An algorithm for on-line inverse kinematics with path tracking capability under velocity and acceleration constraints

Gianluca Antonelli Stefano Chiaverini Giuseppe Fusco

Dipartimento di Automazione, Elettromagnetismo,
Ingegneria dell'Informazione e Matematica Industriale
Università degli Studi di Cassino
via G. Di Biasio 43, 03043 Cassino (FR), Italy
{antonelli,chiaverini,fusco}@unicas.it

Abstract

Typical tasks for robotic manipulators are specified in terms of an end-effector trajectory. On the other hand, the robot arm is actuated at the joints thus requiring control actions to be performed by the joint servos. Inverse kinematics algorithms map the desired task-space trajectory into a suitable joint-space trajectory. In this paper, a new second-order inverse kinematics algorithm is developed which takes into account joint velocity and acceleration limits while pursuing tracking of the assigned end-effector path. The proposed technique is designed for real-time operation and requires a light computational burden. This goal is achieved by properly modifying the time law with a time warp when joint limits are encountered. Case studies are developed to analyze the performance of the proposed method.

1 Introduction

A manipulation task is usually given in terms of a desired end-effector trajectory. Since the manipulator is controlled by joint servos, a mapping from the task space to the joint space is required. An effective approach to the motion control problem for robotic manipulators is the so-called *kinematic control*. This is based on an inverse kinematic transformation which feeds the reference values corresponding to an assigned end-effector trajectory to the joint servos.

A closed-form analytical solution can be found only for manipulators having simple geometric structures. For all those structures that cannot be solved in closed-form, a number of algorithmic techniques have been devised. These are mainly based on inversion of the mapping established between the joint space and the task space by the manipulator's Jacobian matrix. These techniques suffer from errors due to both long-term numerical integration drift and incorrect initial joint angles. Algorithmic solutions that overcome these problems are based on the use of feedback correction; these are termed closed-loop inverse kinematics (CLIK) algorithms. A CLIK algorithm based on the Jacobian transpose was first independently proposed in [2, 13]. Similar algorithms can be devised that use either the Jacobian pseudoinverse, e.g. [6], or a damped least-squares inverse, e.g. [3, 11].

A common problem of the above inverse kinematics algorithms is that the generated joint motion may cause violation of the speed and/or acceleration limits of the joint actuators. A possible solution would be to include saturations on the joint variables in the algorithms; however, path tracking would no longer be guaranteed. On the other hand, the fulfillment of joint constraints is often approached as an optimization problem for kinematically redundant manipulators, e.g., [1, 4, 9], thus preventing application to nonredundant structures. In the case of off-line inverse kinematics, some solutions have been proposed to handle velocity limits and torque minimization [12] while performing a point-to-point trajectory [10]. However, off-line techniques cannot be applied in the case of real-time end-effector trajectory generation and point-to-point methods do not embed path tracking constraints.

In this paper, a new inverse kinematics algorithm is proposed which takes into account joint velocity and acceleration constraints. Following the work reported in [5], this goal is achieved by suitably slowing down the task-space trajectory when joint limits are encountered. The time law is modified through a time warp such that the introduced virtual time allows fulfillment of the velocity and acceleration constraints. At each step, the time warp is obtained on-line as the result of a quadratic minimization problem [7]. When at some configuration the minimization problem does not have solution, a new *sub-optimal* algorithm is adopted which satisfyies the joint limits by relaxing the path constraints. Application of the proposed method is finally demonstrated in numerical case studies.

2 Second-order inverse kinematics algorithm

It is well known that the direct kinematics of a serial-chain manipulator can be written in the form

$$\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{q}), \tag{1}$$

where $\boldsymbol{x}$ is the vector of the end-effector variables and $\boldsymbol{q}$ is the vector of the joint variables.

By twice differentiating eq. (1) one obtains

$$\ddot{\boldsymbol{x}} = \boldsymbol{J}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}, \tag{2}$$

in which $\boldsymbol{J} = \mathrm{d}\boldsymbol{f}/\mathrm{d}\boldsymbol{q}$ is the manipulator's analytical Jacobian matrix.

The motion of the manipulator is specified by assigning in real-time the end-effector desired trajectory $\boldsymbol{x}_{\mathrm{d}}(t)$; in the case of a second-order inverse kinematics problem the functions

$$\boldsymbol{x}_{\mathrm{d}}(t) \tag{3}$$

$$\dot{\boldsymbol{x}}_{\mathrm{d}}(t) = \frac{\mathrm{d}\boldsymbol{x}_{\mathrm{d}}}{\mathrm{d}t}(t) \tag{4}$$

$$\ddot{\boldsymbol{x}}_{\mathrm{d}}(t) = \frac{\mathrm{d}^2\boldsymbol{x}_{\mathrm{d}}}{\mathrm{d}t^2}(t) \tag{5}$$

are supposed to be available on-line.

Second-order algorithmic solutions to the inverse kinematics problems are based on the inversion of the direct differential kinematics equation (2) and are implemented in discrete-time form. It can be recognized that a discrete-time version of second-order inverse kinematics algorithm can be written in the general form

$$\ddot{\boldsymbol{q}}_k = \boldsymbol{H}_k\boldsymbol{w}_k \tag{6}$$

$$\dot{\boldsymbol{q}}_k = \dot{\boldsymbol{q}}_{k-1} + \ddot{\boldsymbol{q}}_k\Delta t \tag{7}$$

$$\boldsymbol{q}_k = \boldsymbol{q}_{k-1} + \dot{\boldsymbol{q}}_k\Delta t - \ddot{\boldsymbol{q}}_k\Delta t^2/2, \tag{8}$$

where the subscript k denotes quantities evaluated at the k-th time instant t_k, and Δt is the time interval, i.e.

$$t_k = t_{k-1} + \Delta t. \tag{9}$$

Since the joint positions and velocities are obtained by numerical integration of the joint accelerations, a drift of the solution occurs that can be overcome by the use of a feedback correction term. Further, by assuming a square non-singular Jacobian matrix, the following expression for $\boldsymbol{H}_k$ and $\boldsymbol{w}_k$ is chosen

$$\boldsymbol{H}_k = \boldsymbol{J}_{k-1}^{-1} \tag{10}$$

$$\boldsymbol{w}_k = \ddot{\boldsymbol{x}}_{\mathrm{d},k} - \dot{\boldsymbol{J}}_{k-1}\dot{\boldsymbol{q}}_{k-1} + \boldsymbol{K}_p(\boldsymbol{x}_{\mathrm{d},k} - \boldsymbol{x}_{k-1}) + \boldsymbol{K}_v(\dot{\boldsymbol{x}}_{\mathrm{d},k} - \dot{\boldsymbol{x}}_{k-1}), \tag{11}$$

where $\boldsymbol{K}_p$ and $\boldsymbol{K}_v$ are suitable constant positive-definite matrices and $\boldsymbol{J}_k$ is a shorthand notation for $\boldsymbol{J}(\boldsymbol{q}_k)$.

3 Handling of velocity and acceleration constraints

When inverse kinematics algorithms are utilized to generate the joint motion corresponding to an assigned end-effector trajectory, the resulting joint trajectory may exceed the speed limits and/or the maximum acceleration achievable by the joint actuators, expressed by

$$-\dot{q}_{i,\,\mathrm{lim}} \le \dot{q}_{i,k} \le \dot{q}_{i,\,\mathrm{lim}} \tag{12}$$

$$-\ddot{q}_{i,\,\mathrm{lim}} \le \ddot{q}_{i,k} \le \ddot{q}_{i,\,\mathrm{lim}}, \tag{13}$$

where $\dot{q}_{i,\,\mathrm{lim}}$ is the speed limit and $\ddot{q}_{i,\,\mathrm{lim}}$ is the acceleration limit for the i-th joint.

An effective strategy to solve the problem of ensuring path-tracking capabilities under joint constraints, is to slow down the time law of the desired trajectory when velocity or accelerations limits are encountered. This can be obtained by introducing a virtual time τ via a time warp in the trajectory generation, i.e.

$$\widetilde{\boldsymbol{x}}_{\mathrm{d}}(t) = \boldsymbol{x}_{\mathrm{d}}\big(\tau(t)\big), \tag{14}$$

where $\tau(t)$ is an increasing function such that $\tau(0) = 0$ and $\tau(t) \le t$. The latter condition is imposed so that the virtual time τ is not allowed to overtake the real time t; in other words, the execution time of the trajectory will never be shorter than its planned duration.

To solve the second-order inverse kinematics problem with respect to the virtual time, it is necessary to work out an algorithm in discrete time form which generates the time warp. To this aim, the value of the virtual time at the k-th instant, τ_k, is recursively calculated as

$$\tau_k = \tau_{k-1} + \Delta\tau_k, \tag{15}$$

where the increment $\Delta\tau_k$ solves, at each step, the following problem:

$$\begin{aligned} &\max \Delta\tau_k \\ &\text{subject to:} \\ &\Delta\tau_k \ge 0 \\ &t_k \ge \tau_{k-1} + \Delta\tau_k \\ &-\dot{q}_{i,\,\mathrm{lim}} \le \dot{q}_{i,k} \le \dot{q}_{i,\,\mathrm{lim}} \quad i = 1\ldots,n \\ &-\ddot{q}_{i,\,\mathrm{lim}} \le \ddot{q}_{i,k} \le \ddot{q}_{i,\,\mathrm{lim}} \quad i = 1\ldots,n. \end{aligned} \tag{16}$$

Since the relationship between the variables $\dot{q}_{i,k}$, $\ddot{q}_{i,k}$ and the increment $\Delta\tau_k$ —established through the equations (6,7,14)— is nonlinear because of (14), it is very difficult to find the general solution to the problem (16).

On the other hand, algorithmic solutions to the inverse kinematics problem are based on reduced-order approximations of the inverse kinematics. In detail, the second-order algorithm here considered is based on the linear mapping (6,10,11). Therefore, consistently with the above approximation, it is possible to assume that at each step the increment $\Delta\widetilde{\boldsymbol{x}}_{\mathrm{d}}$ of the desired trajectory $\widetilde{\boldsymbol{x}}_{\mathrm{d}}$ is a linear function of the time increment as follows

$$\Delta\widetilde{\boldsymbol{x}}_{\mathrm{d},k} = \dot{\widetilde{\boldsymbol{x}}}_{\mathrm{d},k}\Delta t \tag{17}$$

$$\widetilde{\boldsymbol{x}}_{\mathrm{d},k} = \widetilde{\boldsymbol{x}}_{\mathrm{d},k-1} + \Delta\widetilde{\boldsymbol{x}}_{\mathrm{d},k} \tag{18}$$

Based on (14), the equations (17,18) can be rewritten in the form

$$\Delta\widetilde{\boldsymbol{x}}_{\mathrm{d},k} = \dot{\boldsymbol{x}}_{\mathrm{d}}(\tau_{k-1})\Delta\tau_k \quad (19)$$

$$\widetilde{\boldsymbol{x}}_{\mathrm{d},k} = \boldsymbol{x}_{\mathrm{d}}(\tau_{k-1}) + \dot{\boldsymbol{x}}_{\mathrm{d}}(\tau_{k-1})\Delta\tau_k \quad (20)$$

which shows that at each step the reference trajectory can be computed by using the known functions (3,4) evaluated at the current virtual time instant.

In addition, the velocity and acceleration feedforward terms in (11) require computation of the actual desired velocity and acceleration seen with respect to the real time. These can be approximated from the known functions (4) and (5) as

$$\dot{\widetilde{\boldsymbol{x}}}_{\mathrm{d},k} = \dot{\boldsymbol{x}}_{\mathrm{d}}(\tau_{k-1})\frac{\Delta\tau_k}{\Delta t}, \quad (21)$$

$$\ddot{\widetilde{\boldsymbol{x}}}_{\mathrm{d},k} = \ddot{\boldsymbol{x}}_{\mathrm{d}}(\tau_{k-1})\left(\frac{\Delta\tau_k}{\Delta t}\right)^2. \quad (22)$$

It is worth noting that the actual desired velocity and acceleration are decreased with respect to the planned ones when $\Delta\tau_k$ is lower than Δt.

By using (20), (21) and (22) it is now possible to express the joint velocities and accelerations in the above second-order inverse kinematics algorithm (6) and (7) as a function of $\Delta\tau_k$, in the form

$$\ddot{\boldsymbol{q}}_k = \boldsymbol{a}_k\Delta\tau_k^2 + \boldsymbol{b}_k\Delta\tau_k + \boldsymbol{c}_k \quad (23)$$

$$\dot{\boldsymbol{q}}_k = \boldsymbol{d}_k\Delta\tau_k^2 + \boldsymbol{e}_k\Delta\tau_k + \boldsymbol{g}_k \quad (24)$$

which are quadratic in $\Delta\tau_k$. The coefficients that appear in (23,24) are configuration dependent; their expression is given in [7].

At this point, the problem (16) can be reformulated as:

$$\begin{aligned} &\max \Delta\tau_k \\ &\text{subject to:} \\ &\Delta\tau_k \geq 0 \\ &t_k \geq \tau_{k-1} + \Delta\tau_k \\ &-\dot{q}_{i,\lim} \leq d_{i,k}\Delta\tau_k^2 + e_{i,k}\Delta\tau_k + g_{i,k} \leq \dot{q}_{i,\lim} \\ &\qquad i = 1\ldots,n \\ &-\ddot{q}_{i,\lim} \leq a_{i,k}\Delta\tau_k^2 + b_{i,k}\Delta\tau_k + c_{i,k} \leq \ddot{q}_{i,\lim} \\ &\qquad i = 1\ldots,n \\ &\Delta\tau_k \leq \Delta t \end{aligned} \quad (25)$$

where the last constraint has been added to avoid the large jerks that might be experienced if the virtual time were allowed to recover the real time in a single step.

The solution to (25) is easier to find than the one to (16) since the problem is now quadratic and one-dimensional. Details on the solution process are given in [7]. By denoting the sought solution as $\Delta\tau_{sol,k}$, the second-order inverse kinematics algorithm (23,24,8) becomes:

$$\ddot{\boldsymbol{q}}_k = \boldsymbol{a}_k\Delta\tau_{sol,k}^2 + \boldsymbol{b}_k\Delta\tau_{sol,k} + \boldsymbol{c}_k \quad (26)$$

$$\dot{\boldsymbol{q}}_k = \boldsymbol{d}_k\Delta\tau_{sol,k}^2 + \boldsymbol{e}_k\Delta\tau_{sol,k} + \boldsymbol{g}_k \quad (27)$$

$$\boldsymbol{q}_k = \boldsymbol{q}_{k-1} + \dot{\boldsymbol{q}}_k\Delta t - \ddot{\boldsymbol{q}}_k\Delta t^2/2 \quad (28)$$

which requires all quantities already computed to solve (25).

4 Scaling technique

At some system states the problem (25) may not have solution, i.e., there is no choice for $\Delta\tau$ —even the null one— ensuring the path under the given joint constraints. A simple example of such a state, is that of a vehicle approaching an unplanned narrow turn with high velocity and low braking capability. It must be remarked that the proposed algorithm computes the time warp on-line; in fact, differently from off-line approaches (e.g. the dynamic scaling technique [8]) in which *all* the motion is suitably slown down, the time scaling is here decided at each step. In this Section, an algorithm to handle those situations is presented. Notice that the proposed approach is aimed at handling only the critical situations and is no longer used as soon as the problem (25) admits a solution.

Considering the first-order differential kinematics

$$\dot{\boldsymbol{x}} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} \quad (29)$$

and its inversion with a CLIK approach in discrete time

$$\dot{\boldsymbol{q}}_k = \boldsymbol{J}_{k-1}^{-1}\left[\dot{\boldsymbol{x}}_{\mathrm{d}}(\tau_k) + \boldsymbol{\Lambda}_p\big(\boldsymbol{x}_{\mathrm{d}}(\tau_k) - \boldsymbol{x}_{k-1}\big)\right], \quad (30)$$

where $\boldsymbol{\Lambda}_p$ is a suitable constant positive-definite matrix, our objective is to find, at time t_k, a joint velocity vector $\dot{\boldsymbol{q}}_k$ that satisfies the velocity and acceleration constraints. Since the problem (16) does not have solution, in order to guarantee the limits for $\dot{\boldsymbol{q}}_k$ and $\ddot{\boldsymbol{q}}_k$, we will not be able to strictly satisfy also the path constraints.

Roughly speaking, in a path-tracking task when we cannot achieve the desired velocity at the end effector, our aim is to keep at least its direction. If this is not possible under the given joint constraints, we want to generate an end-effector velocity whose direction is *as close as possible* to the desired one.

If at least one of the joint velocities is out of its limits we first consider the coefficients α_i obtained by:

$$\alpha_i = \frac{|\dot{q}_i|}{\dot{q}_{i,\lim}} \qquad i = 1,\ldots,n. \quad (31)$$

Then, by denoting as $\alpha = \max_i(\alpha_i)$, we scale the velocity vector given by (30) so as to obtain

$$\dot{\boldsymbol{q}}_k = \frac{1}{\alpha}\boldsymbol{J}_{k-1}^{-1}\left[\dot{\boldsymbol{x}}_{\mathrm{d}}(\tau_k) + \boldsymbol{\Lambda}_p\big(\boldsymbol{x}_{\mathrm{d}}(\tau_k) - \boldsymbol{x}_{k-1}\big)\right]. \quad (32)$$

Notice that this solution corresponds to an end-effector velocity which is parallel to the vector $\dot{\boldsymbol{x}}_{\mathrm{d}}(\tau_k) + \boldsymbol{\Lambda}_p(\boldsymbol{x}_{\mathrm{d}}(\tau_k) - \boldsymbol{x}_{k-1})$, i.e., the end effector is moving along the desired direction.

In absence of acceleration constraints eq. (32) guarantees the fulfillment of the velocity constraints while

keeping the end-effector velocity along the desired direction. However, when acceleration constraints must be also taken into account, eq. (32) gives only a first tentative value for the joint velocity vector.

In fact, from $\dot{\boldsymbol{q}}_k$ in (32) it is possible to compute the corresponding acceleration as

$$\ddot{\boldsymbol{q}}_k = (\dot{\boldsymbol{q}}_k - \dot{\boldsymbol{q}}_{k-1})/\Delta t. \tag{33}$$

Then, for each joint acceleration that is over its limits, the corresponding joint velocity is modified as follows

$$\dot{q}_{j,k} = \dot{q}_{j,k-1} + \mathrm{sign}(\ddot{q}_{j,k}) \cdot \ddot{q}_{j,\mathrm{lim}}\, \Delta t. \tag{34}$$

Notice that the obtained joint velocity vector leads to a joint motion complying with the given limits; however, the end effector is driven to a direction which is no longer parallel to the vector $\dot{\boldsymbol{x}}_\mathrm{d}(\tau_k)+\boldsymbol{\Lambda}_p(\boldsymbol{x}_\mathrm{d}(\tau_k)-\boldsymbol{x}_{k-1})$.

It is worth remarking that satisfaction of all the constraints —i.e., path, velocity and acceleration— can be achieved, in the worst condition, if, no matter how many joints velocities are saturated, no more than one acceleration is saturated. In this case, in fact, it is still possible to achieve a scaling factor for all the velocity components, thus keeping the desired end-effector velocity direction.

Handling of the time warp is also of concern. In this case, however, the virtual time increment is computed *after* that the joints velocity and acceleration are obtained. An euristic solution, consistent with eq. (21), is the following

$$\Delta\tau_k = \frac{\|\boldsymbol{J}_k\dot{\boldsymbol{q}}_k\|}{\|\dot{\boldsymbol{x}}_\mathrm{d}(\tau_k) + \boldsymbol{\Lambda}_p(\boldsymbol{x}_\mathrm{d}(\tau_k) - \boldsymbol{x}_{k-1})\|}\, \Delta t, \tag{35}$$

which acts so as to slow down the desired trajectory of a quantity proportional to the mismatching between the obtained end-effector velocity and the desired one.

5 Case studies

The proposed algorithm has been implemented in the MATLAB programming environment to demonstrate its application to numerical case studies. For the sake of clarity, a simple two-degree-of-freedom RR planar manipulator is considered whose link lengths are 0.3683 m and 0.2413 m. These correspond to the kinematic parameters of the Direct Drive Manipulator Research and Development Package by Integrated Motions Incorporated.

The initial configuration and end-effector position are

$$\begin{aligned} \boldsymbol{q}(0) &= [-0.4757 \quad 1.2496]^\mathrm{T} \quad \mathrm{rad}, \\ \boldsymbol{x}(0) &= [0.5 \quad 0]^\mathrm{T} \quad \mathrm{m}. \end{aligned}$$

The path to be followed is a circle with center at $[0.4 \quad 0]^\mathrm{T}$ m and radius 0.1 m; it can be recognized that this path does not involve crossing of singular configurations. The assigned time law is a quintic polynomial that allows spanning of the whole path with null initial and final velocities and acceleration; the planned duration of the motion is 1 s.

The closed-loop inverse kinematics algorithm is implemented in discrete time, with a time interval $\Delta t = 0.001$ s and gain matrices $\boldsymbol{K}_p = \mathrm{diag}\{1000, 1000\}$, $\boldsymbol{K}_v = \mathrm{diag}\{1000, 1000\}$, $\boldsymbol{\Lambda}_p = \mathrm{diag}\{100, 100\}$.

The joint limits are set as follows:

$$\begin{aligned} \dot{q}_{1,\mathrm{lim}} &= 1.5 \text{ rad/s} & \ddot{q}_{1,\mathrm{lim}} &= 10 \text{ rad/s}^2 \\ \dot{q}_{2,\mathrm{lim}} &= 1.5 \text{ rad/s} & \ddot{q}_{2,\mathrm{lim}} &= 15 \text{ rad/s}^2. \end{aligned}$$

In a first run the technique explained in Section 4 has not been used. This corresponds to using the algorithm described in [7]; the obtained results are reported in Figure 1. It can be recognized that at the beginning of the motion both the joint velocities and accelerations are kept inside their limits. However, the algorithm fails to find a solution after almost half path has been accomplished.

The new technique to handle configurations at which there is no admissible $\Delta\tau_k$, presented in Section 4, is then used to execute the same task as before and the results are shown in Figure 2. It can be recognized that the path is safely executed on-line without *a priori* knowledge of the trajectory and keeping the velocity and acceleration constraints in their bounds. Notice that in this case the scaling technique is used only in the time intervals $t \in [0.78, 0.82]$ s, $t \in [0.99, 1.12]$ s and $t \in [1.54, 1.68]$ s. As expected, in this case the desired path is not strictly followed; however, the actual end-effctor path is kept close to the desired one.

To investigate the robustness of the algorithm with respect to changes in the desired time law, we have finally assigned to the manipulator the same task as before to be executed in half the time (i.e., the planned duration of the motion is lowered to 0.5 s). The simulation results are reported in Figure 3. It can be recognized that the obtained time law is very similar to the one resulting from the previous simulation; this confirms that this motion is nearly the fastest that can be achieved along the given path with the given joint limits. The small overshoot with respect to the final end-effector position is due to the very fast approch to the endpoint of the trajectory. This effect can be reduced with some forward knowledge of the desired motion that can be recovered by looking at the real-time trajectory data overtaking the virtual-time ones.

6 Conclusions

In this paper a new CLIK algorithm for end-effector path tracking has been developed which takes into account both velocity and acceleration joint limits. To the purpose, at each step, the algorithm computes a time warp that suitably modifies the assigned time law. The proposed method operates on-line; therefore, the

algorithm can be adopted for real-time kinematic control of manipulators. Future work will embed forward knowledge of the desired motion to improve the algorithm's performance. The problem of handling torque constraints is also currently under investigation.

References

[1] B. Allotta, G. Bioli, and V. Colla, "A repeatable control scheme for redundant manipulators observing joint mechanical limits," *Proc. of Robotics Towards 2000: 27th Int. Symp. on Industrial Robots*, Milano, I, pp. 521–526, Oct. 1996.

[2] A. Balestrino, G. De Maria and L. Sciavicco, "Robust control of robotic manipulators," *Prep. of the 9th IFAC World Congress*, Budapest, H, **6**, pp. 80–85, July 1984.

[3] F. Caccavale, S. Chiaverini, and B. Siciliano, "Second-order kinematic control of robot manipulators with Jacobian damped least-squares inverse: Theory and experiments," *IEEE/ASME Trans. on Mechatronics*, **2**(3)188–194, 1997.

[4] T.F. Chan and R.V. Dubey, "A weighted least-norm solution based scheme for avoiding joint limits for redundant manipulators," *IEEE Trans. on Robotics and Automation*, **11**, pp. 286–292, 1995.

[5] P. Chiacchio and S. Chiaverini, "Coping with joint velocity limits in first-order inverse kinematics algorithms: Analysis and real-time implementation," *Robotica*, **13**, pp. 515–519, 1995.

[6] P. Chiacchio, S. Chiaverini, L. Sciavicco and B. Siciliano, "Closed-loop inverse kinematics schemes for constrained redundant manipulators with task-space augmentation and task-priority strategy," *The Int. J. of Robotics Research*, **10**, pp. 410–425, 1991.

[7] S. Chiaverini, G. Fusco, "A new inverse kinematics algorithm with path tracking capability under velocity and acceleration constraints," *Proc. of the 38th Conf. on Decision and Control*, Phoenix, AZ, pp. 2064–2069, Dec. 1999.

[8] J.M. Hollerbach, "Dynamic scaling of manipulator trajectories," *Trans. of the ASME — J. of Dynamic Systems, Measurements, and Control*, **106**, pp. 102–106, 1984.

[9] H. Seraji and R. Colbaugh, "Improved configuration control for redundant robots," *J. of Robotic System*, **7**, pp. 897–928, 1990.

[10] Z. Shiller, "Time-Energy Optimal Control of Articulated Systems with Geometric Path Constraints," *Proc. of the 1994 IEEE Int. Conf. on Robotics and Automation*, San Diego, CA, pp. 2680–2685, 1994.

[11] C.W. Wampler and L.J. Leifer, "Applications of damped least-squares methods to resolved-rate and resolved-acceleration control of manipulators," *Trans. of the ASME — J. of Dynamic Systems, Measurements, and Control*, **110**, pp. 31–38, 1988.

[12] ,C.Y.E. Wang, W.K. Timoszyk and J.E. Bobrow, "Weightlifting Motion Planning for A Puma 762 Robot," *Proc. of the 1999 IEEE Int. Conf. on Robotics and Automation*, Detroit, MI, pp. 480–485, May 1999.

[13] W.A. Wolovich and H. Elliott, "A computational technique for inverse kinematics," *Proc. of the 23rd IEEE Conf. on Decision and Control*, Las Vegas, NV, pp. 1359–1363, Dec. 1984.

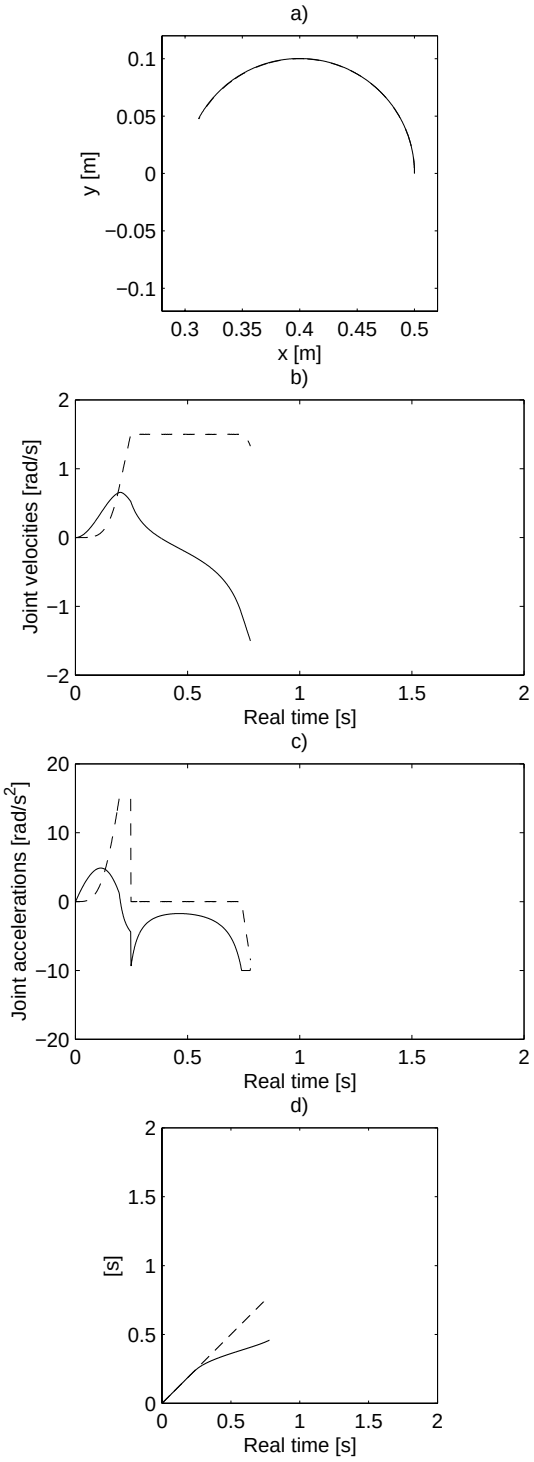

Figure 1: Second-order CLIK algorithm with time warp under velocity and acceleration constraints: a) desired (dashed) and actual (solid) path of the end effector; b) joint 1 (solid) and joint 2 (dashed) velocities; c) joint 1 (solid) and joint 2 (dashed) accelerations; d) real time (dashed) and virtual time (solid).

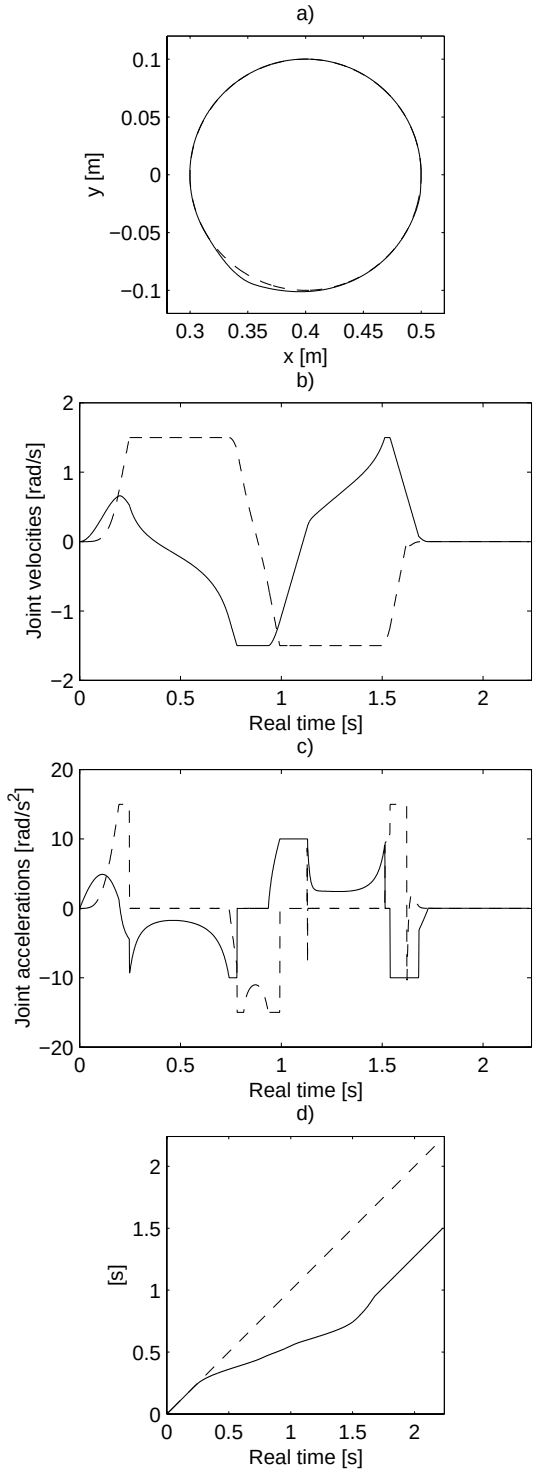

Figure 2: Second-order CLIK algorithm with time warp under velocity and acceleration constraints resorting to the new scaling technique: a) desired (dashed) and actual (solid) path of the end effector; b) joint 1 (solid) and joint 2 (dashed) velocities; c) joint 1 (solid) and joint 2 (dashed) accelerations; d) real time (dashed) and virtual time (solid).

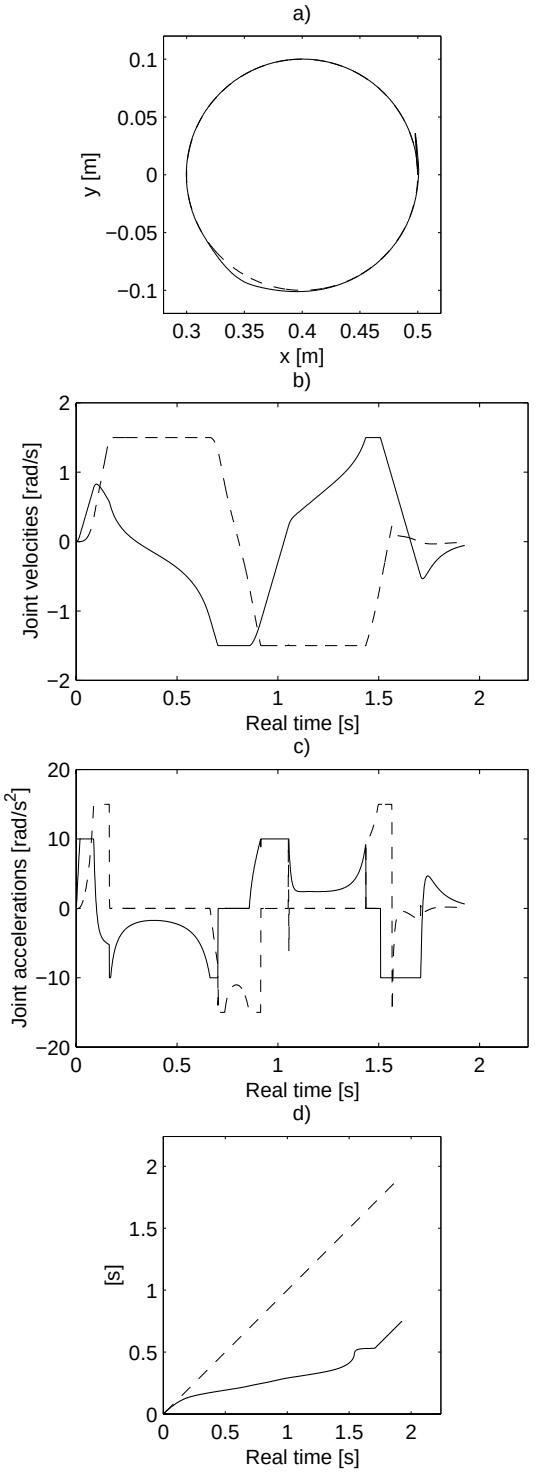

Figure 3: Second-order CLIK algorithm with time warp under velocity and acceleration constraints resorting to the new scaling technique, final time $t_f = 0.5\,\mathrm{s}$: a) desired (dashed) and actual (solid) path of the end effector; b) joint 1 (solid) and joint 2 (dashed) velocities; c) joint 1 (solid) and joint 2 (dashed) accelerations; d) real time (dashed) and virtual time (solid).