is discretised in $n+1$ points. The i-th point is indicated by θ_i. Discretisation is based on the assumption that $a(\theta)$ is constant between two discretisation points, therefore $b'(\theta)$ is also constant, and $b(\theta)$ is linear in θ:

$$\begin{aligned} &b(\theta) = b(\theta_{i-1}) + (\theta - \theta_{i-1})\left(\frac{b_i - b_{i-1}}{\theta_i - \theta_{i-1}}\right) \\ &\theta \in [\theta_{i-1}, \theta_i] \end{aligned} \tag{10}$$

Since $a(\theta)$ is discontinuous at θ_i points, we evaluate $a(\theta)$ at the midpoint of the interval. The following notations are introduced for the variables:

$$\begin{aligned} b_i &:= b(\theta_i) && i = 0,\ldots,n \\ a_i &:= a(\bar{\theta}_i) && i = 1,\ldots,n \\ \bar{\theta}_i &= \frac{(\theta_i + \theta_{i-1})}{2} && i = 1,\ldots,n \end{aligned} \tag{11}$$

Based on the discretisation assumption, the relation between a_i and b_i can be defined in the following manner:

$$\begin{aligned} &b_i - b_{i-1} = 2a_i d\theta_i \\ &d\theta_i = \theta_i - \theta_{i-1} \qquad i = 1,\ldots,n \end{aligned} \tag{12}$$

The motion equation of a p-DoF robotic arm is defined by using second-order Lagrange equation (Verscheure et al., 2009). Since $a(\theta)$ is discontinuous in θ_i, the dynamics is also evaluated at the midpoints:

$$\begin{aligned} &\boldsymbol{\tau}_i = \boldsymbol{m}(\bar{\theta}_i)a_i + \boldsymbol{c}(\bar{\theta}_i)\frac{b_{i-1}+b_i}{2} + \boldsymbol{d}(\bar{\theta}_i) \\ &\boldsymbol{m}(\bar{\theta}_i) := \mathbf{M}(\boldsymbol{s}(\bar{\theta}_i))\boldsymbol{s}'(\bar{\theta}_i) \\ &\boldsymbol{c}(\bar{\theta}_i) := \mathbf{M}(\boldsymbol{s}(\bar{\theta}_i))\boldsymbol{s}''(\bar{\theta}_i) + \mathbf{C}(\boldsymbol{s}(\bar{\theta}_i), \boldsymbol{s}'(\bar{\theta}_i))\boldsymbol{s}'(\bar{\theta}_i) \\ &\boldsymbol{d}(\bar{\theta}_i) := \boldsymbol{d}(\boldsymbol{s}(\bar{\theta}_i)) \qquad i = 1,\ldots,n, \end{aligned} \tag{13}$$

where $\boldsymbol{\tau}_i \in \mathbb{R}^p$ is the generalized force vector, $\mathbf{M}(\boldsymbol{q}) : \mathbb{R}^p \to \mathbb{S}^p_{++}$ is the mass matrix, where $\mathbb{S}^p_{++}$ is the set of symmetric positive definite matrices, $\mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) : \mathbb{R}^{2p} \to \mathbb{R}^{p\times p}$ is a matrix accounting for centrifugal, Coriolis effects (it is linear in $\dot{\boldsymbol{q}}$), and $\boldsymbol{d}(\boldsymbol{q}) : \mathbb{R}^p \to \mathbb{R}^p$ is accounting for joint position dependent forces including gravity.

The derivatives of the path should be specified for (13). When $\boldsymbol{s}(\theta)$ is differentiable, $\boldsymbol{s}'(\theta)$ and $\boldsymbol{s}''(\theta)$ can be calculated analytically. In our approach we assume that $\boldsymbol{s}(\theta)$ is only given at θ_i points, so an approximation method is needed for the derivatives (e.g.: a sixth-order Runge–Kutta approximation) (Lipp and Boyd, 2014).

The objective function of the optimisation problem is T, which can be expressed by variable $b(\theta)$:

$$T = \int_0^T 1dt = \int_{\theta(0)}^{\theta(T)} \dot{\theta}^{-1} d\theta = \int_0^1 b(\theta)^{-1/2} d\theta \tag{14}$$

In discrete time T can be determined by the introduced variable b_i (Lipp and Boyd, 2014):

$$T = 2\sum_{i=1}^{n}\left(\frac{d\theta_i}{b_i^{1/2} + b_{i-1}^{1/2}}\right) \tag{15}$$

It is important to note, that (15) is a convex function of b_i. Based on (12, 13, 15) the following convex optimisation problem can be created (Verscheure et al., 2009; Lipp and Boyd, 2014):

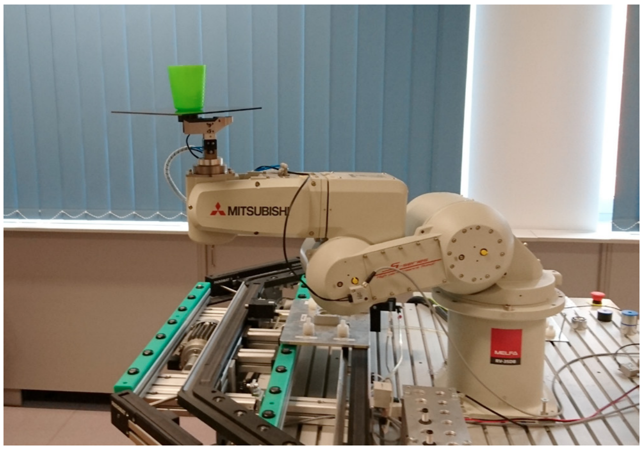

Fig. 1. 6-DoF Mitsubishi robot manipulator with a glass on a tablet.

$$
\begin{aligned}
\min_{a_i, b_i, \boldsymbol{\tau}_i} \quad & 2\sum_{i=1}^{n}\left(\frac{d\theta_i}{b_i^{1/2}+b_{i-1}^{1/2}}\right) \\
\text{subject to} \quad & \boldsymbol{\tau}_i = \boldsymbol{m}(\bar{\theta}_i)a_i + \boldsymbol{c}(\bar{\theta}_i)\frac{b_{i-1}+b_i}{2} + \boldsymbol{d}(\bar{\theta}_i) \\
& b_i - b_{i-1} = 2a_i d\theta_i \\
& (b_0, b_i, a_i, \boldsymbol{\tau}_i) \in \mathcal{C} \qquad i = 1, \ldots, n,
\end{aligned} \tag{16}
$$

where $\mathcal{C}$ is a second-order cone constraint, which has the following general form (Boyd and Vandenberghe, 2004):

$$|\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}|_2 \leq \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{d} \tag{17}$$

Problem (16) is a finite dimensional, convex problem, namely a Second-Order Cone Programming (SOCP) formula. SOCP problems can be efficiently solved by modern optimisation tools using interior-point methods.

3. WAITER PROBLEM FORMULATION

As a practical example for the path tracking problem let us consider a waiter motion which consists of moving a tablet carrying one or more glasses along a prescribed spatial path as fast as possible so that the objects placed on it do not slide. An example model for the waiter problem is shown in Fig. 1. This problem is introduced in the work of Flores and Kecskeméthy (2013), where an indirect approach is presented, that is based on a bang-bang control. The optimal trajectory is constructed by integrating the system equations forward and backward in time from switching points. The limitation of this method is accurately determining these switching points which can be hard if the path and its derivatives are not given by splines. In this section a different approach is introduced where the constraints are defined so that the problem can be written as an SOCP control problem introduced in Section 2. Similar result to the presented work from a different approach can be found in Duijkeren et al. (2015).

To solve the so called "Waiter Motion Problem" two sets of constraints need to be defined. The first set contains the limits of the joint velocities and accelerations:

$$\dot{\boldsymbol{q}}_{min} \leq \dot{\boldsymbol{q}}(t) \leq \dot{\boldsymbol{q}}_{max} \tag{18}$$
$$\ddot{\boldsymbol{q}}_{min} \leq \ddot{\boldsymbol{q}}(t) \leq \ddot{\boldsymbol{q}}_{max}, \tag{19}$$

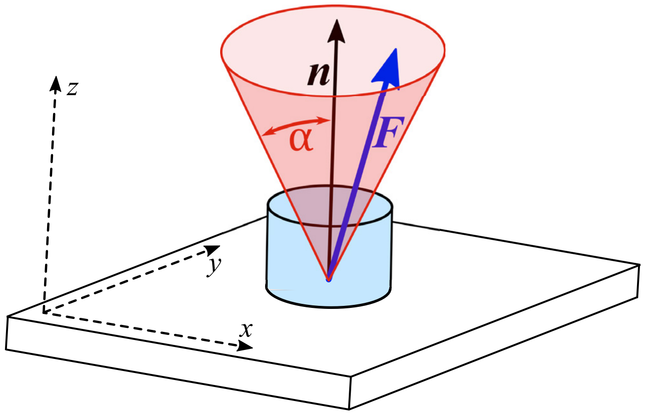

Fig. 2. Example of a friction constraint. The F force applied to the object placed on the tablet.

where we assume that the velocity constraint is symmetric, e.g. $\dot{\boldsymbol{q}}_{min} = -\dot{\boldsymbol{q}}_{max}$. These limits are typically provided by the manufacturer of the robotic arm.

After discretising the previous equations, and using (8) and (9) the above mentioned constraints can be written in the following form:

$$\boldsymbol{s}'^2(\theta_i)b_i \leq \dot{\boldsymbol{q}}_{max}^2 \qquad i = 0, \ldots, n \tag{20}$$

$$\ddot{\boldsymbol{q}}_{min} \leq \boldsymbol{s}'(\bar{\theta}_i)a_i + \boldsymbol{s}''(\bar{\theta}_i)(b_i + b_{i-1})/2 \leq \ddot{\boldsymbol{q}}_{max} \qquad i = 1, \ldots, n \tag{21}$$

The second set of constraints is given by the no sliding condition of the object. Since the goal is that, the object does not move on the tablet, a static friction model is suitable:

$$\sqrt{F_x^2 + F_y^2} \leq \mu_0 F_z \quad \text{with} \quad \boldsymbol{F} = m(\ddot{\bar{\boldsymbol{r}}}(t) - \boldsymbol{g}), \tag{22}$$

where $\boldsymbol{F}$ is the force applied to the object excluding the gravity. F_x, F_y, and F_z are the components of F in the local coordinate frame of the tablet, m is the object mass, $\boldsymbol{g}$ is the gravitational acceleration vector and μ_0 is the dry friction coefficient between the contact surfaces. The $\ddot{\bar{\boldsymbol{r}}}$ marks the translational part of object acceleration, which is the first three co-ordinate of the general acceleration $\ddot{\boldsymbol{r}}$.

An illustration of the applied forces can be seen in Fig. 2. This condition is clearly neither linear nor convex in this form. In order to transform it to a convex form, express the left side from the definition of the Euclidean norm then rearrange:

$$|\boldsymbol{F}|_2 \leq \sqrt{\mu_0^2 + 1} \cdot F_z \tag{23}$$

As a next step the following statements are introduced:

$$\mu_0 = \tan\alpha \tag{24}$$
$$F_z = \boldsymbol{F} \cdot \boldsymbol{n}, \tag{25}$$

where α is the friction angle, $\boldsymbol{n}$ is the normal vector of the tablet plane. Substituting the above mentioned equations into (23) we get the convex form of the no sliding condition:

$$|\boldsymbol{F}|_2 \leq \frac{\boldsymbol{F} \cdot \boldsymbol{n}}{\cos\alpha} \tag{26}$$

Using the right hand side of (22) we get:

$$|m(\ddot{\bar{\boldsymbol{r}}}(t) - \boldsymbol{g})|_2 \leq \frac{m(\ddot{\bar{\boldsymbol{r}}}(t) - \boldsymbol{g}) \cdot \boldsymbol{n}}{\cos\alpha} \tag{27}$$

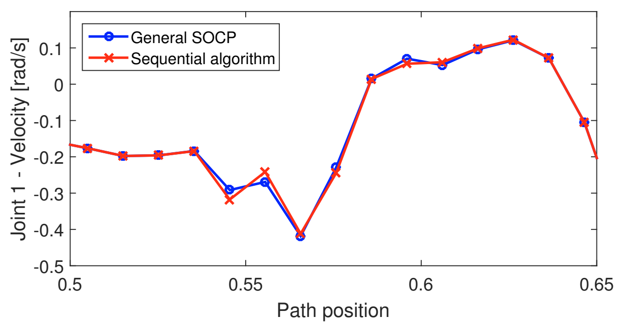

Fig. 3. Part of the resulted velocities with the two approaches.

The scalar m mass can be taken out from the norm in order to simplify it. Also the formula (7), the second derivative of the operational coordinates can be inserted:

$$\begin{aligned}&|\bar{\boldsymbol{p}}'(\bar{\theta}_i)a_i + \bar{\boldsymbol{p}}''(\bar{\theta}_i)(b_i + b_{i-1})/2 - \boldsymbol{g}|_2 \leq \\ &\quad ((\bar{\boldsymbol{p}}'(\bar{\theta}_i)a_i + \bar{\boldsymbol{p}}''(\bar{\theta}_i)(b_i + b_{i-1})/2 - \boldsymbol{g}) \cdot \boldsymbol{n})/\cos\alpha \\ &\quad i = 1, \ldots, n \end{aligned} \tag{28}$$

Equations (20), (21), and (28) form a system of constraints that can be used in problem (16) in order to track a predefined path without the slip of the object.

The complete problem can be written in the following mathematical form:

$$\begin{aligned}\min_{a_i, b_i, \boldsymbol{\tau}_i} \quad & 2\sum_{i=1}^{n}\left(\frac{d\theta_i}{b_i^{1/2} + b_{i-1}^{1/2}}\right) \\ \text{subj. to} \quad & \boldsymbol{\tau}_i = \boldsymbol{m}(\bar{\theta}_i)a_i + \boldsymbol{c}(\bar{\theta}_i)\frac{b_{i-1} + b_i}{2} + \boldsymbol{d}(\bar{\theta}_i) \\ & b_i - b_{i-1} = 2a_i d\theta_i \\ & \boldsymbol{s}'^2(\theta_i)b_i \leq \dot{\boldsymbol{q}}_{max}^2 \\ & \ddot{\boldsymbol{q}}_{min} \leq \boldsymbol{s}'(\bar{\theta}_i)a_i + \boldsymbol{s}''(\bar{\theta}_i)(b_i + b_{i-1})/2 \\ & \boldsymbol{s}'(\bar{\theta}_i)a_i + \boldsymbol{s}''(\bar{\theta}_i)(b_i + b_{i-1})/2 \leq \ddot{\boldsymbol{q}}_{max} \\ & |\bar{\boldsymbol{p}}'(\bar{\theta}_i)a_i + \bar{\boldsymbol{p}}''(\bar{\theta}_i)(b_i + b_{i-1})/2 - \boldsymbol{g}|_2 \leq \\ & \quad (\bar{\boldsymbol{p}}'(\bar{\theta}_i)a_i + \bar{\boldsymbol{p}}''(\bar{\theta}_i)(b_i + b_{i-1})/2 - \boldsymbol{g}) \cdot \frac{\boldsymbol{n}}{\cos\alpha} \\ & \qquad i = 1, \ldots, n \end{aligned} \tag{29}$$

Equation (29) is the complete form of an SOCP formula which solves the Waiter Motion Problem.

4. SEQUENTIAL ALGORITHM

Although the Waiter Motion Problem can be solved with the presented method, the calculations can take a large amount of time for a path with a relative high dimension (e.g. $n = 10^4$). In this section a sequential algorithm is introduced that provides a solution which is feasible (comply with all the constraints), is close to the optimal one, and can be calculated faster than the general SOCP optimisation problem.

In order to formulate the sequential algorithm some slight modifications are needed. First, using equation (12) the a_i variables can be eliminated from the constraints by:

$$a_i = \frac{b_i - b_{i-1}}{2d\theta_i} \qquad i = 1, \ldots, n \tag{30}$$

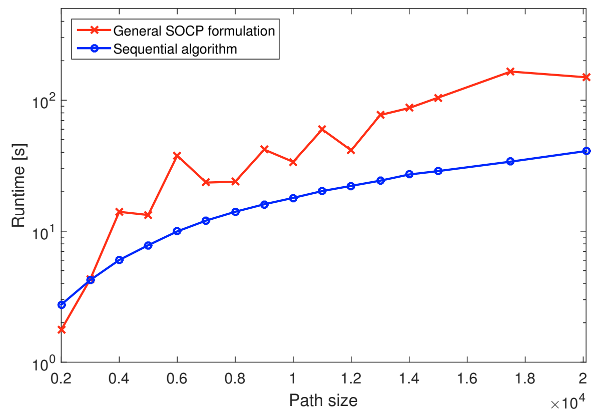

Fig. 4. Runtimes of the general SOCP and the sequential algorithms.

The limits of the joint velocities are not dependent on a_i thus equation (20) remains unchanged. The limits of the joint accelerations become:

$$\boldsymbol{e}_i := \frac{\boldsymbol{s}''(\bar{\theta}_i)}{2} + \frac{\boldsymbol{s}'(\bar{\theta}_i)}{2d\theta_i} \tag{31}$$

$$\boldsymbol{f}_i := \frac{\boldsymbol{s}''(\bar{\theta}_i)}{2} - \frac{\boldsymbol{s}'(\bar{\theta}_i)}{2d\theta_i} \tag{32}$$

$$\ddot{\boldsymbol{q}}_{min} \leq \boldsymbol{e}_i b_i + \boldsymbol{f}_i b_{i-1} \leq \ddot{\boldsymbol{q}}_{max} \qquad i = 1, \ldots, n \tag{33}$$

This modification can be done for friction constraints as well:

$$\boldsymbol{u}_i := \frac{\bar{\boldsymbol{p}}''(\bar{\theta}_i)}{2} + \frac{\bar{\boldsymbol{p}}'(\bar{\theta}_i)}{2d\theta_i} \tag{34}$$

$$\boldsymbol{v}_i := \frac{\bar{\boldsymbol{p}}''(\bar{\theta}_i)}{2} - \frac{\bar{\boldsymbol{p}}'(\bar{\theta}_i)}{2d\theta_i} \tag{35}$$

$$\begin{aligned}&|\boldsymbol{u}_i b_i + \boldsymbol{v}_i b_{i-1} - \boldsymbol{g}|_2 \leq ((\boldsymbol{u}_i b_i + \boldsymbol{v}_i b_{i-1} - \boldsymbol{g}) \cdot \boldsymbol{n})/\cos\alpha \\ &\quad i = 1, \ldots, n\end{aligned} \tag{36}$$

As seen above, these equations now only depend on b_i and b_{i-1}.

With the modified constraints (33), and (36), a two dimensional SOCP problem can formulated for a fix i:

$$\begin{aligned}\min_{x,y} \quad & \frac{1}{\sqrt{y} + \sqrt{x}} \\ \text{subject to} \quad & 0 \leq x \leq \bar{x} \\ & 0 \leq y \leq \bar{y} \\ & \ddot{\boldsymbol{q}}_{min} \leq \boldsymbol{e}_i y + \boldsymbol{f}_i x \leq \ddot{\boldsymbol{q}}_{max} \\ & |\boldsymbol{u}_i y + \boldsymbol{v}_i x - \boldsymbol{g}|_2 \\ & \quad \leq ((\boldsymbol{u}_i y + \boldsymbol{v}_i x - \boldsymbol{g}) \cdot \boldsymbol{n})/\cos\alpha,\end{aligned} \tag{37}$$

where $x := b_{i-1}$, $y := b_i$, and $\bar{x}$, $\bar{y}$ are upper bounds on x and y respectively. Let us assume that we have an initial guess for x from a previous calculation, and for y from (20) then $\bar{x}$ and $\bar{y}$ can be set accordingly. The optimal y value can be the new guess for b_i. In case the optimal value of x is smaller than the previously calculated value, the calculations have to be done backward as well.

Based on these results the following algorithm can be constructed:

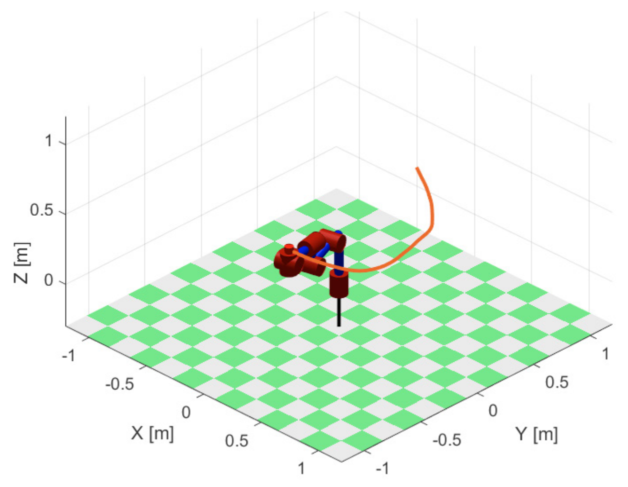

Fig. 5. The path of the endpoint and the robot model in the Robotics Toolbox.

Step 0 Initialize b_i from (20): $b_i^{(0)} := \dot{\boldsymbol{q}}^2_{max}/\boldsymbol{s}'^2(\theta_i)$ for $i = 0, \dots, n$.

Step 1 Let $b_0^{(1)} = b_0^{(0)}$, and for $i = 1, \dots, n$ solve (37) with $\bar{x} := b_{i-1}^{(1)}$ and $\bar{y} := b_i^{(0)}$. At every step, the new optimal value is stored by $b_i^{(1)} := y$.

Step 2 Let $b_n^{(2)} = b_n^{(1)}$, and for $i = n, \dots, 1$ solve (37) with $\bar{x} := b_{i-1}^{(1)}$ and $\bar{y} := b_i^{(2)}$. At every step, the optimal value is stored by $b_{i-1}^{(2)} := x$

At last the $b_i^{(2)}$ values are the final solution for the Waiter Motion Problem. In case the a_i values are also needed, they can be calculated by (30).

However the result (b_i) can differ from the optimal solution at some values of i, our investigations show that this difference is small, furthermore all the constraints are met thus the result is feasible. Our investigations show that this difference is small. An example for this difference is shown in Fig. 3. The relative error in T is at most 0.1% for the path presented in Section 5.

To compare the presented algorithms, both of them are implemented in C++ using the SOCP solver of Gurobi, Gurobi Optimization, Inc. (2015), and tested on a desktop computer Intel i5-6500 @ 3.2Mhz having 16GB RAM.

Comparison of the runtimes is shown in Fig. 4. The complete SOCP formulation solves a large n-dimensional problem however the presented sequential algorithm solves only, a small, two-dimensional problem $2n$ times which is faster for large n, therefore the sequential approach is more time-efficient. Fluctuation of the SOCP runtime is due to the presolve feature of Gurobi and the different convergence rate of the interior-point method used in the solver. It can be seen that for small path size our approach is slower or equally slow compared to the SOCP formulation. This is because of the overhead of the sequential algorithm. For larger path size the overhead becomes small compared to the calculation time. It can be shown that the introduced algorithm is also linear in path size.

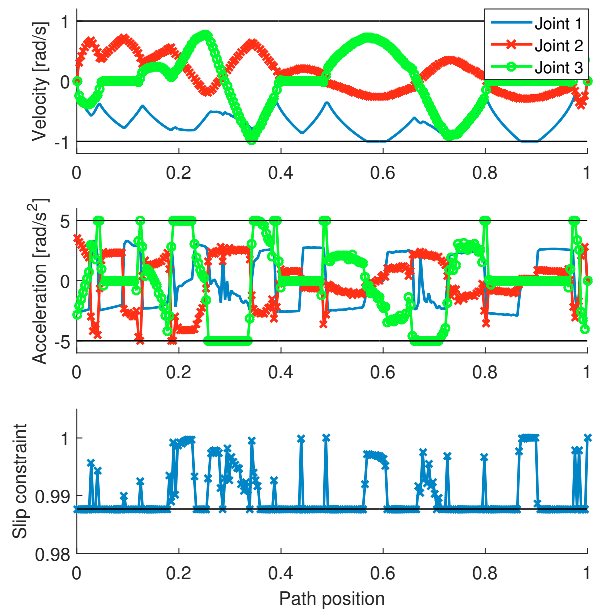

Fig. 6. The generated profiles for the waiter demonstration. The constraints are also visible in the figure. The slip constraint is equal to $\frac{\boldsymbol{F}\cdot\boldsymbol{n}}{|\boldsymbol{F}|_2}$ (26).

5. EXPERIMENTAL RESULTS

To validate the presented algorithms for the waiter motion problem, an experimental real-life demonstration is created using a 6-DoF robot manipulator. A Mitsubishi RV-3SDB industrial robot is used (see Figure 1), which is a well-known industrial manipulator.

The predefined, fixed path for the near-time optimal method was generated in the following way. The path of the end-effector was manually defined in operational space, and using inverse kinematics the joint space path was calculated. For inverse kinematics the Robotics Toolbox was used in Matlab (Corke, 2011). The algorithm of the inverse kinematics is based on the Denavit-Hartenberg parameters of the robot. The predefined path can be seen in Figure 5.

For the time-optimal problem formulation it was also necessary to determine the friction angle (α) used in (24). We measured α by placing the glass on the table, and increased the angle of the tablet until the glass begun to slide. The actual value of α was 9° in our scenario.

For a test scenario we used velocity, acceleration constraints, and a constraint accounting for sliding. Torque constraints were not used, because the dynamic parameters of the robot were not known at the time. The results were not effected, because these do not appear in the slip constraints. Based on the algorithm presented in Section 4 time parameterized path (trajectory) was generated for the waiter motion problem. The velocity and the acceleration profiles for the first three joints, and the slipping profile can be seen in Figure 6.

To control the robotic manipulator, we used the Real-Time External Control capability (Mitsubishi, 2014) of the

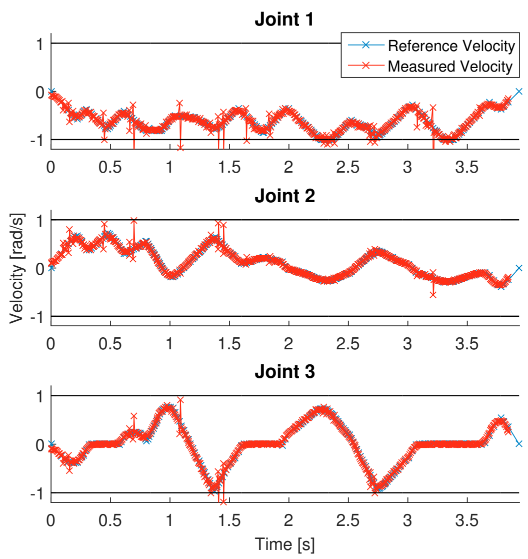

Fig. 7. The difference between the reference and the measured velocity profile for the first three joints.

robot. This control mode allows us to set the desired joint positions in each motor cycle. The desired joint positions are equal to the points of the generated trajectory. We selected the actual position of the path based on the time coordinate of the trajectory. To make the motion smoother, interpolation is used between two path points.

The Real-Time External Control communication protocol in based on TCP/IP, the control cycle is approximately 7.1 ms in case of Mitsubishi RV-3SDB robot (Mitsubishi, 2014). Our complete control software (solver algorithm and the control software for External Control) is implemented in C++.

As we use the predefined path as a control signal, the difference between the predefined and the traversed path only depends on the accuracy of the robotic arm, which is negligibly small (in our case max. ± 0.02 mm in operational space (Mitsubishi, 2009)). On the other hand, error in the velocity and acceleration profiles can be higher due to the approximations used for the first and second derivatives of the path ($\boldsymbol{s}'(\theta_i)$, $\boldsymbol{s}''(\theta_i)$). The measured error in the velocity profiles for the first three joints can be seen in Figure 7.

6. CONCLUSION

In this paper, an SOCP based formulation and a sequential algorithm for the waiter motion problem have been presented based on non-linear change of variables. The main benefit of the solver is the lower computational time. The runtime of the presented sequential solver was compared to a state of art optimisation tool. The use of the presented approach for the waiter motion problem was shown in a real life scenario with a 6-DoF robot manipulator as well.

ACKNOWLEDGEMENTS

This work was partially supported by the MTA-BME Control Engineering Research Group of the HAS at the Budapest University of Technology and Economics.

REFERENCES

Ardeshiri, T., Norrlöf, M., Löfberg, J., and Hansson, A. (2011). Convex Optimization approach for Time-Optimal Path Tracking of Robots with Speed Dependent Constraints. In *Proceedings of 18th IFAC World Congress*, 14648–14653. Milano, Italy.

Bobrow, J.E., Dubowsky, S., and Gibson, J.S. (1985). Time-optimal Control of Robotic Manipulators along Specified Paths. *International Journal of Robotics Research*, 4(3), 3–17.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G.A., Burgard, W., Kavraki, L.E., and Thrun, S. (2005). *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Cambridge, MA, USA.

Corke, P.I. (2011). *Robotics, Vision and Control: Fundamental Algorithms in Matlab*. Springer.

Duijkeren, N.V., Debrouwere, F., Pipeleers, G., and Swevers, J. (2015). Cartesian constrained time-optimal point-to-point motion planning for robots : the waiter problem. In *Benelux Meeting on Systems and Control, Lommel, Belgium*.

Flores, F.G. and Kecskeméthy, A. (2013). Time-Optimal Path Planning Along Specified Trajectories. In H. Gattringer and J. Gerstmayr (eds.), *Multibody System Dynamics, Robotics and Control*, 1–16. Springer Vienna.

Gurobi Optimization, Inc. (2015). Gurobi Optimizer Version 6.5.1. [Computer software]. URL `http://www.gurobi.com`.

Hauser, K. (2013). Fast Interpolation and Time-Optimization on Implicit Contact Submanifolds. In *Proceedings of Robotics: Science and Systems*. Berlin, Germany.

LaValle, S.M. (2006). *Planning Algorithms*. Cambridge University Press, New York, NY, USA.

Lipp, T. and Boyd, S. (2014). Minimum-time speed optimisation over a fixed path. *International Journal of Control*, 87(6), 1297–1311.

Max, G. and Lantos, B. (2014). Time optimal control of four-in-wheel-motors driven electric cars. *Periodica Polytechnica Electrical Engineering and Computer Science*, 58(4), 149–159.

Mitsubishi (2009). *RV-3SD/3SDJ/3SDB/3SDBJ Series: Standard Specifications Manual*.

Mitsubishi (2014). *CR750/CR751 Series Controller Instruction Manual*.

Oberherber, M., Gattringer, H., and Müller, A. (2015). Successive dynamic programming and subsequent spline optimization for smooth time optimal robot path tracking. *Mechanical Sciences*, 6(2), 245–254.

Shin, K. and Mckay, N.D. (1985). Minimum-time control of robotic manipulators with geometric path constraints. *IEEE Transactions on Automatic Control*, 30(6), 531–541.

Siciliano, B. and Khatib, O. (eds.) (2008). *Springer Handbook of Robotics*. Springer, Berlin, Heidelberg, Germany.

Verscheure, D., Demeulenaere, B., Swevers, J., Schutter, J.D., and Diehl, M. (2009). Time-Optimal Path Tracking for Robots: A Convex Optimization Approach. *IEEE Transactions on Automatic Control*, 54(10), 2318–2327.