Robotics and Autonomous Systems 59 (2011) 519–529

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

# Sliding mode speed auto-regulation technique for robotic tracking

Fabricio Garelli [a,*], Luis Gracia [b], Antonio Sala [b], Pedro Albertos [b]

[a] CONICET and Universidad Nacional de La Plata, C.C.91 (1900), La Plata, Argentina

[b] Instituto de Automática e Informática Industrial, Universitat Politècnica de València, Camino de Vera s/n, 46022 Valencia, Spain

ARTICLE INFO

*Article history:*
Received 12 February 2010
Accepted 23 March 2011
Available online 13 April 2011

*Keywords:*
Robotic tracking
Sliding mode
Input saturation
Multivariable systems

ABSTRACT

In advanced industry manufacturing involving robotic operations, the required tasks can be frequently formulated in terms of a path or trajectory tracking. In this paper, an approach based on sliding mode conditioning of a path parametrization is proposed to achieve the greatest tracking speed which is compatible with the robot input constraints (joint speeds). Some distinctive features of the proposal are that: (1) it is completely independent of the robot parameters, and it does not require a priori knowledge of the desired path either, (2) it avoids on-line computations necessary for conventional analytical methodologies, and (3) it can be easily added as a supervisory block to pre-existing path tracking schemes. A sufficient condition (lower bound on desired tracking speed) for the sliding mode regulation to be activated is derived, while a chattering amplitude estimation is obtained in terms of the sampling period and a tunable first-order filter bandwidth. The algorithm is evaluated on the freely accessible 6R robot model PUMA-560, for which a path passing through a wrist singularity is considered to show the effectiveness of the proposal under hard tracking conditions.



## 1. Introduction

A major issue in robotics is the tracking of *reference trajectories*. In most practical applications that use industrial and/or mobile robots [1,2] (e.g., machining, arc-welding, adhesive application, spray painting, assembling, inspection, object transportation in warehouses, surveillance in known environments, etc.), the robot task is based on tracking a given *path* with *negligible* error and with the highest possible velocity, so that the cycle time of the robot task is minimized. In this manner, both quality and productivity indexes can be enlarged. However, both the accuracy and the speed with which this tracking can be performed is strongly related with the joint actuators physical limitations, which are seldom considered in commercial robots to regulate the robot forward motion. Instead, the tracking speed usually has to be computed a priori by the robot operator in order to avoid an error message.

The reference path, i.e. the path to be followed, can usually be expressed as a one-dimensional curve in the Cartesian space, i.e., a time-dependent vector which can be parameterized in terms of a scalar *motion parameter* whose first-order time derivative is related to the path *tracking speed* by well-known expressions. In this sense, the idea of a parameterized path has been successfully used in several research works to adjust the tracking speed so that the robot is able to track different types of references, even those ones crossing robot kinematics singularities [3–6]. Among the more significant works in this research line, a self-paced fuzzy controller was designed in [7] to adjust the tracking speed of two-dimensional paths in accordance with contour conditions such as curvature. Similarly, a path parametrization satisfying input and state constraints was obtained in [8] using look-ahead optimization and a prediction of the evolution of the robot, for which a priori knowledge of the desired path and a robot model are required. More recently, a time warp is considered in [9] to slow down the task-space trajectory when joint limits are encountered. In [10], instead, the power limits of the electrical motors driving the robot are considered to measure the maximum possible velocity and force that can be physically generated by the robot to perform the required task. Finally, path tracking is rigorously divided into a geometric (desired error) and a dynamic (desired speed) task in [11], where speed profiles are assigned for nonlinear systems to track non-smooth paths.

This paper proposes a simple method which allows regulating the robotic tracking speed in order to avoid path deviations because of joint actuators constraints. In order to achieve this goal, a sliding mode auxiliary loop is added to conventional path tracking schemes, which is inspired on recent reference conditioning algorithms developed to deal with constraints in multivariable control systems [12,13]. It acts as a supervisory block, since it is only activated when the desired speed would lead the joint actuators to reach their limit values. Interestingly, a practical consequence is the fact that, if a sufficiently high

* Corresponding author.
*E-mail addresses:* fabricio@ing.unlp.edu.ar (F. Garelli), luigraca@isa.upv.es (L. Gracia), asala@isa.upv.es (A. Sala), pedro@aii.upv.es (P. Albertos).

0921-8890/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.robot.2011.03.007

speed reference (motion parameter) is set, the method computes the maximal tracking speed which is compatible with the joint actuator limits. As an advantage over most of the above cited proposals, the proposed technique is independent of the main path tracking control algorithm and it does not require a priori knowledge of the desired path. Since the method was thought to be used with commercial industrial robots, speed joint constraints are assumed (see Section 2). A well-known six-revolute (6R) robotic arm is taken as case study, for which a path passing through a robot singularity is considered. The method implementation can be even carried out by means of analog electronics since the switching device is confined to the low-power side of the system.

The article is organized as follows. In the next section the classical kinematic control scheme for robotic path tracking is recalled, and some common alternatives to deal with actuator constraints are introduced. Section 3 presents some basic concepts of variable structure theory and develops the sliding mode auto-regulation technique for tracking speed in order to avoid path errors due to actuator nonlinearities. In Section 4 simulation results are presented using the free-access 6R robot model PUMA-560, for which the main distinctive features of the method are illustrated. Finally, some conclusions are given.

## 2. Classical control scheme for robotic path tracking

Due to the computational complexity of advanced control algorithms developed in current robotics research [14,15], classical control techniques are still widely used in industrial robot applications. In most practical robot systems, the controller consists of three nested control loops: an analog actuator current controller, an analog velocity controller, and a typically digital position controller. The great majority of industrial robot manufacturers implement the inner control loops (i.e., the current loop and the velocity loop) internally in the so-called *joint controllers* and do not allow the robot operator to modify these loops. Conversely, the outer-loop position controller is usually open for the user and can be manipulated.

Let us now discuss some common setups for robot path tracking in the above described framework.

### 2.1. Kinematic control setups

*Workspace-coordinates kinematic control.* Let us denote as $\mathbf{p}_{\text{ref}}(t)$ the position reference defining the desired path in some user-chosen workspace coordinates (for instance, Cartesian position and Euler-angle orientation of the end effector). As mentioned in the introduction, the trajectory $\mathbf{p}_{\text{ref}}(t)$ can be usually expressed in terms of a desired path function $\mathbf{f}(\lambda)$ whose argument is the so-called motion parameter $\lambda(t)$ as

$$\mathbf{p}_{\text{ref}} = \mathbf{f}(\lambda), \tag{1}$$

and, therefore, the desired speed comes from:

$$\dot{\mathbf{p}}_{\text{ref}} = \frac{\partial \mathbf{f}}{\partial \lambda}\dot{\lambda}. \tag{2}$$

A kinematic control block in a robot closes a loop using position information in both joint coordinates, to be denoted as $\mathbf{q}$, and workspace coordinates $\mathbf{p}$, as well as desired position and speed information from the target trajectory.

The relationship between the $\mathbf{q}$ configuration and the end-effector position/orientation $\mathbf{p}$ is highly nonlinear, generically expressed as:

$$\mathbf{p} = \mathbf{l}(\mathbf{q}), \tag{3}$$

where the function $\mathbf{l}$ is called the kinematic function of the robot model. The first-order kinematics results in:

$$\dot{\mathbf{p}} = \frac{\partial \mathbf{l}}{\partial \mathbf{q}}\dot{\mathbf{q}} = J(\mathbf{q})\dot{\mathbf{q}}, \tag{4}$$

where $J(\mathbf{q})$ is denoted as the Jacobian matrix or simply *Jacobian* of the kinematic function.

Fig. 1 shows a common setup for the kinematic control block in robot path tracking, consisting of a two-degree of freedom (2-DOF) control structure which incorporates a correction based on the position error $\mathbf{e}_p = \mathbf{p}_{\text{ref}} - \mathbf{p}$ by means of the position loop controller $C_p$ plus a feedforward term depending on the first-order time derivative of the position reference, i.e. $\dot{\mathbf{p}}_{\text{ref}}$.

Note that, in this scheme the error correction is performed in the Cartesian space and then the inverse of the robot Jacobian $J(q)$ is used to obtain the joint velocity vector $\dot{\mathbf{q}}$. Indeed, for a non-redundant manipulator (square Jacobian), the joint velocities $\dot{\mathbf{q}}$ producing a particular end-effector motion $\dot{\mathbf{p}}_0$ can be written as:

$$\dot{\mathbf{q}} = J^{-1}(\mathbf{q})\dot{\mathbf{p}}_0, \tag{5}$$

and the kinematic control loop is in charge of determining the desired value for $\dot{\mathbf{p}}_0$ as a function of current position ($\mathbf{p}$) and current target trajectory point ($\mathbf{p}_{\text{ref}}$) and speed $\dot{\mathbf{p}}_{\text{ref}}$. Once $\dot{\mathbf{p}}_0$ is computed, (5) is applied and sent to the actuators.

The Jacobian of a generic robotic arm can be easily obtained with the vectorial approach described in [16]. It is well-known that there are certain workspace limits and internal positions where $J$ is singular. This matter is later discussed in Section 4.

*Joint-coordinates kinematic control.* Another conventional approach for kinematic control consists of performing the error correction directly in the joint space [17]. This second approach requires to compute the position inverse kinematics, i.e., $\mathbf{q} = \mathbf{l}^{-1}(\mathbf{p})$. In any case, the proposed technique also applies for that or any other kinematic control.

### 2.2. Dealing with actuator constraints

In order to account for input constraints, joint speed saturation is from now on considered between the desired joint speeds $\dot{\mathbf{q}}_d$ of Fig. 1 and the achievable ones, denoted as $\dot{\mathbf{q}}_{ds}$.

Naturally, the maximum values of the *robot control signals*, which are given by the power constraints of the actuators, limit the path tracking speed. Basically, the following three approaches can be found in practical applications in order to face with robot actuators constraints:

(a) To use a (conservative) low tracking speed, so that the robot control signals never exceed their maximum values.
(b) To also use a fixed tracking speed, but higher than the previous one, in such a way that the robot control signals saturate at least once during the tracking.
(c) To compute for each point on the path the maximum tracking speed allowed by the limits of the control signals and to use that value for the motion parameter speed.

The first approach is extremely conservative and thus a not advisable solution. In effect, it gives rise to an excessively slow path tracking, which indeed wastes the tracking capabilities of the robotic system. The second approach, which is the classical one, has as its main drawback that when the control signals are saturated the robot losses the reference and even leaves the desired path, which makes it inappropriate for high-accuracy applications. The third option is the best choice among the three listed practical approaches; however, it depends on the desired path and on the robot Jacobian and, hence, it is more involved computationally and, furthermore, modeling errors might give a speed over the desired limits.

In practical implementations of the second or third options, the actuator limitations are typically faced in two different ways: