Control Engineering Practice 59 (2017) 127–136

Contents lists available at ScienceDirect

Control Engineering Practice

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

Motion planning for robotic manipulators using robust constrained control

Andrea Maria Zanchettin*, Paolo Rocco

Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria, Piazza Leonardo Da Vinci 32, Milano, Italy

ARTICLE INFO

Keywords:
Robotic manipulators
Trajectory planning
Constraints
Robustness
Uncertainty
Industrial robots

ABSTRACT

Since their first appearance in the 1970's, industrial robotic manipulators have considerably extended their application fields, allowing end-users to adopt this technology in previously unexplored scenarios. Correspondingly, the way robot motion can be specified has become more and more complex, requiring new capabilities to the robot, such as reactivity and adaptability. For an even enhanced and widespread use of industrial manipulators, including the newly introduced collaborative robots, it is necessary to simplify robot programming, thus allowing this activity to be handled by non-expert users. Next generation robot controllers should intelligently and autonomously interpret production constraints, specified by an application expert, and transform them into motion commands only at a lower and real-time level, where updated sensor information or other kind of events can be handled consistently with the higher level specifications. The availability of several execution strategies could be then effectively exploited in order to further enhance the flexibility of the resulting robot motion, especially during collaboration with humans.

This paper presents a novel methodology for motion specification and robust reactive execution. Traditional trajectory generation techniques and optimisation-based control strategies are merged into a unified framework for simultaneous motion planning and control. An experimental case study demonstrates the effectiveness and the robustness of this approach, as applied to an image-guided grasping task.

1. Introduction

Robot motions are typically programmed by means of Cartesian and angular position and velocity profiles of the end-effector along a given path. Programming a specific task, which happens in today's motion generation algorithms, results in solving the motion planning problem by actually over-constraining the space of solutions in order to select a particular end-effector motion among others. Even more advanced and commercially available trajectory planning strategies prevent the low-level controller from adapting or modifying the generated trajectory based on real-time events or sensor readings, or need a lot of handling logics to be pre-programmed, rarely guaranteeing hard real-time capabilities or reduced reaction times.

Constraint-based programming of robot motions represents the most natural solution to the problem of indirectly planning a robot trajectory based on process requirements, which are specified by means of constraints. The constraint-based approach, originally introduced in Siciliano and Slotine (1991) based on the task-function formalism, Samson, Espiau, and Borgne (1991), has been recently and intensively exploited within the so-called iTaSC (instantaneous Task Specification using Constraints) framework, see De Schutter et al. (). While originally developed to cope only with instantaneous constraints (i.e., those corresponding to the current time instant), recent extensions towards a more comprehensive constraints representation are reported, see e.g. Decre, Bruyninckx, and De Schutter (2013). A very similar approach has been presented within the Stack of Tasks framework, see Mansard, Khatib, and Kheddar (2009), Escande, Mansard, and Wieber (2014). Similar to the Stack of Tasks, the iCAT (inequality Control objectives, Activations and Transitions) task priority framework was introduced in Simetti and Casalino (2016) to deal with smooth transitions during task activation/deactivation. Yet another similar prioritised framework has been recently proposed in Tazaki and Suzuki (2014). In Kermorgant and Chaumette (2014), another control method dealing with constraints is presented: constraints (such as joint limits and field-of-view constraints) are conveniently defined as additional costs to be optimised and activated by suitable thresholds.

A constraint-based approach to accommodate joint position limits, as well as velocity, acceleration or even torque ones has been proposed in Antonelli, Chiaverini, and Fusco (2003), Flacco and De Luca (2015) with reactive capabilities based on time-based trajectory scaling. These methods, however, only rely on pure kinematic or dynamic rescaling of a predefined trajectory, which limits the reaction capabilities of the robot. In fact, a simple scaling technique not always guarantees the existence of a feasible motion. Moreover, such approaches only deal

* Corresponding author.
E-mail addresses: andreamaria.zanchettin@polimi.it (A.M. Zanchettin), paolo.rocco@polimi.it (P. Rocco).

http://dx.doi.org/10.1016/j.conengprac.2016.11.010
Received 10 May 2016; Received in revised form 8 September 2016; Accepted 16 November 2016

with decoupled constraints in the joint space like the avoidance of joint position, velocity or acceleration limits.

In the framework of constraint-based trajectory planning, it is important to mention the works in Macfarlane and Croft (2003), Biagiotti and Melchiorri (2008), Kroeger and Wahl (2010) which present various approaches for online generation of smooth and time-optimal trajectories for multi-axes machines and robots.

For a better and reliable task execution, the problem of trajectory planning and the subsequent (constraint-based) control should be merged in a unified framework with capabilities of both motion planning and reactive control/execution. Beside the constraint-based specification, the first idea of connecting planning and reactive/adaptation capabilities was developed in Quinlan and Khatib (1993), within the well-known Elastic Strips framework, and later refined within the Elastic Bands framework, Brock and Khatib (2002). Other examples of real-time reaction planners can be found in, e.g., Haddadin et al. (2010), where virtual and physical contact are merged to form the desired reactive behaviour, or in Khansari-Zadeh and Billard (2012), where the reactive behaviour of the robot is obtained by the imposition of a proper attractor dynamics. Finally (Hauser, 2012) presents a re-planning strategy to be executed when the nominal plan fails, due to unpredictable obstacles movements.

This work presents a method to combine a trajectory generation algorithm with a constrained optimisation problem, which relies on a continuously updated reference trajectory and a reactive control strategy. Further details on the features of the proposed algorithm and on how the method stands with respect to available results are discussed in the next Section. The remainder of this paper is organised as follows. Section 2 describes the motivations underlying this work and discusses the original contribution with respect to the existing literature. In Section 3, the main contribution is detailed and an algorithm for constraint-based reactive trajectory generation is presented. Section 4 briefly discusses two extensions to cope with redundant robots and to enforce a compliant behaviour of the robot, respectively. Section 5 complements the specification of the algorithm with best practice to guarantee robustness of the overall system with respect to measurement uncertainties. Finally, an experimental case study, describing how a selected application may benefit from the proposed approach, is reported in Section 6.

2. Motivations and problem setting

In this Section, we first motivate our work and then discuss the main features of the approach as compared with the state of the art.

2.1. Motivations and key concepts

Today's robots are able to execute tasks that are way more complex than those they were originally designed for. Is worth noticing that the VAL scripting language, developed by Unimation Robotics in the mid 1970's, consisted in no more than 30 instructions. Nowadays, advanced scripting languages for modern industrial robots, like ABB's RAPID or KUKA KRL, include hundreds of different instructions. Therefore, as the adoption of robots is becoming more and more pervasive, novel applications require more and more advanced functionalities, hence more involved programming primitives, especially to promptly react or adapt to unpredicted situations.

The main idea behind this work is to develop a control architecture that endows the robot with advanced flexibility during task execution. Specifically, we propose to use control tools to move from an *imperative programming* paradigm (i.e. specifying the robot how to perform a task), in which process requirements are semantically and uniquely mapped by the robot programmer into a suitable end-effector velocity profile, towards a *declarative programming* paradigm (i.e. specifying requirements for task execution, and leave to the robot the autonomy to execute it properly). With this paradigm, process requirement and constraints are turned by the controller into motion commands only at run-time,

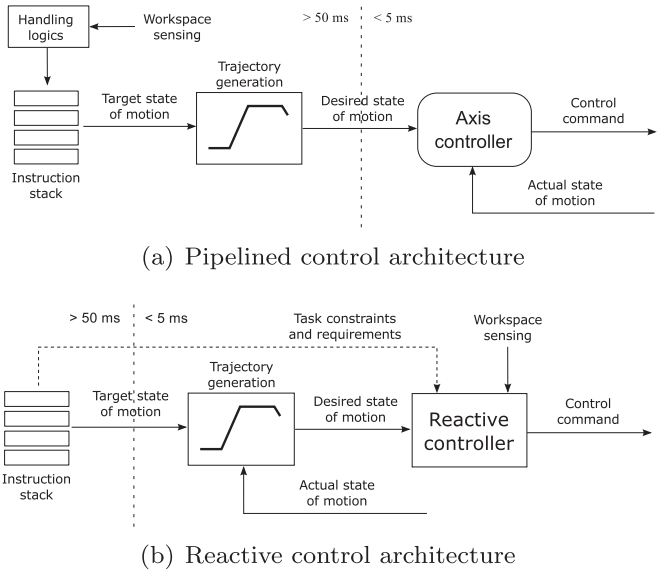

Fig. 1. Proposed control architecture (bottom) as compared to a traditional solution (top).

hence embedding the capability of handling real-time events, with reduced pre-programmed control logics. This shift of paradigm represents a major improvement for the control of new generation robots working in unstructured environments, and thus allows a more widespread use of robotic technology, especially in SMEs.

A pictorial view of the idea is sketched in Fig. 1, as compared to the traditional pipelined approach. In traditional robot programming, see Fig. 1(a), the trajectory is offline computed and possibly optimised, whilst being only online evaluated, hence leading to a completely pipelined and non reactive execution. This architecture may introduce inefficient handling strategy of sensed events, as the event has to traverse the whole architecture before ultimately influencing the control commands. Moreover, the programmer should specify how to handle each single event or deviation from the nominal task execution when a corresponding trajectory has been already generated. This setting has clearly two major drawbacks: (a) the user of the robotic application should be simultaneously an expert of both the process and the typical robotic programming structures, and (b) a significant lag between an event and the corresponding handle triggering may be thus introduced.

Within this work, we propose a method, see Fig. 1(b), to continuously re-generate, and hence evaluate, the reference trajectory towards a given goal depending on the current state of the robot. The reactive controller generates the next control command based on the current desired state of motion and of the workspace sensing, with the aim of tracking the desired motion to the extent allowed by the constraints specified by the programmer. If this entails a deviation from the planned trajectory, a new trajectory from the current state of motion to the target is generated. Furthermore the robot is endowed with reactive capabilities within a millisecond time scale, something difficult, if not impossible, to achieve with a traditional pipelined architecture. This way the robotic programmer is free to focus only on process requirements (which are then suitably turned into constraints) without caring of *what-if* handling strategies to define each possible non-nominal behaviour to cope with at execution time. The main task of the programmer is then to specify all the common features of each possible nominal behaviour, whilst a particular one will be eventually selected during execution, based on the actual context (i.e., presence of human co-workers, moving obstacles, etc).

2.2. Contributions and comparisons

Most of the available results in the literature, like the iTaSC (De