Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 44 (2016) 341 – 346

www.elsevier.com/locate/procedia

6th CIRP Conference on Assembly Technologies and Systems (CATS)

Towards Energy Optimization using Trajectory Smoothing and Automatic Code Generation for Robotic Assembly

Daniel Gleeson[a,b,*], Staffan Björkenstam[a], Robert Bohlin[a], Johan S. Carlson[a], Bengt Lennartson[b]

[a]*Geometry and Motion Planning, Fraunhofer-Chalmers Centre, Chalmers Science Park, SE-412 88 Göteborg, Sweden*

[b]*Automation Research Group, Department of Signals and Systems, Chalmers University of Technology, SE-412 96 Göteborg, Sweden*

[*] Corresponding author. Tel.: +46 (0)31-772 4243; fax: +46 (0)31-772 4260. *E-mail address:* daniel.gleeson@fcc.chalmers.se

Abstract

In automated industrial production, the efficiency of robotic motions directly affects both the final throughput and the energy consumption. By simulating and optimizing robot trajectories, cycle times and energy consumption can be lowered, or redundant robots can be detected. Here a polynomial basis function trajectory parametrization is presented, which enables direct export to executable robot code, and reduces the number of variables in the optimization problem. The algorithm finds time-optimal trajectories, while including collision avoidance and fulfilling joint, velocity and acceleration limitations. Applied torques are used as an approximation of the energy consumption to analyse the smooth trajectories, and successful tests show potential reductions of 10% for a standard industrial robot stud welding station.

Keywords: Robotics; Motion Planning; Optimal Control; Industrial Production

1. Introduction

In manufacturing industries using industrial robots and having a high level of automation, e.g. the automotive industry, the setup of an assembly line can be a highly complex task. It has to be performed every time a new product is to be produced, and when changes are made to the production line. By automating as much as possible of the setup phase, new products can come faster into production, and factory down-time can be reduced. Using optimization, the cycle times of the robots in each work station can be lowered, and there is an opportunity to also reduce the energy consumed.

Both in academia and in industry, the problem of simplifying the generation of assembly line robot code [1], and creating flexible production lines [2] has received a lot of attention. State of the art methods of today are helping process engineers finding short and fast robot trajectories, solving high dimensional path planning [3,4], scheduling [5,6] and workload distribution [7] problems. In academia more robust and effective algorithms are being developed [8].

The workflow in industry can still be further improved by improving the trajectories and removing any manual steps required. An example of such manual step was addressed in paper [9], where the goal was to remove the manual task of choosing zone radii for a given piecewise linear trajectory with via points. The problem was set up using variables that can be used directly as parameters in the set of available robot controller functions, which makes it possible to directly export the solutions to robot code. Only part of the available variable freedom was used in the optimization, since the initial via points were fixated and the only parameters that affected the geometrical shape were the zone radii.

The contribution of this work is to further improve the solutions by using non-fixated via points, giving the optimization algorithm a larger search space. This is achieved by reparameterizing the problem using piecewise polynomial functions, improving the robustness of the trajectory parametrization. The limited number of control variables available as robot controller commands is still used, so that solutions can be directly exported to robot code. An approximation of the energy consumed by the robot is also used to study the potential of offline energy optimization of robot trajectories.

2. Method

The method and optimization algorithm presented in this paper is a development and reformulation of the work presented in Gleeson et al.[9], which in turn is largely based on the ideas presented in Björkenstam et al.[10]. Summarizing the contin-

doi:10.1016/j.procir.2016.02.099

uous problem formulation in general terms, we have an optimal control problem (1) of finding the control signal u, which minimizes the cost functional J, composed of initial and final costs Φ, as well as running costs described by the Lagrangian L. Furthermore, the control and state should fulfill dynamic constraints (1b), as well as equality (1c) and inequality (1d) constraints.

$$\min_u J = \Phi(x(t_a), t_a, x(t_b), t_b) + \int_{t_a}^{t_b} L(x(t), u(t), t)dt \tag{1a}$$

$$\text{such that} \quad \dot{x}(t) = f(x(t), u(t), t) \tag{1b}$$

$$g(x(t), u(t), t) \geq 0 \tag{1c}$$

$$H(x(t_a), t_a, x(t_b), t_b) = 0 \tag{1d}$$

for $t \in [t_a, t_b]$.

The discretization method and trajectory parametrization used in [9] is also used here, but the problem is reformulated to decrease the number of variables and make the problem easier for the optimization algorithm. Since an interior point algorithm, the Ipopt solver developed by Wächter and Biegler[11], is used to solve the resulting optimization problem, a feasible initial guess will in general reduce the number of steps and the time it takes to convergence to an optimal solution. The variables in the new formulation are more physical and easier to use when an initial feasible point is to be set up. The initial point makes use of the solution given by the path planning algorithm developed by Bohlin and Kavraki[4], which is a piecewise linear collision free trajectory.

The benefit of using this trajectory parametrization is that the reduced convergence issues make it possible to relax the constraints on the via points and allow the optimizer to use a larger part of the search space. With this increased flexibility comes also the possibility of considering other objective functions. Time optimization is used here, retaining the correspondence to the trajectories produced by the robot controllers of today. An energy consumption model is used to compare the solutions to each other and give an indication of how large the potential is for energy reductions. To include energy optimization would require an outer optimization loop, and this will be the focus of future work. Still, optimizing with respect to time will smooth the trajectory, giving noticeable energy reduction.

2.1. Parametrizing the trajectory

To be able to generate robot code for a specific trajectory, a number of variables will have to be defined to specify the robot path. The overall structure of a robot path is defined by its initial point q_{start}, final point q_{end} and via points it should reach between them. These points are vectors of joint values, with typically six joints for a standard industrial robot. The via point joint vectors are denoted q_{mid} as they define the midpoint of a via point zone. For each via point, a zone radius defines an area where the robot is allowed to deviate from the otherwise piecewise linear path, smoothing out sharp corners and making it possible to maintain a velocity through the transition between linear segments.

A simplified sketch of a robot trajectory in joint space can be seen in Fig. 1 along with some notations used to describe

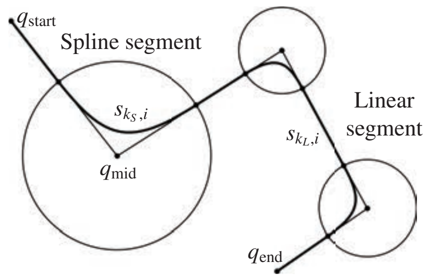

Fig. 1: Notations for linear and spline segments along the trajectory. The parametrization parameters, $s_{k,i}$ define N positions in each segment.

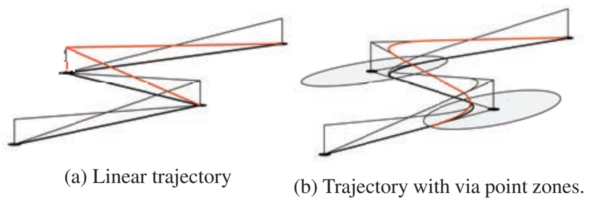

Fig. 2: The two figures show the trajectory and corresponding polynomials for a linear trajectory and a spline trajectory defined by via point zones. For the spline trajectory the corresponding polynomials of each via point is non zero within neighbouring via point zones.

the variables and different parts of the trajectory. Each segment is parametrized by a parameter in a unit length interval $s_{k,i} \in [k, k + 1]$. For the seven segments of the trajectory $k \in \{0, ..., 6\}$ seen in the figure, variables are specifically shown for the spline segment $k_S = 1$ and the linear segment $k_L = 4$. The shape of robot trajectories used here is the same as was used in [9], which have been found to correspond very well to the interpolated joint trajectories used by ABB robot controllers. Similar trajectories are also used by other industrial robot manufacturers (e.g. KUKA) even if there are minor differences.

In [9], the parametrization of the trajectory was divided into linear phases and spline phases, and the starting point and end point of each phase had to be defined and linked to the neighbouring phases. Points along the trajectory within each phase are then defined using these starting points and end points, as well as the via points of the spline phases. But as previously stated, it is only the via points and zone sizes that define the shape of the path. Here we instead parameterize the trajectory directly from these variables without explicitly defining the coordinates of the transitions between linear segment and spline segment. In order to do this we have to specify how each via point affects the position of points along the trajectory by finding and composing the polynomials that make up the trajectory. The parametrization of the trajectory will then consist of a summation over via point-vectors q_i and corresponding polynomial functions p_i:

$$q(s) = \sum_i q_i p_i. \tag{2}$$

The polynomial $p_i(s, r_A, r_B, \ell)$ will be a function of the trajectory parameter s, the zone sizes r_A and r_B, and the distance